Heiko Müller, Melanie Weis, Jens Bleiholder, Ulf Leser

Erkennen und Bereinigen von Datenfehlern in naturwissenschaftlichen Daten

Naturwissenschaftliche Daten sind aufgrund ihres Entstehungsprozesses oft mit einem hohen Maß an Unsicherheit behaftet. Bei der Integration von Daten aus verschiedenen Quellen führen diese Unsicherheiten, neben der vielfältigen syntaktischen und semantischen Heterogenität in der Repräsentation von Daten, zu Konflikten, die in einer verringerten Qualität des integrierten Datenbestandes münden. Obwohl Konflikte oftmals nur durch Domänenexperten endgültig aufgelöst werden können, kann und muss die Arbeit dieser Experten durch geeignete Werkzeuge unterstützt werden. Im folgenden Artikel stellen wir drei solcher Werkzeuge vor, die zurzeit in verschiedenen Forschungsgruppen an der Humboldt-Universität zu Berlin entwickelt werden.

1 Einleitung

Naturwissenschaftliche Daten sind typischerweise, neben den Objekten des betrachteten Diskursbereichs, Ergebnisse von Experimenten (Messungen) oder aus solchen Messungen abgeleitete Informationen. In der Genomforschung sind experimentelle Daten zum Beispiel die reinen DNA-Sequenzen. Durch Genomvergleich, statistische Analyse und Ähnlichkeitssuchen werden dann vielfältige Eigenschaften von Sequenzabschnitten abgeleitet, wie die Vorhersage von Genen und deren Funktion, funktionelle Abschnitte von Proteinen oder vermutete Protein-Protein-Wechselwirkungen. Sowohl die experimentellen Ausgangsdaten als auch die Analyseergebnisse werden in Datenbanken abgelegt [Galperin 2005].

1.1 Qualität naturwissenschaftlicher Daten

Die Qualität naturwissenschaftlicher Daten ist abhängig vom Typ der Daten und vom eigentlichen Datenerzeugungsprozess. Im Bereich der Genomdaten liegen die geschätzten Fehlerraten zwischen 0,01% und 2% für die experimentelle Bestimmung von Genomsequenzen und bis zu 50% für abgeleitete Informationen über strukturelle oder funktionale Eigenschaften der Sequenzen [Makarov 2002] [Müller et al. 2003]. Das größte Hindernis hinsichtlich einer erfolgreichen wirtschaftlichen oder wissenschaftlichen Nutzung der Daten sind Fehler, die bezüglich der erwarteten Repräsentation eine ungenaue oder fehlerhafte Abbildung realer Sachverhalte darstellen. Mit dem Erkennen und Bereinigen solcher Fehler werden wir uns im Folgenden befassen.

Die Ursachen fehlerhafter Daten sind im Prozess der Datengewinnung zu sehen. Die Durchführung von Experimenten birgt generell die Gefahr von Fehlern im Versuchsaufbau oder systematischen Fehlern während der Versuchsdurchführung, die unentdeckt bleiben. Gerade biotechnische Experimente, die meistens mit Hilfe lebender Organismen durchgeführt werden, erzeugen eine inhärente und nur durch kostspielige Mehrfachdurchführung von Experimenten zu begegnende Unsicherheit. So wurde jede Base des Human-Genome-Projektes im Durchschnitt sechsmal sequenziert [Dennis & Gallagher 2002], um die angestrebte Genauigkeit von 99,99% zu erreichen; andere Sequenzierprojekte erzeugen dagegen Daten mit wesentlich höheren Fehlerraten. Da die molekularbiologische Forschung viele Vorgänge in Genomen bzw. Organismen noch nicht versteht, sind abgeleitete Informationen oftmals das Ergebnis von »*Best efforts*« und daher mit einer schwierig zu schätzenden Fehlerrate behaftet. Die Vorhersage von Genen gelingt beispielsweise für Prokaryonten (Einzeller ohne eigenen Zellkern) aufgrund der einfacheren Genomstruktur mit sehr hoher Genauigkeit, hat sich für höhere Lebewesen wie auch den Menschen aber als ungleich schwieriger herausgestellt und erreicht in der Frage der exakten Bestimmung von Genen nur eine Genauigkeit von um die 50% [Makarov 2002]. Darüber hinaus beruhen Analysen oftmals auf veralteten oder unvollständigen Daten sowie auf anderen wiederum unsicheren Analyseergebnissen.

1.2 Möglichkeiten zur Verbesserung der Datenqualität

Fehlerhafte Daten lassen sich mit Hilfe von Integritätsbedingungen identifizieren. Diese formulieren in einer gegebenen Sprache unter Ausnutzung von Domänenwissen Bedingungen, die von korrekten Datenbankinstanzen erfüllt werden müssen. Bei der Datenbereinigung werden die Daten so lange manipuliert, bis sämtliche Integritätsbedingungen erfüllt sind. Dies wird als *constraint repair problem* bezeichnet und erfolgt entweder durch Löschen und Einfügen ganzer Datensätze [Embury et al. 2001] oder durch Modifikation der Werte [Bohannon et al. 2005].

Im naturwissenschaftlichen Bereich und insbesondere in der Genomforschung wird die Formulierung solcher Integritätsbedingungen durch unsicheres und unvollständiges Domänenwissen und durch eine Vielzahl an Ausnahmen erschwert. Außerdem sind solche Bedingungen meist nur sehr grobe Einschränkungen des gültigen Wertebereichs, und es können bei weitem nicht alle Fehler aufgedeckt werden. Zum anderen berücksichtigen Ansätze zur automatischen Bereinigung Domänenwissen nur unzureichend, weshalb die Qualität des resultierenden Datensatzes von der Güte der Integritätsbedingungen abhängig ist.

Wie oben bereits erwähnt, lässt sich eine zuverlässige Verbesserung der Qualität naturwissenschaftlicher Daten hinsichtlich Korrektheit durch Wiederholung oder durch zusätzliche Experimente erlangen. Dabei entsteht eine Reihe von unterschiedlichen Datensätzen, die dasselbe Objekt beschreiben. Aufgrund unterschiedlicher Rahmenbedingungen oder experimenteller Methoden können Widersprüche in den Daten auftreten, zu deren Auflösung zum Beispiel Wissen über die Stärken und Schwächen der experimentellen Verfahren, der jeweiligen Arbeitsgruppen und Rahmenbedingungen oder statistische Größen wie Wertehäufigkeiten benutzt werden können. Das Resultat ist eine widerspruchsfreie Vereinigung der experimentellen Ergebnisse mit einer daraus resultierenden hohen Qualität.

Die Ausführung zusätzlicher Experimente ist aber vergleichsweise teuer und unter Umständen auch unmöglich. Im naturwissenschaftlichen Bereich kommt es uns dabei zugute, dass weltweit unterschiedliche Arbeitsgruppen oftmals eine teilweise oder vollständig überlappende Menge an Objekten untersuchen, wie z.B. eine bestimmte Proteinfamilie oder das gleiche Genom. Es handelt sich dabei sowohl um die Ausführung von Experimenten als auch um die überlappende Auswertung experimenteller Ergebnisse. Dies erfolgt häufig unter Ausnutzung unterschiedlicher Techniken und Reagenzien. Somit stehen oftmals genügend überlappende Datensätze bereit. In einigen Fällen können unter Ausnutzung von Domänenwissen die benötigten Daten aus existierenden Daten abgeleitet werden, wie dies in [Müller 2003] zur Korrektur fehlerhaft annotierter Startpositionen der Abschrift von DNA-Molekülen, als erster Schritt der Proteinsynthese, unter Verwendung existierender Proteindaten beschrieben wurde. Aufgrund dieser Umstände ist die Integration unter Ausnutzung der existierenden Redundanzen ein aussichtsreicher Ansatz zur Erkennung und Bereinigung von Datenfehlern in naturwissenschaftlichen Daten [Hardison 2003].

1.3 Qualitätsverbesserung mittels Datenintegration

Im Rahmen der Datenintegration zum Zwecke der Qualitätsverbesserung geht es zunächst darum, die jeweiligen Datensätze zu identifizieren, die dasselbe Objekt beschreiben. Zwischen diesen können dann Widersprüche in der Objektbeschreibung sichtbar gemacht werden. Wir betrachten hier nur Konflikte auf Datenebene und nicht auf Schemaebene. Widersprüchliche Werte sind zunächst als potenzielle Fehler anzusehen. In [Brenner 1999] wird auf diese Weise aufgezeigt, dass für mindestens 8% der vorhergesagten Gene des Genoms von *Mycoplasma genitalium* die funktionellen Beschreibungen als zweifelhaft anzusehen sind, da in diesen Fällen unterschiedliche Arbeitsgruppen zu widersprüchlichen Ergebnissen gekommen sind.

