Seminararbeit im Fachbereich Datenbanken

Seminar Schema Evolution

Thema: Schema-Matching

Autor: Elke Klippstein

Betreuer: Hr. Aumüller

Inhaltsverzeichnis

1 Einleitung

Was ist Schema-Matching? Dazu muss man etwas weiter ausholen und erklären was Schema Mapping ist. Unter Schema Mapping versteht man, die konkrete Abbildung eines Schemas auf ein anderes Schema [WI05]. Schema Matching ist das automatische Erkennen eines solchen Mappings.
In vielen Datenbank Anwendungsbereichen [RH02], z. B. Datenintegration oder E-Business stellt Schema Matching ein generelles Problem dar. In laufenden Implementationen wird Schema-Matching typischerweise per Hand ausgeführt, dies führt zu großen Einschränkungen. Auf der anderen Seite haben viele frühere Forschungstechniken vorgeschlagen, die eine teilweise Automatisierung von Verschmelzungsoperationen für spezifische Anwendungsbereiche erreichen. In den folgenden Kapiteln werden einige Vorgehensweisen genauer untersucht und beschrieben. Im einzelnem bedeutet das, dass man zwischen verschieden Schemata unterscheidet. Es gibt die Unterscheidung zwischen Schema- und Instanz-Level, Element- und Struktur-Level und zum Schluss die sprachbasierten - und beschränkungsbasierten Matcher.
Eine Grundoperation zum Manipulieren von Schemainformationen ist ein Match, das heißt es nimmt zwei Schemata als Eingabe und erstellt eine Verschmelzung zwischen den Elementen dieser zwei Schemata, bei der die Semantik einander entspricht. Matching spielt eine zentrale Rolle in vielen Anwendungen, z. B. weborientierte Datenintegration, E-Commerce, Schemaintegration, Schema Evolution und Migration, Data-Warehouse und componentenbasierte Entwicklungen. Zur Zeit wird Schema Matching manuell ausgeführt, jedoch mit Unterstützung eines graphischen User Interfaces. Offensichtlich ist das manuell ausgeführte Schema Matching mühsam, zeitintensiv, fehleranfällig und deshalb ein teurer Prozess. Dies ist ein sich ausweitendendes Problem in der schnell ansteigenden Anzahl an zu integrierenden Webdaten und E-Bussines. Ein weiteres Problem sind die komplexen Datenbankanwendungen und ihre große Anzahl an Schematas die verarbeitet werden sollen. Der Arbeitsaufwand steigt linear mit der Anzahl der zu verarbeitenden Vergleiche an. Eine schnellere und weniger arbeitsintensive Entwicklung wird benötigt, also eine automatische Unterstützung für das Schema Matching. Dies führt zu einer Entwicklung von anwendungsspezifischen Werkzeugen, welche einen automatischen Schemamatch beinhalten. Solch eine Implementation kann also auch eine Schlüsselkomponente in einem umfassenden Modell Management Ansatz haben. In den folgenden Kapiteln werden einige Arten von Schema Matching und Anwendungen beleuchtet.

2 Anwendungsgebiete

2.1 Schema Integration

Schema Integration ist ein Problem, dass seit den frühen 80er Jahren beim Schema Matching auftaucht. Auf dem Gebiet der künstlichen Intelligenz besteht das Problem der Integration unabhängig entwickelter Ontologien in eine Einzelontologie. Durch die unabhängige Entwicklung von Schemata sind diese meist von unterschiedlicher Struktur und Terminologie. Dies tritt offenbar dann auf, wenn die Schemas aus verschiedenen Gebieten kommen. Jedoch tritt es auch dann auf, wenn man versucht den gleichen Weltausschnitt zu modellieren, da sie von verschiedenen Menschen in verschieden Kontexten entwickelt wurden. Ein erster Schritt ist die Identifizierung und Charakterisierung dieser Zwischenschema Beziehungen. Sind diese Elemente identifiziert, können Matching Elemente unter zusammenhängenden, integriertem Schema oder Sichten vereinheitlicht werden. Während dieser Integration (wird manchmal auch als separater Schritt, Programm oder Abfrage gemacht), werden zugelassene Übersetzungen von Daten vom Orginalschema in die integrierte Repräsentation erstellt. Ein Problem entsteht auch beim Integrieren unabhängig entwickelter Schema mit gegebenem konzeptionalen Schema. Dies verlangt das man die Struktur und die Terminologie zweier Schematas miteinander vereinbart, welches Schema Matching beinhaltet.