Nachdem potenzielle Fehler identifiziert wurden, müssen gezielt die qualitativ besseren Werte ausgewählt werden. Hierzu wird domänenspezifisches Expertenwissen benötigt. Unter der Annahme, dass einer der vorhandenen Werte wirklich besser ist und man ihn mit Hilfe des Expertenwissens identifizieren kann, gilt es nun, den Experten in der Formulierung entsprechender Regeln zu unterstützen. Hierzu kann man mit Verfahren des Data Mining Informationen zu möglichen Konfliktursachen liefern. Diese Informationen kann der Experte dann ausnutzen, um systematische Unterschiede zu erkennen, sie zu verstehen und gezielt Regeln zur Bestimmung der korrekten Werte zu spezifizieren. Die Formulierung dieser Regeln sollte wieder durch geeignete Spezifikationssprachen unterstützt werden.

Für die genannten drei Fragestellungen beschreiben wir in dem folgenden Artikel Algorithmen bzw. Sprachen. Der Artikel ist wie folgt gegliedert: Wir zeigen zunächst anhand eines einfachen Beispiels Eigenschaften und Besonderheiten naturwissenschaftlicher Daten. Danach beschreiben wir die einzelnen Phasen der Datenintegration zum Zweck der Qualitätssteigerung. In Abschnitt 3 wird die Duplikaterkennung beschrieben, in Abschnitt 4 Verfahren zur Aufdeckung potenzieller Fehlerursachen und in Abschnitt 5 Sprachen zur Formulierung von Konfliktlösungsstrategien. In Abschnitt 6 fassen wir unsere Ergebnisse zusammen.

2 Naturwissenschaftliche Daten

Der Begriff der naturwissenschaftlichen Daten ist weit gefächert. Eine generelle Eigenschaft der Daten ist die bereits beschriebene unabhängige Datenerzeugung durch unterschiedliche Arbeitsgruppen mit unterschiedlichen Methoden. Die Lösung der dabei auftretenden Heterogenitäten, wie unterschiedliche Datenmodelle oder -formate, ist Bestandteil einer Vielzahl von Forschungsaktivitäten und wird von uns hier nicht weiter behandelt. Wir haben bisher Genomdaten als einen typischen Repräsentanten naturwissenschaftlicher Daten verwendet. Zur Veranschaulichung betrachten wir im Folgenden ein vereinfachtes Szenario aus dem Bereich der Vogelkunde (Ornithologie). Dieses erlaubt uns die weitestgehende Abstraktion von domänenspezifischen Eigenschaften, ermöglicht aber dennoch die Beschreibung der wesentlichen Dateneigenschaften, der gewählten Vorgehensweise und der verwendeten Methoden.

In vielen Regionen setzen sich Forscher mit dem Bestand (bedrohter) Tierarten auseinander. Dabei ist ein wichtiger Schritt die Erfassung des korrekten Bestandes, wie dies z.B. bei Greifvögeln oder Eulen der Fall ist [LV Eulenschutz 2004]. Je nachdem auf welche Art und Weise die Vögel erfasst werden (von weitem oder aus der Nähe beobachtet, gefangen usw.), sind die dabei aufgenommenen Daten mehr oder weniger präzise. Bei Arten, die in ihrem Bestand schon so weit bedroht sind, dass es nur eine überschaubare Anzahl an Individuen gibt (z.B. beträgt der Bestand an Uhus in der Schweiz nur ca. 200 Stück), ist es sinnvoll, einzelne Tiere zu unterscheiden. Eine Markierung durch Ringe oder Chips ist oftmals nicht möglich oder nicht erwünscht. Ein Ausschnitt möglicher Daten zur Erfassung des Eulenbestandes einer virtuellen Region ist in Tabelle 1 gezeigt.

Bez.	Gattung	Art der Beobachtung	Spannweite in cm	Größe	Gewicht	Gefieder Muster	Gefieder Farbe	Beobachter	Geschlecht
1	Bubo Scandiacus	gemessen	153	58	1675 g	gescheckt	weiß	Klaus	m
2	B. Scandiacus	geschätzt	150	60	1.7 kg	gefleckt	schnee-weiß	Harry	⊥
3	Pseudocops	gemessen	118	53	2320 g	gestreift	braun	Klaus	⊥
4	Kauz	gemessen	118	55	0.622 kg	gescheckt	hell-braun, grau	Klaus	w
5	Pseudoscops	beobachtet	120	50	2500 g	gestreift	braun	Peter	⊥
6	Bubo Bubo	geschätzt	110	45	2300 kg	gefleckt	dunkelbraun, grau	Harry	w
7	Bubo Bubo	beobachtet	110	45	2.3 kg	gefleckt	dunkelbraun, grau	Peter	1
8	Kauz	geschätzt	120	55	600 g	gefleckt	hell-braun, grau	Harry	0

Tab. 1: Beispiel einer Datentabelle zur Erfassung des Eulenbestandes einer unbestimmten Region

Wie in anderen Bereichen auch, ist den Objekten kein einheitlicher globaler Bezeichner (Schlüssel) zu ihrer Identifizierung und Unterscheidung zu Eigen. Es gibt diese höchstens innerhalb der einzelnen messenden Personengruppen, aber nicht gruppenübergreifend. Daraus resultieren schwierig zu erkennende Duplikate, d.h. unterschiedliche Datensätze, die dasselbe Objekt der realen Welt repräsentieren. In unserem Beispiel werden die Duplikate unfreiwillig dadurch erzeugt, dass unterschiedliche Beobachter in der gleichen Region unterwegs sind und möglicherweise die gleichen Individuen aufzeichnen. Wenn diese keine Fußringe oder ähnliche Markierungen tragen, muss man versuchen, sie anhand der beobachteten Eigenschaften voneinander zu unterscheiden. Widersprüche entstehen durch die unterschiedlichen Hauptaugenmerke oder Fähigkeiten der Beobachter und die von ihnen verwendeten Methoden.

In unserem Beispiel seien die Eulen 1 und 2, 3 und 5, 6 und 7 sowie 8 und 4 Duplikate. Bereits am Eulenpaar 1 und 2 ist eine Reihe von Widersprüchen zu erkennen. So werden zum Beispiel verschiedene Repräsentationen oder Bezeichnungen der Gattung und der Farbe verwendet. Die Messungen der Spannweite, der Größe und des Gewichts weisen ungleiche Werte auf. Diese können auf die abweichende Präzision der verwendeten Methoden zurückgeführt werden. Ein weiterer Unterschied hinsichtlich der Repräsentation ist in der Verwendung verschiedener Maßeinheiten bei der Gewichtsangabe zu erkennen. Zur Beschreibung der Musterung sind die Synonyme *gefleckt* und *gescheckt* verwendet worden. Im Feld Geschlecht bestehen Unterschiede, da manche Beobachter das Geschlecht gar nicht aufführen.

Wir hatten bereits mögliche Fehlerursachen in naturwissenschaftlichen Daten angesprochen. Im Folgenden unterscheiden wir zwischen zufälligen und systematischen Fehlern. Zufällige Fehler entstehen durch willkürliche bzw. nicht reproduzierbare Mess- bzw. Beobachtungsfehler. Hierzu zählen auch Messungenauigkeiten. Systematische Fehler sind dagegen nachvollziehbar, d.h., sie resultieren aus einem systematischen (Fehl-) Verhalten, das sich in Form von Regeln (in einer geeigneten Sprache) beschreiben lässt. Beispiele hierfür sind unterschiedliche Ansprüche an die Messgenauigkeit oder unterschiedlich normierte Messdaten. Die im Rahmen der Konfliktauflösung verwendeten Lösungsstrategien resultieren im Idealfall aus der Kenntnis möglicher Fehlerursachen.

2.1 Klassifikation der Attribute

Gerade im wissenschaftlichen Bereich ist eine Besonderheit hinsichtlich der Erkennung von Duplikaten und der Definition von Konflikten zu beachten. Neben den eigentlichen Objekteigenschaften, d.h. den Mess- und Beobachtungswerten, werden oftmals auch administrative Daten zu einem Objekt und dem Messverfahren gespeichert (Metadaten). Metadaten sind in unserem Beispiel die Attribute *Bezeichner*, *Art der Beobachtung* und *Beobachter*. Weitere typische Beispiele wären Datumsangaben bzgl. der Erstellung und Modifikation von Einträgen.

Konflikte in Metadaten sind natürlich und müssen nicht aufgelöst werden, spielen aber oft eine große Rolle bei der Auflösung von Konflikten in den anderen Daten. Im Rahmen der Datenintegration ist daher vorab eindeutig zu klären, welches die Attribute sind, die ein Objekt beschreiben und in denen Widersprüche relevant sind. Wir nennen diese Attribute die Objektbeschreibungen. In unserem Beispiel sind dies die Attribute *Gattung, Spannweite in cm, Größe, Gewicht, Gefiedermuster, Gefiederfarbe* und *Geschlecht*.