2.2 Data-Warehouse

Eine Variation des Schemaintegrationsproblems, welches in den 90er Jahren bekannt wurde, ist die Integration von Datenquellen in ein Data-Warehouse. Der Extraktionsprozess erfordert eine Umwandlung von Daten aus dem Ursprungsformat in das Warenhausformat. Die Match-Operation ist hilfreich zur Gestaltung der Umformung. Eine Vorgehensweise zur Erschaffung einer angemessenen Transformation von einer gegebenen Datenquelle beginnt beim Finden eines Elements, das sowohl in der Quelle als auch im Warehouse vorhanden ist. Diese Vorgehensweise nennt man auch Matchoperation. Nach dem ein Mapping erstellt wurde, muss der Data-Warehousedesigner die genaue Semantik jedes Quellelementes untersuchen und eine Umformung erstellen, die die Semantik mit der Zielsemantik in Einklang bringt. Eine andere Vorgehensweise zur Integration einer neuen Datenquelle S', ist die Wiederverwendung einer existierenden Quelle-zu-Warehouse Umformung S $\rightarrow$ W. Als erstes werden gemeinsame Elemente von S' und S heraus gesucht und dann S $\rightarrow$ W wiederverwendet.

2.3 E-Commerce

E-Commerce gibt eine neue Motivation für das Schema Matching, das Übersetzen von Nachrichten. Handelspartner tauschen häufig Informationen über Transaktionen aus. Gewöhnlich hat jeder Handelspartner sein eigenes Nachrichtenformat. Die Nachrichtenformate sind in ihrer Syntax verschieden, z.B. EDI, XML oder kundenspezifische Struktur. Um es den Systemen möglich zu machen Nachrichten auszutauschen, müssen Anwendungsentwickler die Informationen zwischen den benötigten Formaten von unterschiedlichen Handelspartner konvertieren. Ein Teilproblem der Nachrichtenübersetzung ist die Übersetzung zwischen verschiedenen Nachrichtenschemata. Nachrichtenschematas haben verschiedene Namen, sowie verschiedene Datentypen und unterschiedlich zulässige Werte. Die Übersetzung zwischen verschiedenen Nachrichtenschemas ist teilweise ein Schema

Matching Problem. Heutzutage brauchen Anwendungsentwickler spezifische Anleitungen, wie zugehörige Nachrichten formatiert sind. Eine Matchoperation würde das Ausmaß von manueller Arbeit zum Generieren eines Entwurfsmapping zwischen den zwei Nachrichten Schemas reduzieren, welches ein Anwendungsdesigner nachträglich für gültig erklären und verändern kann wie er es braucht. Schema Match ist hilfreich für Anwendungen die für das Semantik-Web berücksichtigt werden.

2.4 Semantische Anfrageprozesse

Schemaintegration, Datawarehouse und E-Commerce sind alle ähnlich in der betroffenen Designanalyse von Schematas, diese produzieren Mappings und möglicherweise ein integriertes Schema. Ein etwas anderes Szenario ist der semantische Anfrageprozess - ein Laufzeit-Szenario, wo ein Benutzer die Ausgabe einer Anfrage spezifiziert (z.B. die SELECT Klausel in SQL), und das System deutet an wie diese Ausgaben erstellt werden (z.B. durch Bestimmung der FROM und WHERE Klausel in SQL). Diese Nutzerspezifikation sind möglicherweise keine Elementnamen im Datenbankschema. Deshalb muss das System, in der ersten Anfrageverarbeitung, die nutzerspezifizierten Konzepte in der Anfrageausgabe auf Schemaelemente abbilden. Dies ist eine natürliche Anwendung von einer Matchoperation. Nach dem Mapping der Anfrageausgabe zu den Schemaelementen des Systems muss eine Qualifikation (z.B. eine WHERE Klausel) abgeleitet werden, die die Semantik des Mapping vorgibt. Techniken für das Erzielen dieser Qualifikationen wurden in den letzten 20 Jahren erschlossen [KKFG84].