3 Duplikaterkennung

Der erste Schritt beim Finden und Lösen von Datenkonflikten besteht darin, die Datensätze zu identifizieren, die zwar in ihren Werten verschieden sind, aber letztendlich das gleiche Objekt der realen Welt beschreiben. Dies wird im Allgemeinen als Duplikaterkennung bezeichnet. In unserem Beispiel gilt es, während der Duplikaterkennung herauszufinden, dass die acht Tupel in der Tabelle in Wirklichkeit nur vier verschiedene Eulen darstellen.

Wir gehen auf folgende Kernprobleme der Duplikaterkennung ein, gemäß dem in [Weis & Naumann 2005] beschriebenen Framework für Duplikaterkennung:

- Welche Informationen beschreiben ein Objekt und sind demnach für die Duplikaterkennung von Nutzen?
- Wie unterscheiden wir Duplikate von Nichtduplikaten?
- Welcher Algorithmus wird verwendet, um Duplikate effizient, effektiv und auf großen Datenmengen zu identifizieren?

3.1 Objektbeschreibung

Die Erkennung von Duplikaten beruht auf dem paarweisen Vergleich von Tupeln bezüglich ihrer Objektbeschreibungen. Der erste Schritt ist deshalb die Festlegung der interessierenden Attribute, wie dies im vorherigen Abschnitt erfolgt ist.

Im Rahmen unserer Forschung haben wir zusätzlich Methoden entwickelt, die Objektbeschreibungen anhand eines hierarchischen XML-Schemas automatisch bestimmen können [Weis & Naumann 2005]. Diese können auf naturwissenschaftliche Daten, die oftmals in XML vorliegen (z.B. SwissProt), angewendet werden. Des Weiteren können die Methoden auch auf dem relationalen Modell angewendet werden, wenn das Schema aus mehreren Tabellen besteht, die durch Fremdschlüsselbeziehungen miteinander verknüpft sind und somit eine hierarchische Struktur aufweisen. Allerdings sind die Methoden nur eingeschränkt in der Lage, Objekte innerhalb einer Tabelle zu identifizieren (z.B. Eulen in der Tabelle Eulenbeobachtung) und deren Beschreibung zu ermitteln. In diesen Fällen muss ein Domänenexperte die Objekte und deren Beschreibung herausfinden.

3.2 Duplikatklassifizierung

Duplikate werden identifiziert, indem Objekte anhand ihrer Beschreibungen paarweise verglichen werden. Wir beschränken unsere Diskussion auf ähnlichkeitsbasierte Duplikaterkennung, die anhand eines gegebenen Ähnlichkeitsmaßes und eines Ähnlichkeitsschwellwerts Objektpaare als Duplikate klassifiziert, wenn ihre Ähnlichkeit über dem Schwellwert liegt.

Definition: Gegeben seien zwei Tupel t_1 und t_2, ein Ähnlichkeitsmaß $sim(t_1, t_2)$ und ein Schwellwert s_{dup}, so gilt:

$sim(t_1, t_2) > s_{dup} \Rightarrow t_1, t_2$ sind Duplikate.

Die Qualität des Ergebnisses hängt demnach vom Ähnlichkeitsmaß ab. Wir unterscheiden domänenspezifische und domänenunabhängige Ähnlichkeitsmaße. Der Vorteil domänenspezifischer Klassifizierer ist, dass sie von einem Experten der Anwendungsdomäne, in unserem Fall z.B. ein Ornithologe, entwickelt werden. Dies ist jedoch mühselig. Domänenunabhängige Klassifizierer lassen sich prinzipiell auf jede Domäne mit guten Ergebnissen anwenden [Monge & Elkan 1997, Ananthakrishna et al. 2002].

Ähnlichkeitsmaße basieren oft auf der Ähnlichkeit von Zeichenketten, gemessen durch den Editierabstand (*edit distance*). Dieser ist definiert als die minimale Anzahl von Lösch-, Einfüge- und Ersetzungsoperationen, die notwendig sind, um eine Zeichenkette in eine andere umzuwandeln. In diesem Fall gilt:

Definition: Gegeben sind zwei Werte a_1 und a_2 eines Attributes des Datentyps STRING, mit jeweiliger Länge l_1 und l_2 und ein Schwellwert $s_{ed} > 0$. Wir sehen a_1 und a_2 als gleichbedeutend an, dargestellt durch $a_1 \cong a_2$, wenn für ihren Editierabstand gilt:

$$ed(a_1, a_2) < s_{ed} * max(l_1, l_2)$$

Für numerische Datentypen ist der Editierabstand ungeeignet. Für solche Attribute benutzen wir die Abweichung.

Definition: Gegeben sind zwei Werte a_1 und a_2 eines numerischen Attributes und ein Schwellwert $s_{num} > 0$. Wir sehen a_1 und a_2 als gleichbedeutend an, wenn für ihre Abweichung gilt:

$$\delta(a_1, a_2) < s_{num} * max(a_1, a_2)$$

Werte, die nicht als gleichbedeutend anzusehen sind, bezeichnen wir als ungleich. Die Ermittlung der Schwellwerte s_{dup}, s_{ed} und s_{num} ist für die Effektivität der Duplikaterkennung entscheidend. Diese können durch Domänenexperten spezifiziert werden. Alternativ können sie durch Sampling oder maschinelles Lernen bestimmt werden.

Oftmals ist es sinnvoll, die *Inverse Document Frequency* (IDF) zu benutzen, um Attributwerte verschieden zu gewichten. Wir verwenden eine Abweichung der IDF, genannt *softIDF* (kurz *sIDF*), die nicht auf einzelnen Attributwerten, sondern auf Paaren von Attributwerten, die als ähnlich angesehen werden, basiert.

Definition: Sei T die Menge aller Tupel der betrachteten Relation. Seien T_1 die Menge aller Tupel, in denen Attribut A den Wert a_1 enthält, und T_2 die Menge aller Tupel, in denen Attribut A den Wert a_2 enthält. Wir definieren:

$$sIDF(a_1, a_2) = log\left(\frac{|T|}{|T_1 \cup T_2|}\right)$$

Für eine Menge von Attributpaaren definieren wir die *sIDF* einer Menge als die Summe der *sIDF* der einzelnen Attributpaare.

Definition: Gegeben zwei Tupel t_1 und t_2. Mit $t[A_i]$ bezeichnen wir den Wert, den Tupel t für Attribut A_i enthält. Wir stellen t_1 und t_2 als eine Menge von Wertepaaren über der Menge aller Attribute des den Daten zugrunde liegenden Schemas R dar:

$$\tau(t_1, t_2) = \bigcup_{A_i \in R} \{(t_1[A_i], t_2[A_i])\}$$

Die Menge A_{sim} gleichbedeutender Information für Tupel t_1 und t_2 ist dann definiert als:

$$A_{sim}(t_1,t_2) = \{(a_1,a_2) \in \tau(t_1,t_2) \mid (a_1 \cong a_2)\}$$

Die Menge ungleicher Information ist definiert als

$$A_{diff}(t_1,t_2)=\{(a_1,a_2) \in \tau(t_1,t_2) \mid \neg (a_1 \cong a_2)\}$$

Eine Sonderrolle nehmen fehlende Werte, die so genannten NULL-Werte, ein (im Beispiel dargestellt durch das Symbol $\perp$). Sind einer oder beide Werte eines Wertepaares NULL, so fließt dieses Paar generell nicht in die Berechnung der Ähnlichkeit mit ein, d.h., es ist weder in A_{sim} noch in A_{diff} enthalten.

Das Ähnlichkeitsmaß ist definiert als:

$$sim(t_1, t_2) = \frac{sIDF(A_{sim})}{sIDF(A_{diff}) + sIDF(A_{sim})}$$

Wir betrachten nun unser Eulenbeispiel mit folgenden Schwellwerten: s_{ed} = 0,2, s_{num} = 0,05 und s_{dup} = 0,7. Bei vereinfachender Annahme, jedes Attribut habe die gleiche Relevanz (*sIDF* = 1), berechnet sich *sim(Eule1, Eule2)* wie folgt:

Die ähnlichen Attribute sind *Gattung, Spannweite* und *Größe*. Als unterschiedlich werden *Gewicht, Gefiedermuster* und *Farbe* angesehen. Da *Geschlecht* bei *Eule2* nicht spezifiziert ist, geht das Attribut nicht in die Ähnlichkeit ein. Somit ergibt sich:

sim(*Eule1*, *Eule2*) = 3 / (3 + 3) = 1/2.

Demnach sind *Eule1* und *Eule2* keine Duplikate. Auf ähnliche Weise berechnen wir

sim(*Eule3*, *Eule5*) = 1, also Duplikate,
sim(*Eule6*, *Eule7*) = 5/7 (Duplikat) und
sim(*Eule8*, *Eule4*) = 4/7 (Nichtduplikat).

Unschwer zu erkennen ist, dass Fehler, die aufgrund von Synonymen oder Inkonsistenzen in der Wahl der Maßeinheiten (z.B. g, kg in *Gewicht*) entstehen, das Ähnlichkeitsmaß negativ beeinflussen. Durch zusätzliche Informationen wie Thesauri, Ontologien und Domänenwissen kann das Ähnlichkeitsmaß verfeinert werden. Durch solches zusätzliches Wissen kann zum Beispiel ermittelt werden, dass auch *Eule1* und *Eule2* Duplikate sind.

3.3 Effiziente Algorithmen

Die Anforderungen an die Duplikaterkennung sind sowohl Effektivität als auch Effizienz und Skalierbarkeit. Ersteres hängt ausschließlich vom Klassifizierer ab, wenn alle Objektpaare betrachtet werden. Doch bei großen Datenmengen, die üblicherweise bei naturwissenschaftlichen Daten erreicht werden, ist es unmöglich, alle Paare zu vergleichen. Um die Effizienz zu steigern, müssen teure Vergleiche eingespart werden. Dafür gibt es prinzipiell zwei Ansätze: exaktes Filtern und heuristisches Filtern.

Exaktes Filtern filtert Objektpaare, die ohnehin vom Klassifizierer als Nichtduplikate erkannt werden. Zu dieser Sorte Filter gehört zum Beispiel die obere Grenze eines Ähnlichkeitsmaßes. Für diese gilt:

Definition: Sei Filter $f(t_1, t_2)$ für das Tupelpaar t_1, t_2 so definiert, dass $f(t_1, t_2) \geq \text{sim}(t_1, t_2)$. Sei s_{dup} der Ähnlichkeitsschwellwert zum Ähnlichkeitsmaß $sim(t_1, t_2)$. Wenn $f(t_1, t_2) < s_{dup}$, dann gilt $sim(t_1, t_2) < s_{dup}$, so dass t_1, t_2 als Nichtduplikat gefiltert werden kann.

Zu dem von uns vorgestellten Ähnlichkeitsmaß haben wir in [Weis & Naumann 2005] folgenden Filter definiert:

Definition: Sei t_1 ein Tupel, A_{shared} die Menge aller Attributwerte, die in t_1 und anderen Tupeln für ein Attribut A gleichbedeutend vorkommen, und sei A_{unique} die Menge aller Attributwerte, die ausschließlich in t_1 gleichbedeutend vorkommen und in keinem Tupel NULL sind. Der Filter ist definiert als:

$$f(t_1) = \frac{sIDF(A_{shared})}{sIDF(A_{unique}) + sIDF(A_{shared})}$$

Die Besonderheit an diesem Filter ist, dass er nicht nur einzelne Objektpaare, sondern mehrere Objektpaare durch eine Berechnung einspart. Auf unser Beispiel angewendet, erhalten wir für

Eule1 $f(Eule1) = 4 / (2 + 4) = 4/6$, da das Tupel vier Attributwerte mit anderen Tupeln teilt (Gattung, Spannweite und Größe mit *Eule2* und Musterung mit *Eule4*) und zwei Attribute (Gewicht, Farbe) in keinem anderen Tupel vorkommen. Das Geschlecht wird nicht berücksichtigt, da es in manchen Tupeln NULL ist. Da $f(Eule1) < s_{dup}$ wird *Eule1* gefiltert, was insgesamt 7 Vergleiche spart. Auf gleiche Weise wird auch *Eule2* gefiltert, für die $f(Eule2) = 4/6$ gilt, was wiederum 6 Vergleiche spart. Die Implementierung des Filters stellt eine effiziente Ermittlung von A_{shared} und A_{unique} sicher. Dafür wird zum einen eine geeignete Graphstruktur verwendet, zum anderen wird die Anzahl der nötigen Attributvergleiche minimiert und durch geeignete Verfahren beschleunigt. Details werden in [Weis & Naumann 2004] beschrieben.

Ein weiterer exakter Filter für Objektpaare berechnet die Ähnlichkeit nur für Paare, die mindestens in einem Attribut ähnlich sind, da die Ähnlichkeit sonst 0 ist. Dieser Filter spart in unserem Beispiel 12 von insgesamt 28 Vergleichen.

Heuristisches Filtern ist weniger konservativ und filtert Vergleiche, die wahrscheinlich Duplikate ergeben, gibt dafür aber keine Garantie. Ein Beispiel dafür haben wir für XML-Daten bereits in [Weis & Naumann 2004] vorgestellt. Dort werden nur Duplikate zwischen Elementen gesucht, die gleiche Vorfahren haben, da angenommen wird, dass Duplikate im gleichen Kontext (die Vorfahren) vorkommen. Nehmen wir zum Beispiel an, dass Beobachtungen pro Region in XML geschachtelt werden. Da Eulen keine Zugvögel sind, ist es unwahrscheinlich, dass die gleiche Eule in verschiedenen Regionen beobachtet wird. Demnach genügt es, Duplikate zu Eulen innerhalb einer Region zu suchen. Diese Methode lässt sich, wie auch unsere Heuristiken zur Auswahl der Objektbeschreibung, auf relationale Daten, die durch Fremdschlüssel zusammenhängen, anwenden.

Für relationale Daten gibt es eine Vielzahl weiterer Methoden. Eine beliebte ist die *Sorted Neighborhood Methode* (SNM) [Hernández & Stolfo 1995], die aus drei Phasen besteht. In der ersten Phase werden für jedes Objekt Schlüssel aus den einzelnen Attributwerten der Objektbeschreibung generiert. In der SORT-Phase werden die Tupel nach diesem Schlüssel sortiert. In der MERGE-Phase wird über das sortierte Feld ein Fenster geschoben, und nur Tupel innerhalb des Fensters werden verglichen.

Nachdem Duplikaterkennung durchgeführt wurde, wird jedem Tupel eine ObjektID hinzugefügt. Tupel, die das gleiche Objekt beschreiben, bekommen die gleiche ObjektID zugewiesen. Demnach sehen (im idealen Fall) unsere Beispieldaten nach der Duplikaterkennung wie in Tabelle 2 aus. Dieses Ergebnis ist die Basis für den nächsten Schritt, die Konflikterkennung.

4 Erkennen von Konflikten und möglichen Ursachen

Die eindeutige Objektkennzeichnung, die den Tupeln als Ergebnis der Duplikaterkennung zugewiesen wurde, wird zu deren Gruppierung in so genannte Duplikatgruppen verwendet (Tab. 2). Jede dieser Duplikatgruppen kann aus einem oder mehreren Tupeln bestehen, die alle dasselbe Objekt repräsentieren. Jede Duplikatgruppe soll bei der Datenintegration zu einem einzigen Tupel zusammengefasst werden. Hierbei müssen vorhandene Konflikte und Widersprüche aufgelöst werden. Wir beschränken uns im Folgenden auf Duplikatpaare, d.h. die Gruppen aus zwei Tupeln. Dies entspricht dem Fall der Integration zweier Datenquellen.

4.1 Erkennen von Konflikten

Als Konflikt oder Widerspruch bezeichnen wir jene Situation, in der unterschiedliche Werte oder Zeichenketten zur Repräsentation desselben Sachverhalts bzw. derselben Eigenschaft verwendet werden. In Tabelle 2 sind die nach dieser Definition widersprüchlichen Werte unseres Beispiels grau hinterlegt.

Ein Sonderfall stellt die Situation dar, in der eines der Tupel eines Duplikatpaares einen NULL-Wert verzeichnet, während im anderen Tupel ein konkreter Wert steht. Wir bezeichnen diese Situation als Unsicherheit. Im Folgenden werden Konflikte und Unsicherheiten in einem Framework behandelt.

Konflikte in dem hier definierten Sinne resultieren nicht immer aus Fehlern. Konflikte in Metadaten etwa sind keine Fehler, sondern reale Unterschiede (sie sind deshalb in Tabelle 2 schraffiert dargestellt). Die Konflikte innerhalb der Objektbeschreibungen können wir grob in syntaktische und semantische Konflikte unterteilen. Erstere resultieren aus der Verwendung unterschiedlicher Bezeichnungen (Synonyme) oder unterschiedlicher Maßeinheiten zur Repräsentation desselben Sachverhalts. Bei korrekter Interpretation stellen sie dennoch die gleiche Information dar. Semantische Konflikte beziehen sich hingegen auf inhaltliche Fehler, d.h. unterschiedliche Information zum selben Sachverhalt.