3 Der Matchoperator

Um den Matchoperator, match, zu bestimmen, muss man eine Darstellung für seine Eingangsschema und das Ausgabemapping bestimmen. Einige Vorgehensweisen sind abhängig von der Art der Schemainformationen, wie man es nutzt und wie man es interpretiert. Sie hängen stark von der internen Repräsentation der Informationen ab, es ist aber schwierig die gewünschten Informationen so zu repräsentieren, das sie aussagekräftig genug sind. Deshalb definiert man ein Schema als ein Menge von Elementen, die durch eine Struktur verbunden sind. In der Praxis muss man eine Repräsentationsart wählen, z. B. ER-Modell, objektorientiertes Modell, XML oder Graphen. In jedem Fall gibt es eine natürliche Verbindung zwischen den gebildeten Blöcken der Repräsentation und der Notation von Elementen und deren Strukturen, also Objekte und Beziehungen im ER-Modell, Objekte und Beziehungen im OO-Modell, Unterelemente und IDREFs in XML, und Knoten und Kanten in Graphen. Man definiert Mapping als eine Menge von Mappingelementen, wobei jedes anzeigt das bestimmte Elemente von Schema S1 auf bestimmte Elemente von Schema S2 abgebildet werden. Jedes Mappingelement kann einen Mappingausdruck haben, welcher spezifiziert wie die Elemente von S1 und S2 verbunden sind. Der Mappingausdruck kann gerichtet sein, z. B. eine Referenz von einem Element von S1 zu einem Element aus S2 oder er ist ungerichtet, wenn eine Beziehung zwischen einer Kombination von Elementen von S1 und S2 besteht. Das Mapping benutzt einfache Beziehungen über skalare (z.b. $=, \leq$), satzorientierte Beziehungen (z.B. überlappen, enthalten [LNE89], oder andere Bedingungen die in der Ausdruckssprache benutzt werden.

S1 elements	S2 elements
Cust	Customer
C#	CustID
CName	Company
FirstName	Contact
LastName	Phone

Fig. 1: Beispiel eines Inputschematas

Figur 1 zeigt zwei Schemata S1 und S2, welche Kundeninformationen darstellen. Ein Mapping zwischen S1 und S2 sollte ein Mappingelemtent Cust.C♯ enthalten, welches mit Customer.CustID mit dem Mappingausdruck Cust.C♯ = Customer.CustID in Verbindung steht. Ein Mappingelement mit dem Ausdruck „Concatenate (Cust.FirstName, Cust.LastName) = Customer.Contact“ beschreibt eine Verschmelzung zwischen zwei S1 Elementen und einem S2 Element. Eine Matchoperation ist eine Funktion, die zwei Schema S1 und S2 als Eingabe benutzen und diese zwei Schemata verschmelzen zu einer Ausgabe, auch Matchergebnis genannt. Jedes Mappingelement vom Matchergebnis spezifiziert diese bestimmten Elemente von S1 entsprechend logisch, dies nennt man match, bestimmte Elemente von S2 , bei denen die Semantik ähnlich ist werden durch die Mappingparameter ausgedrückt. Dieses Kriterium benutzt Matchelemtente von S1 und S2, welche auf Heuristiken basieren. Diese sind nicht einfach mathematisch zu erfassen, aber es kann uns zu einer Anleitung für die Implementierung von Matches führen. Ähnlich wie bei früheren Arbeiten konzentriert man sich auf Matchalgorithmen, die ein Mapping zurückgeben, welches keine Mappingausdrücken beinhaltet. Daher wird ein Mapping oft als eine Ähnlichkeitsbeziehung abgebildet, $\cong$, über den Potenzmengen von S1 und S2, bei denen jedes Paar in $\cong$ ein Mappingelement vom Mapping darstellt. Beispielsweise das Ergebnis vom Match in Figur 1 könnte „Cust.C♯ $\cong$ Customer.CustID“, „CustCName $\cong$ Customer.Company“

sein. Eine Spezifikation von diesen Ergebnissen vom Match würde die Mappingausdrücke von jedem Element beinhalten, wie „Cust.C♯ = Customer.CustID“ und „Cust.CName = Customer.Company“. Einige Matchentwicklungen sind gleichzusetzen mit den Joinprozessen in relationalen Datenbanken, d.h. beide Operationen, Match- und Joinoperation, sind binäre Operationen, die Paare von entsprechenden Elementen von ihren Eingabeoperanten bestimmen. Natürlich gibt es auch einige Unterschiede zwischen diesen beiden Operationen. Matchoperatoren sind Metadaten (Schemaelemente) und Joinoperatoren sind Daten (Reihen von Tabellen). Wobei Matchoperatoren komplexer sind, als Joinoperatoren. Jedes Element im Joinergebnis vereint nur ein Element von ersten mit einem zweitem Matchingelement von der Eingabe, jedoch kann ein Element im Matchergebnis mit mehreren Elementen von beiden Eingaben verknüpft sein. Joinsemantik ist durch einen einzelnen Vergleichsausdruck (z.b. eine Vergleichsbedingung im natürlichem Join) gegeben, dieser muss für alle Matching Eingabeelemente gelten. Im Gegensatz dazu kann jedes Element in einem Matchergebnis einen anderen Mappingausdruck haben. Die Semantik vom Match ist weniger eingeschränkt als beim Join, aber sie ist schwieriger zu erfassen. Die Ähnlichkeit von Match und Join erstreckt sich bis zu Outermatchoperationen, diese sind ein nützliches Gegenstück vom Match, genauso wie Outerjoin ein Gegenstück von Join ist. Ein rechter (oder linker) Outermatch gewährleistet, das jedes Element von S2 (oder S1) sich auf das Mapping bezieht. Ein voller Outermatch garantiert, das jedes Element von S1 und S2 in das Mapping einbezogen wird. Um sicherzustellen, dass jedes Element von S1 durch das Mapping, zurückgegeben durch den Match, referenziert wurde, kann das Mapping durch andere Mappings, die auf S verweisen, zusammengesetzt werden. Obwohl die Verwendung von Outermatch einige Feinheiten betreffen, ist die Implementation eine geradlinige Erweiterung vom Match, die durch einen Algorithmus für die Matchoperation gegeben ist. Outermatch kann einfacher errechnet werden durch hinzufügen von Elemente zum Matchergebnis, diese beziehen sich auf die anderen nicht referenzierten Elemente von S1 oder S2. Deshalb wird der Outermatch in diesem Dokument nicht weiter betrachtet.

4 Architektur von generischen Matches

Um Matchansätze zu betrachten, ist es hilfreich, in Gedanken eine Architektur zur Implementation zu haben. Deshalb beschreibt man eine high-level Architektur für eine generische, kundenspezifische Implementation von Match. Diese Architektur wird in Figur 2 veranschaulicht. Die verwendeten Clients sind schemabezogene Anwendungen und Werkzeuge von verschiedenen Bereichen, z.B. e-Bussines, Portale und DataWarehousing. Jeder dieser Clients benutzt eine generische Matchimplementation zur automatischen Bestimmung von Matches zwischen zwei Eingabeschemas. XML-Schema Editor, portale Entwicklungsunterstützung, Datenbankmodellierungswerkzeuge können Bibliotheken nutzen, um existierende Schematas auszuwählen. Dies ist links in der Figur 2 dargestellt. Die