OID	Bez	Gattung	Art der Beobachtung	Spannweite in cm	Größe	Gewicht	Gefieder Muster	Gefieder Farbe	Beobachter	Geschlecht
1	1	Bubo Scandiacus	gemessen	153	58	1675 g	gescheckt	weiß	Klaus	m
1	2	B. Scandiacus	geschätzt	150	60	1.7 kg	gefleckt	schnee-weiß	Harry	⊥
2	3	Pseudocops	gemessen	118	53	2320 g	gestreift	braun	Klaus	⊥
2	5	Pseudoscops	beobachtet	120	50	2500 g	gestreift	braun	Peter	⊥
3	6	Bubo Bubo	geschätzt	110	45	2300 kg	gefleckt	dunkelbraun, grau	Harry	w
3	7	Bubo Bubo	beobachtet	110	45	2.3 kg	gefleckt	dunkelbraun, grau	Peter	1
4	4	Kauz	gemessen	118	55	0.622 kg	gescheckt	hell-braun, grau	Klaus	w
4	8	Kauz	geschätzt	120	55	600 g	gefleckt	hell-braun, grau	Harry	0

Tab. 2: Ergebnis der Duplikaterkennung mit nachfolgender Gruppierung

4.2 Mustersuche in widersprüchlichen Daten

Nach der Identifikation von Konflikten ist der nächste Schritt ihre Auflösung. Da eine automatische Konfliktauflösung nur in Ausnahmefällen möglich ist, wird die Unterstützung eines Domänenexperten benötigt. Ziel unserer Arbeiten im Bereich des Suchens von Mustern in widersprüchlichen Daten (*contradiction pattern mining*) ist es, dem Experten ergänzende Informationen zu einzelnen Konflikten zu liefern, die auf die potenzielle Konfliktursache schließen lassen und so eine qualitative Bewertung der widersprüchlichen Werte ermöglichen. Dies ist ein wichtiger Aspekt einer gezielten Konfliktlösung, bei der man verhindern will, dass alle Konflikte gleich behandelt werden (müssen).

Im Folgenden beschränken wir uns auf systematische Fehler. Die ihnen zugrunde liegenden Systematiken können als Muster in den widersprüchlichen Daten dargestellt bzw. gefunden werden. Ausgehend von diesen Mustern versucht man dann, Rückschlüsse auf mögliche Konfliktursachen zu ziehen. Als Muster bezeichnen wir Regelmäßigkeiten, die sich wie folgt ausdrücken lassen:

> WENN *Bedingung* DANN *Konfliktklasse*

Die Bedingung tritt dabei gehäuft in Zusammenhang mit einer bestimmten Konfliktklasse auf und ihre Evaluierung liefert dem Experten Hinweise auf mögliche Konfliktursachen. Eine einfache Konfliktklasse ist zum Beispiel das Vorkommen eines Konfliktes in einem bestimmten Attribut. In Tabelle 2 kann man erkennen, dass Konflikte der Klasse »*Konflikt in Spannweite*« immer dann auftreten, wenn einer der Beobachter gemessen hat, während der zweite Beobachter geschätzt oder beobachtet hat. Abhängig vom Augenmaß und den verwendeten Messinstrumenten haben wir hier unterschiedlich genaue experimentelle Methoden, durch die die Unterschiede in den gemessenen Werten zu erklären sind. Dies lässt sich durch folgende Regelmäßigkeit ausdrücken:

> WENN *Art der Beobachtung = gemessen/geschätzt* DANN *Konflikt in Spannweite*

Für das Attribut Größe gilt dies nur in zwei von drei Fällen, da die Methoden nicht immer zwingend zu unterschiedlichen Ergebnissen kommen. Aus diesem Grund müssen bei der Mustersuche auch Ausnahmen zugelassen sein. Andere Konfliktklassen beziehen sich direkt auf einzelne Werte oder Wertepaare, die in Konflikten vorkommen. In unserem Beispiel steht die Konfliktklasse *Gefiedermuster = gescheckt/gefleckt* in direktem Zusammenhang mit der Tatsache, dass einer der Beobachter Harry und der andere Klaus war. Eine nahe liegende Schlussfolgerung ist hier die synonyme Verwendung der Begriffe gescheckt und gefleckt durch die beiden Beobachter.

Für die Suche nach Mustern der ersten Konfliktklasse haben wir in [Müller et al. 2004] einen Algorithmus vorgestellt. Dieser baut auf existierenden Methoden des Data Mining auf und ist auf zwei Datenquellen ausgelegt. Er soll im Folgenden kurz beschrieben werden. Am Ende des Abschnitts gehen wir kurz auf mögliche Anpassungen für die hier beschriebene Situation mit einer Quelle ein.

Die Grundidee zeigt Abbildung 1. Gegeben sind zwei Datenquellen r_1 und r_2 mit identischem Schema, die einen Ausschnitt der Daten aus unserem Beispiel darstellen (aus Gründen der Übersichtlichkeit sind die aufgeführten Attribute Art der Beobachtung, Größe und Gefiedermuster mit A_1, A_2 und A_3 abgekürzt worden). Über die *OID* werden die Duplikate zwischen den Quellen erkannt. Die einzelnen Quellen sind frei von Duplikaten. Die Quellen werden dann über einen Join auf der *OID* vereint. Zur Vereinfachung des Algorithmus wird die resultierende Tabelle r_m um einen Konfliktindikator (*CI*) pro Objekteigenschaft erweitert. Die Werte der Konfliktindikatoren zeigen an, ob in einem Duplikat ein Konflikt in dem entsprechenden Attribut vorliegt oder nicht.

Für jeden Konfliktindikator können z.B. mit Hilfe von Entscheidungsbäumen [Breiman et al. 1984] oder Assoziationsregeln [Agrawal & Srikant 1994] mit fester Konsequenz [Liu et al. 1998] Klassifikationsregeln gelernt werden. In [Webb et al. 2003] werden für eine ähnliche Problemstellung insgesamt drei Ansätze miteinander verglichen. Dabei wird die Suche nach Assoziationsregeln als die am besten geeignete Methode beschrieben. Für eine effiziente Suche nach Fehlermustern muss das ursprüngliche Verfahren zur Erkennung von Assoziationsregeln allerdings erheblich modifiziert werden.

Unsere Fehlermuster haben alle die Form:

> WENN Bedingung DANN $CI_i = 1$

Definition: Die Bedingung ρ ist eine Konjunktion von Termen der Form $r_1[A_j] = x$ bzw. $r_2[A_j] = x$ sowie $CI_k = 1$ oder 0. Die Länge einer Bedingung bezeichnet die Anzahl der in ihr enthaltenen Terme.

Definition: Für jede Objekteigenschaft A_i kann die vereinte Relation anhand des Konfliktindikators CI_i in zwei disjunkte Teilmengen aufgeteilt werden. Mit CA_i bezeichnen wir dabei die Teilmenge der Tupel, für die $CI_i = 1$ gilt, und mit NA_i die Teilmenge der Tupel, für die $CI_i = 0$ gilt. Mit $|CA_i|$ bzw. $|NA_i|$ bezeichnen wir die Anzahl der Tupel in den einzelnen Mengen. In Abbildung 1 umfasst CA_i für A_2 das Tupel mit der $OID = 1$ und NA_i in diesem Fall das Tupel mit der $OID = 4$. Für A_3 ist NA_i leer, während CA_i beide Tupel aus r_m umfasst.

r_1

OID	A_1	A_2	A_3
1	gemessen	58	gescheckt
2	gemessen	53	gestreift
4	gemessen	55	gescheckt

r_2

OID	A_1	A_2	A_3
1	geschätzt	60	gefleckt
3	geschätzt	45	gefleckt
4	geschätzt	55	gefleckt

r_m

OID	$r_1[A_1]$	$r_2[A_1]$	CI_1	$r_1[A_2]$	$r_2[A_2]$	CI_2	$r_1[A_3]$	$r_2[A_3]$	CI_3
1	gemessen	geschätzt	1	58	60	1	gescheckt	gefleckt	1
4	gemessen	geschätzt	1	55	55	0	gescheckt	gefleckt	1

Abb. 1: Beispiel zum beschriebenen Algorithmus zur Suche nach Fehlermustern

Zur Festlegung der uns interessierenden Muster haben wir zwei Maße eingeführt, die sich an den für Assoziationsregeln existierenden Maßen *support* und *confidence* orientieren.

Definition: Die Konfliktrelevanz ist die relative Häufigkeit, mit der die Bedingung ρ in CA_i vorkommt. Das Konfliktpotenzial beschreibt den Anteil des Vorkommens der Bedingung ρ in CA_i bezogen auf deren Vorkommen in der Gesamtrelation. Bezeichne $|\rho(CA_i)|$ bzw. $|\rho(r_m)|$ die Anzahl der Tupel in CA_i bzw. r_m, die die Bedingung ρ erfüllen, so gilt:

$$Konfliktrelevanz = \frac{|\rho(C_{Ai})|}{|C_{Ai}|}$$

$$Konfliktpotenzial = \frac{|\rho(C_{Ai})|}{|\rho(r_m)|}$$

Der Algorithmus erhält als Parameter die geforderten Schwellwerte für Konfliktpotenzial und Konfliktrelevanz. Im ersten Schritt werden alle Bedingungen der Länge 1 bestimmt, d.h. alle Terme, die die gegebenen Schwellwerte überschreiten. Danach werden diese in weiteren Schritten zu Bedingungen der Länge 2, 3, 4 usw. zusammengefasst, sofern diese die gegebenen Schwellwerte ebenfalls überschreiten. Dies erfolgt entsprechend des in [Agrawal & Srikant 1994] beschriebenen Algorithmus. Als zusätzliche Modifikation werden nur solche Teilbedingungen miteinander verknüpft, die sich in ihrer Konfliktrelevanz nur innerhalb einer gegebenen Abweichung unterscheiden. Hierdurch wird erreicht, dass sämtliche Terme einer Bedingung in Zusammenhang mit einer nahezu identischen Menge an auftretenden Konflikten stehen.

Bezogen auf das Beispiel in Abbildung 1 besitzt das folgende Fehlermuster eine Konfliktrelevanz von 100% und ein Konfliktpotenzial von 50%:

> WENN $r_1[A_1]$ = *gemessen* $\wedge$ $r_2[A_1]$ = *geschätzt* DANN $CI_2 = 1$

In dem bisher verwendeten Beispiel haben wir anstelle von zwei Relationen nur Duplikatpaare. Dadurch ist keine eindeutige Zuordnung bzw. Reihenfolge gegeben. Unsere Terme können somit keinen Bezug auf r_1 und r_2 nehmen. Die Terme müssen deshalb eine etwas andere Form haben. Eine einfache Möglichkeit besteht darin, sich auf die Existenz eines Wertes in einem Attribut zu beschränken, d.h., anstatt $r_1[A_j] = x$ oder $r_2[A_j] = x$ erhält man $x \in A_j$. Der Ausdruck ist für ein Duplikatpaar dann wahr, wenn in mindestens einem der Tupel das Attribut A_j den Wert x hat. Bezogen auf Tabelle 2 sähe das obige Fehlermuster wie folgt aus:

> WENN *gemessen* $\in$ *Art der Beobachtung* $\wedge$ *geschätzt* $\in$ *Art der Beobachtung* DANN *Konflikt in Größe*

Es besitzt weiterhin die gleichen Werte für Konfliktrelevanz und -potenzial.

5 Auflösen von Konflikten

Nach dem Erkennen von Konflikten beschäftigt sich die Phase der Datenfusion mit dem Auflösen der in den Daten vorhandenen Datenkonflikte. Aus den verschiedenen Repräsentationen eines einzigen Objektes wird eine alleinige konsistente Repräsentation erzeugt. Idealerweise wird Datenfusion im relationalen Umfeld durch deklarative, fusionierende Anfragen an die Daten realisiert, also eine Anfragesprache zur Spezifikation von Konfliktauflösungsregeln verwendet. Im Folgenden analysieren wir zunächst die Möglichkeiten von SQL zur Konfliktauflösung. Da sich diese Möglichkeiten als sehr begrenzt erweisen, stellen wir dann eine speziell zur Datenfusion entwickelte SQL-Erweiterung vor – den FUSE BY-Operator.

Bei der Datenfusion können Unsicherheiten durch einfache Übernahme des von NULL verschiedenen Wertes aufgelöst werden. Konflikte stellen dagegen das interessantere und weitaus schwierigere Problem dar.

5.1 Datenfusion mit SQL

Liegen die Daten unterschiedlicher Quellen in verschiedenen Tabellen vor, können zur Fusion die bekannten Standardoperatoren der relationalen Algebra (Join, Union etc.) verwendet werden. Damit ist aber nur die Behandlung von Unsicherheiten möglich. Widersprüche können diese Operatoren nur in wenigen Spezialfällen auflösen. Auch die bisher in der Literatur vorgeschlagenen Operatoren wie der MatchJoin- [Yan & Özsu 1999] oder der MERGE-Operator [Greco et al. 2001] bieten keine adäquaten Möglichkeiten zum Umgang mit Widersprüchen.

Liegen die Daten wie in unserem Beispiel bereits in einer Tabelle vor, bietet es sich an, die Daten durch Gruppierung und Aggregation zu fusionieren. Zur Gruppierung wird die ObjektID genutzt, wie sie die Duplikaterkennung erzeugt. Leider sind dabei die Möglichkeiten zur Konfliktlösung auf die im SQL-Standard definierten Aggregationsfunktionen (*min*, *max*, *sum*, *count* etc.) begrenzt, was in den meisten Fällen nicht ausreichend ist. Des Weiteren fließen lediglich die Spaltenwerte in die Konfliktlösung mit ein, aber keinerlei zusätzliche hilfreiche Informationen. Projekte wie FRAQL [Sattler et al. 2000] oder AXL [Wang & Zaniolo 2000] beheben einige dieser Unzulänglichkeiten, indem sie benutzerdefinierte Aggregationsfunktionen erlauben (AXL) bzw. einige wenige komplexere Aggregationsfunktionen mit mehr als einem Parameter zur Verfügung stellen. FUSE BY geht darüber hinaus und ermöglicht es, in einfachen Statements komplexe Konfliktlösungen anzugeben.

5.2 Datenfusion als eigenständiger relationaler Operator

Der in unserer Arbeitsgruppe entworfene Fusionsoperator und die dazugehörige SQL-Erweiterung FUSE BY beruht auf dem Prinzip der Gruppierung und Aggregation. Zu einem Objekt der realen Welt gehörende Repräsentationen werden gruppiert und die zwischen den Repräsentationen auftretenden Konflikte werden mit Hilfe von Konfliktlösungsfunktionen aufgelöst. Diese Konfliktlösungsfunktionen sind mächtiger als die in relationalen Datenbanksystemen verwendeten Standard-Aggregationsfunktionen und auch als die in [Sattler et al. 2000] eingeführten Funktionen. SQL-Aggregationsfunktionen bekommen als Parameter lediglich die Spaltenwerte übergeben, wohingegen Konfliktlösungsfunktionen prinzipiell über die gesamten Informationen des Anfragekontextes verfügen, also auch Werte anderer Spalten, Metadaten wie Spaltenname, Tabellenname, Nutzer oder Statistiken der Daten.

Die in der – der Datenfusion unmittelbar vorausgehenden – Phase der Erkennung systematischer Konflikte gesammelten Informationen ermöglichen eine bessere Formulierung dieser Anfrage. Im Folgenden wird kurz beschrieben, wie solch eine FUSE BY-Anfrage gestellt und ausgewertet wird. Weitergehende Informationen findet man in [Bleiholder & Naumann 2005].

5.3 FUSE BY-Anfragen formulieren

Abbildung 2 stellt das Syntaxdiagramm der FUSE BY-Erweiterung vor. FUSE BY ist in Syntax und Semantik an GROUP BY angelehnt. Die folgende Anfrage fusioniert die Tupel der Beispieltabelle und nutzt dabei das bisher gewonnene Wissen um Objekte der realen Welt und systematische Konflikte. Das Ergebnis ist in Tabelle 3 dargestellt, die dazugehörige Anfrage soll die Verwendung von FUSE BY veranschaulichen:

```
SELECT OID,
   RESOLVE(Gattung, Most-
   General),
   RESOLVE(Beobachtung,
   Concat),
   RESOLVE(Spannweite, Avg),
   RESOLVE(Größe, Avg),
   RESOLVE(Gewicht, Max),
   RESOLVE(Gefiedermuster,
   Choose-Depending(Beobach-
   tung, "gemessen")),
   RESOLVE(Gefiederfarbe,
   Choose-Corresponding(Ge-
   fiedermuster)),
   RESOLVE(Beobachter, Choo-
   se-Corresponding(Gefieder-
   muster)),
   RESOLVE(Geschlecht, Vote)
FUSE FROM duplicate_result
   FUSE BY (OID)
   ON ORDER Bez.
```

Diese Anfrage übernimmt die Tabelle, wie sie die Duplikaterkennung erstellt hat, zur Datenfusion und verwendet die Spalte *OID* zur Identifizierung gleicher Objekte. Konflikte in den übrigen Spalten werden durch die angegebenen Konfliktlösungsfunktionen gelöst, auf die später noch einmal genauer eingegangen wird.

5.4 FUSE BY-Anfragen auswerten

Eine FUSE BY-Anfrage wird ausgewertet, indem zuerst die zur Fusion relevanten Tupel aus den Quelltabellen (FUSE FROM-Klausel) zusammengefügt werden, Join- oder sonstige Bedingungen werden ausgewertet. Die FUSE BY-Klausel gibt die Spalte(n) an, nach denen dann gruppiert wird. Pro Gruppe werden exakte Duplikate und subsumierte Tupel entfernt. Ein Tupel t_1 subsumiert ein anderes Tupel t_2, wenn t_2 mehr $\perp$-Werte als t_1 enthält, ansonsten aber in allen Attributwerten mit t_1 übereinstimmt. Mit ON ORDER kann die Reihenfolge der Tupel in den einzelnen Gruppen beeinflusst werden, was bei der Verwendung von ordnungsabhängigen Konfliktlösungsfunktionen eine Rolle spielt. Dann werden die in der SELECT-Klausel angegebenen Konfliktlösungsfunktionen auf die dort angegebenen Spalten angewendet und aus den Tupeln einer Gruppe jeweils ein auszugebendes Tupel erzeugt. Das Endergebnis kann mittels Umbenennungen sowie optionaler HAVING- und ORDER BY-Klauseln beeinflusst werden.