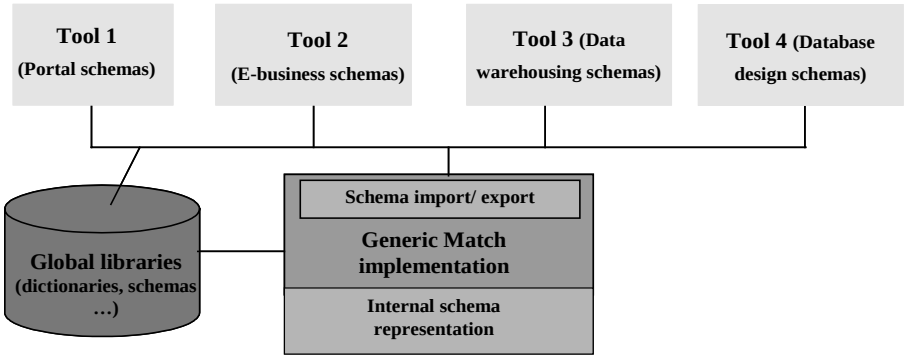

Fig. 2: Highlevel Architektur von generischen Matches

Realisierung von Match kann Bibliotheken und andere Zusatzinformationen nutzen, z.B. Wörterbücher und Thesauri, um Matches zu finden. Es wird angenommen, das die generische Implementation von Match auf das Schema abgebildet wird. Um eine einheitliche Darstellungsweise zu erreichen, muss man die Komplexität von Matches reduzieren, diese befassen sich nicht mit der großen Anzahl von unterschiedlichen (heterogenen) Schemadarstellungen. Werkzeuge die fest in das Framework integriert sind, können direkt in der internen Repräsentation arbeiten. Andere Werkzeuge brauchen Import/Export Programme um zwischen ihrer Schemarepräsentation (z.B. XML, SQL oder UML) und der internen Repräsentation zu übersetzen. Dabei übersetzt ein Importprogramm das Eingangsschema in eine interne Repräsentation und das Exportprogramm ein Mapping, welche durch eine generische Realisierung erstellt wurden, in eine Darstellung, die von jedem Werkzeug benutzt wird. Dies erlaubt der generischen Implementation von Matches lediglich im internen Schema zu arbeiten. Gewöhnlich ist es nicht möglich alle Matches zwischen zwei Schematas vollautomatisch zu bestimmen, hauptsächlich weil die meisten Schematas einige Semantiken haben, die die Anpassungskriterien beeinflussen. Dies ist in der Literatur nicht genau beschrieben oder wird gerade dokumentiert. Deshalb sollte die Anwendung von Match nur solche Matchkandidaten bestimmen, die der Anwender akzeptieren, ablehnen oder ändern kann. Weiterhin sollte der Benutzer fähig sein Übereinstimmungen für Elemente anzugeben, da das System nicht immer zufrieden stellende Matchkandidaten findet.

5 Klassifikation von Schemamatching Methoden

In diesem Abschnitt wird eine Klassifikation von Schemamatchingansätzen angegeben. Dies wird in Figur 3 dargestellt gemeinsam mit einigen Beispielansätzen.