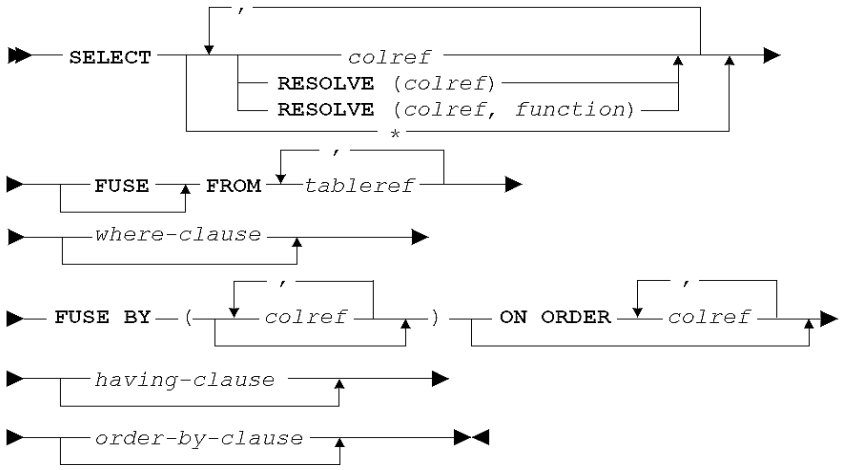

Abb. 2: Syntax von FUSE BY

5.5 Defaultverhalten

FUSE BY-Anfragen erzielen durch ein intuitives Defaultverhalten oftmals schon in einfachen Anfragen gute Ergebnisse. So erzeugt die folgende Anfrage eine Tabelle mit einem Tupel pro Objekt, wobei exakte Duplikate und subsumierte Tupel entfernt werden und die *coalesce*-Funktion als Standard-Konfliktlösungsfunktion verwendet wird, die den ersten Nicht-NULL-Wert der Spaltenwerte einer Gruppe zurückgibt:

```
SELECT *
FUSE FROM Q1
FUSE BY (OID)
```

5.6 Konfliktlösungsfunktionen

Entscheidend für ein qualitativ gutes Fusionsergebnis ist die Verwendung geeigneter Konfliktlösungsfunktionen. In der FUSE BY-Anfrage angegebene Konfliktlösungsfunktionen operieren auf den Spaltenwerten und können weitere zusätzliche Informationen nutzen. Tabelle 4 zeigt mögliche Funktionen. Sie reichen von den Standard-Aggregationsfunktionen *min*, *max*, *avg* bis zu komplexeren Funktionen, die Zusatzinformationen nutzen und oftmals auch domänenabhängig sind. Konflikte in numerischen Daten wie Größe oder Gewicht der im Beispiel beobachteten Eulen können z.B. mittels *avg* aufgelöst werden. Bei der Bestimmung des Geschlechtes kann man die Funktion *vote* nutzen, mit der Intuition, dass der am häufigsten beobachtete bzw. gemessene Wert auch der richtige ist.

Ein Beispiel für eine komplizierte Funktion ist *mostGeneral*. Mit Hilfe einer Taxonomie löst diese Funktion Konflikte in taxonomischen Daten, indem der spe-

OID	Gattung	Art der Beobachtung	Spannweite in cm	Größe	Gewicht	Gefieder Muster	Gefieder Farbe	Beobachter	Geschlecht
1	Bubo Scandiacus	gem./gesch.	151.5	59	1700 g	gescheckt	weiß	Klaus	m
2	Pseudoscops	gem./beob.	119	51.5	2500 g	gestreift	braun	Klaus	$\perp$
3	Bubo Bubo	gesch./beob.	110	45	2300 kg	gefleckt	dunkelbraun, grau	Harry	w
4	Kauz	gem./gesch.	119	55	622 g	gescheckt	hell-braun, grau	Klaus	w

Tab. 3: Ergebnis nach Anwendung des Fusionsoperators

count	Gibt die Anzahl der unterschiedlichen Werte zurück. Keine Konfliktlösung, aber Anzeige eines Konfliktes.
min, *max*	Gibt den maximalen bzw. minimalen Wert zurück. Setzt eine Ordnung (numerische oder lexikographische) auf den Daten voraus.
avg, *median*	Löst Konflikte durch Übernahme des Mittelwertes oder Medians bei numerischen Daten.
random	Wählt einen zufälligen Wert aus den vorhandenen aus. Mögliche, aber nicht unbedingt wünschenswerte Konfliktlösung, da indeterministisch.
first, *last*, *coalesce*	Übernimmt den ersten, letzten bzw. ersten von NULL verschiedenen Wert.
vote	Löst Konflikte durch Mehrheitsentscheid, übernimmt den am häufigsten aufgetretenen Wert. Übernahme des ersten aufgetretenen bei Gleichstand.
group, *concat*	Gibt alle vorhandenen Werte als Wertemenge zurück (Group) bzw. fügt diese aneinander und gibt sie als einen Wert zurück (Concat). Damit wird dem Nutzer die Konfliktlösung überlassen.
chooseCorresponding (*Spalte*)	Übernimmt den Attributwert des Tupels, dessen Attributwert der angegebenen Spalte übernommen wurde.
chooseDepending (*Spalte*, *Wert*)	Wählt den Attributwert des Tupels aus, dessen angegebene Spalte den gegebenen Wert enthält.
mostGeneral (*Taxonomie*), *mostSpecific* (*Taxonomie*)	Nutzt eine Taxonomie, um den speziellsten Oberbegriff bzw. den speziellsten der in Konflikt stehenden Werte zu übernehmen.
choose(*Quelle*)	Übernimmt den Wert der angegebenen Quelle.

Tab. 4: Konfliktlösungsfunktionen

ziellste gemeinsame Oberbegriff genommen wird. Denkbar ist auch die Übernahme des taxonomisch speziellsten der in Konflikt stehenden Werte. Im Beispiel ist dies z.B. die Information über die Gattung der Eulen.

Informationen über (systematische) Konflikte und Fehler können z.B. zur Quellenauswahl genutzt werden. Wird beobachtet, dass eine Arbeitsgruppe nicht in der Lage ist, das Geschlecht der Eulen zu bestimmen, kann man immer die Werte der anderen übernehmen. In diesem Fall würde man *coalesce* verwenden. Liefern beide Arbeitsgruppen Werte, kann man seine Präferenz mittels *choose* ausdrücken. Dies kann seine Anwendung z.B. bei der Verwendung unterschiedlicher Messmethoden der Arbeitsgruppen finden, indem das Ergebnis der jeweils präziseren Gruppe übernommen wird.

Das Stellen einer Fusionsanfrage ist durch die Erweiterung von SQL um FUSE BY relativ simpel und ermöglicht dem Experten die einfache Fusionierung von Daten. Die bei der Konflikterkennung gewonnenen Erkenntnisse erleichtern weiterhin eine gute Fusionierung. Prinzipiell ist auch die automatische Erstellung einer geeigneten FUSE BY-Anfrage aufgrund der gefundenen Konflikte denkbar. Wegen des meist umfangreicheren Wissens eines menschlichen Experten ist dies aber nicht unbedingt empfehlenswert.

5.7 Strategien zur Konfliktlösung

Bei der Auflösung von Konflikten sind im Allgemeinen unterschiedliche Strategien möglich. Mit Hilfe der FUSE BY-Erweiterung ist eine einfache Umsetzung einiger gängiger Konfliktlösungsstrategien möglich:

- Auswahl aufgrund von beschreibenden Daten, also z.B. die Auswahl aufgrund der Quelle, aufgrund der Aktualität, aufgrund der Messmethode oder aufgrund der Kosten.
- Auswahl aufgrund von Charakteristika der Daten, also z.B. Auswahl von Mittelwerten, um den Einfluss von Messfehlern zu vermindern, oder auch die Auswahl des häufigsten Wertes.
- Keine Auswahl eines spezifischen Wertes, sondern Aneinanderreihung aller Werte oder Rückgabe einer Wertemenge (Gruppierung aller Werte). Damit wird die endgültige Auflösung des Konfliktes dem menschlichen Experten überlassen.

Diese Strategien können auf verschiedene Art und Weise durch eine FUSE BY-Anfrage umgesetzt werden. So kann z.B. ein Zeitstempel als zusätzliche Spalte mitgeführt, nach dieser Spalte absteigend sortiert und mit *first* der jeweils aktuellste Attributwert übernommen werden (Auswahl aufgrund der Aktualität). Auch die Lösung von Konflikten in Teilen der Daten oder eine unterschiedliche Konfliktlösung für verschiedene Teile des Datenbestandes ist durch die geschickte Kombination des Fusions- mit anderen relationalen Operatoren (z.B. Selektion) möglich.

6 Zusammenfassung

Die Integration naturwissenschaftlicher Daten kann nicht vollkommen automatisiert werden, jedoch können einzelne Schritte des Integrationsprozesses durch verschiedene Werkzeuge unterstützt werden. Wir haben drei entsprechende Verfahren vorgestellt. Die Erkennung von unterschiedlichen Repräsentationen eines einzigen *real-world*-Objektes wird durch Verfahren der Duplikaterkennung unterstützt. Die FUSEBY-Erweiterung erlaubt die deklarative Verschmelzung dieser unterschiedlichen Repräsentationen, wobei umfangreiche Konfliktlösungsmöglichkeiten angegeben werden können. Die Auswahl einer geeigneten Konfliktlösung wird durch das automatische Finden systematischer Konflikte wesentlich unterstützt.

Die vorgestellten Verfahren befinden sich zurzeit in unterschiedlichen Stadien der Realisierung. Sowohl die Algorithmen zur Duplikaterkennung als auch die zur Mustersuche sind implementiert. Der FUSE BY-Operator wird zurzeit auf der Datenbankbibliothek XXL [Bercken et al. 2001] realisiert.

Literatur

[Agrawal & Srikant 1994] *Agrawal, R.; Srikant, R.:* Fast Algorithms for Mining Association Rules. In: Proc. 20th International Conference on Very Large Database Systems, Santiago de Chile, Chile. 1994.

[Ananthakrishna et al. 2002] *Ananthakrishna, R.; Chaudhuri, S.; Ganti, V.:* Eliminating fuzzy duplicates in data warehouses. In: Proc. 28th International Conference on Very Large Databases, Hong Kong, China, 2002.

[Bercken et al. 2001] *den Bercken, J. V.; Blohsfeld, B.; Dittrich, J.-P.; Krämer, J.; Schäfer, T.; Schneider, M.; Seeger, B.:* XXL – a library approach to supporting efficient implementations of advanced database queries. In: Proc. 27th International Conference on Very Large Databases, Roma, Italy, 2001.

[Bleiholder & Naumann 2005] *Bleiholder, J.; Naumann, F.:* Declarative data fusion – syntax, semantics and implementation. In: Proc. 9th East-European Conference on Advances in Databases and Information Systems, Tallinn, Estonia, 2005, to appear.

[Bohannon et al. 2005] *Bohannon, P.; Fan, W.; Flaster, M.; Rastogi, R.:* A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification. In: Proc. International Conference on Management of Data, Baltimore, MD, 2005.

[Breiman et al. 1984] *Breiman, L.; Friedman, J. H.; Olshen, R. A.; Stone, C. J.:* Classification and Regression Trees. Wadsworth International Group, 1984.

[Brenner 1999] *Brenner, S. E.:* Errors in genome annotation. Trends in Genetics, 15, 4, 132-133, 1999.

[Dennis & Gallagher 2002] *Dennis, C.; Gallagher, R. (Eds.):* The Human Genome. Palgrave Macmillan, 2002.

[Embury et al. 2001] *Embury, S. M.; Brand, S. M.; Robinson, J. S.; Sutherland, I.; Bisby, F. A.; Gray, W. A.; Jones, A. C.; White, R. J.:* Adapting integrity enforcement techniques for data reconciliation Information Systems, Vol. 26, 2001, 657-689.

[Galperin 2005] *Galperin, M. Y.:* The Molecular Biology Database Collection: 2005 update. Nucleic Acids Res. 33 (Database issue), D5-24, 2005.

[Greco et al. 2001] *Greco, S.; Pontieri, L.; Zumpano, E.:* Integrating and managing conflicting data. In: Revised Papers from the 4th Int. Andrei Ershov Memorial Conf. on Perspectives of System Informatics, pp. 349-362, 2001.

[Hardison 2003] *Hardison, R. C.:* Comparative genomics. PLoS Biol 1(2), E58, 2003.

[Hernández & Stolfo 1995] *Hernández, M. A.; Stolfo, S. J.:* The merge/purge problem for large databases. In: Proc. ACM International Conference on Management of Data, San Jose, CA, 1995.

[Liu et al. 1998] *Liu, B.; Hsu, W.; Ma, Y.:* Integrating Classification and Association Rule Mining. In: Proc. 4th ACM International Conference on Knowledge Discovery and Data Mining New York, NY, 1998.

[LV Eulenschutz 2004] *Landesverband für Eulenschutz in Schleswig-Holstein e.V.:* Eulen Welt 2004; *http://www.eulen.de/.*

[Makarov 2002] *Makarov, V.:* Computer programs for eukaryotic gene prediction. Briefings in Bioinformatics, 3, 2, 2002.

[Monge & Elkan 1997] *Monge, A. E.; Elkan, C. P.:* An effcient domain-independent algorithm for detecting approximately duplicate database records. In: Proc. SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, Tuscon, AZ, 1997.

[Müller 2003] *Müller, H.:* Semantic Data Cleansing in Genome Databases. In: Proc. of the VLDB 2003 PhD Workshop, Berlin, Germany, September 12-13, 2003.

[Müller et al. 2003] *Müller, H.; Naumann, F.; Freytag, J.-C.:* Data Quality in Genome Databases. In: Proc. Conference on Information Quality, Boston, Mass., 2003.

[Müller et al. 2004] *Müller, H.; Leser, U.; Freytag, J.-C.:* Mining for Patterns in Contradictory Data. In: Proc. SIGMOD International Workshop on Information Quality for Information Systems, Paris, France, 2004.

[Sattler et al. 2000] *Sattler, K.-U.; Conrad, S.; Saake, G.:* Adding Conflict Resolution Features to a Query Language for Database Federations. In: Proc. 3rd Int. Workshop on Engineering Federated Information Systems, 2000.

[Wang & Zaniolo 2000] *Wang, H.; Zaniolo, C.:* Using SQL to build new aggregates and extenders for object- relational systems. In: Proc. 26th International Conference on Very Large Databases, Cairo, Egypt, 2000.

[Webb et al. 2003] *Webb, G. I.; Butler, S.; Newlands, D.:* On detecting differences between groups. In: Proc. 9th ACM International Conference on Knowledge Discovery and Data Mining, Washington, DC, 2003.

[Weis & Naumann 2004] *Weis, M.; Naumann, F.:* Detecting Duplicate Objects in XML Documents. In: Proc. SIGMOD International Workshop on Information Quality for Information Systems, Paris, France, 2004.

[Weis & Naumann 2005] *Weis, M.; Naumann, F.:* DogmatiX Tracks down Duplicates in XML. In: Proc. International Conference on Management of Data, Baltimore, MD, 2005.

[Yan & Özsu 1999] *Yan, L. L.; Özsu, M.:* Conflict tolerant queries in AURORA. In: Proc. International Conference on Cooperative Information Systems, Edinburgh, Scotland, 1999.

Heiko Müller studierte Informatik an der Technischen Universität Berlin. Seit 2000 ist er Mitglied des Berlin-Brandenburger Graduiertenkollegs »Verteilte Informationssysteme« und arbeitet als wissenschaftlicher Mitarbeiter im Bereich Datenbanken und Informationssysteme der Humboldt-Universität zu Berlin. Im Rahmen seiner Dissertation befasst er sich mit der Bereinigung semantischer Fehler in Genomdatenbanken.

Melanie Weis arbeitet als wissenschaftliche Mitarbeiterin in der Arbeitsgruppe Informationsintegration der Humboldt-Universität zu Berlin und beschäftigt sich im Rahmen ihrer Promotion mit Objektidentifikation in XML-Daten.

Jens Bleiholder arbeitet als wissenschaftlicher Mitarbeiter in der Arbeitsgruppe Informationsintegration der Humboldt-Universität zu Berlin. Er beschäftigt sich im Rahmen seiner Promotion mit der Auflösung von Konflikten bei der Datenfusion.

Ulf Leser arbeitete nach seinem Studium zunächst an integrierten Datenbanken im Rahmen des Human-Genome-Projekts.

Seine Promotion beschäftigte sich mit Anfrageoptimierung und Informationsintegration; danach leitete er industrielle Softwareentwicklungsprojekte im Bereich Data Warehousing und Wissensmanagement. Seit 2002 ist er Professor für Wissensmanagement in der Bioinformatik an der Humboldt-Universität zu Berlin.

Dipl.-Inform. Heiko Müller
Dipl.-Ing. Melanie Weis
Dipl.-Inform. Jens Bleiholder
Humboldt-Universität zu Berlin
Institut für Informatik
Unter den Linden 6
10099 Berlin
hmueller@dbis.informatik.hu-berlin.de
mweis@informatik.hu-berlin.de
jens.bleiholder@staff.hu-berlin.de
http://www.informatik.hu-berlin.de

Prof. Dr. Ulf Leser
Humboldt-Universität zu Berlin
Wissensmanagement in der Bioinformatik
Rudower Chaussee 25
12489 Berlin
leser@informatik.hu-berlin.de
http://www.informatik.hu-berlin.de/wbi/