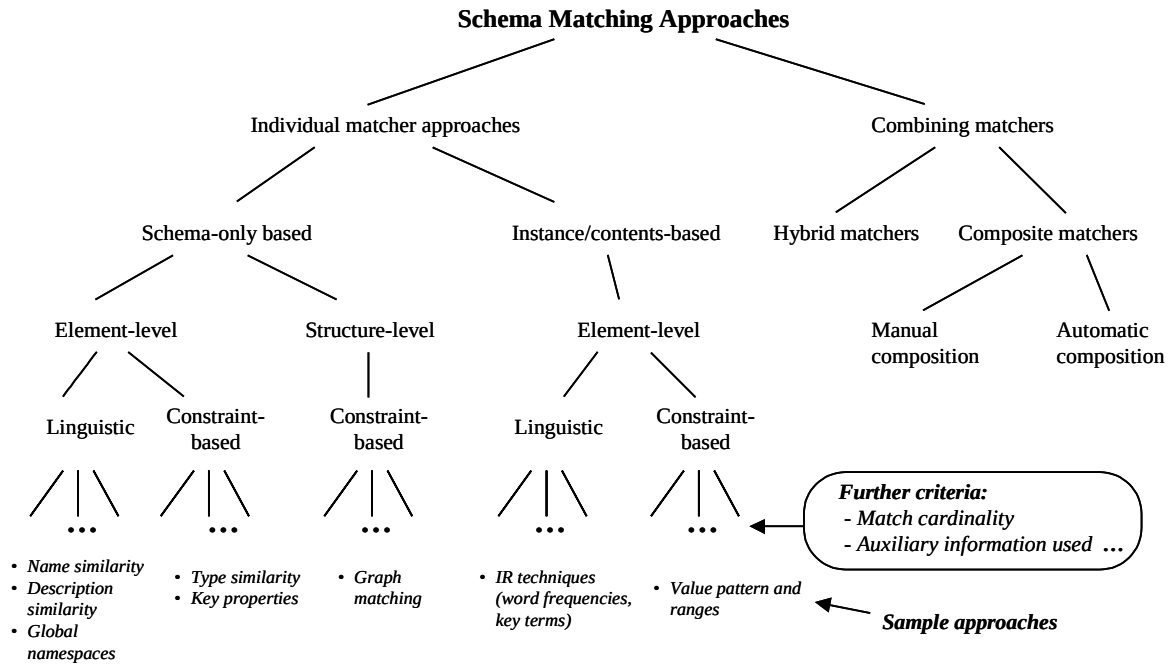

Fig. 3: Klassifikation von Schemamatchingansätzen

Eine Matchanwendung kann mehrere Matchalgorithmen oder Matcher benutzen. Dies erlaubt uns die Tiefe der Verknüpfung von den Anwendungsgebieten und von Schematypen festzulegen. Wenn man mehrere Matcher benutzt, dann ergeben sich zwei Probleme. Erstens, die Realisierung von einzelnen Matchern. Dabei wird ein Mapping basierend auf ein einzelnes Matchingkriterium ausgerechnet. Zweitens, die Kombination von einzelnen Matchern. Entweder durch nutzen einzelner Matchingkriterien (z.B. Namens- und Typgleichheit) innerhalb eines eingebundenen Hybridmatchers oder durch die Kombination von mehreren Matchergebnissen, die von verschiedenen Matchalgorithmen innerhalb eines zusammengesetzten Matchers erzeugt werden. Für Einzelmatcher gelten die folgenden Klassifikationskriterien:

- Instanz vs. Schema: Matchingmethoden können Instanzdaten (z.B. Dateninhalte) oder nur Schemalevel-Informationen berücksichtigen.
- Element vs. Strukturmatching: der Match kann für einzelne Schemaelemente ausgeführt werden, z.B. Attribute oder für Elementkombinationen wie eine komplizierte Schemastruktur.
- Sprache vs. Beschränkung: ein Matcher kann eine sprachbasierte Methode (z.B. bezogen auf Namen und Textbeschreibungen von Schemaelementen) oder eine beschränkungsbasierte Methode (z.B. bezogen auf Schlüssel und Beziehungen) benutzen.

- Matchingkardinalität: das Matchergebnis kann sich auf ein oder mehrere Elemente von einem Schema bzw. einem anderen Schema beziehen, daraus entstehen 4 Fälle 1:1, 1:n, n:1 und n:m. Jedes Mappingelement kann verknüpft sein mit einem oder mehreren Elementen von den zwei Schematas.

- Zusatzinformation: die meisten Matcher verlassen sich nicht nur auf die Eingabeschemata S1 und S2, sondern auch auf Zusatzinformation, wie Wörterbücher, allgemeine Schematas, frühere Entscheidungen von Matchings und Benutzereingaben.

Die Klassifikation kann nicht zwischen verschiedenen Schematypen (Relational, XML, objektorientiert etc.) und ihrer internen Darstellung unterscheiden, weil die Algorithmen meistens abhängig von der Art der Informationen sind, die sie nutzen und nicht von ihrer Darstellung.

6 Matcharten

6.1 Schemalevel Matcher

Schemalevel Matcher berücksichtigen nur Schemainformationen, nicht Instanzdaten. Die verfügbare Information beinhaltet die gebräuchlichen Eigenschaften von Schemaelementen, wie Name, Beschreibung, Datentypen, Beziehungstypen (Part-of, is-a, usw.), Beschränkungen und die Schemastruktur. Üblicherweise findet ein Matcher mehrere Matchkandidaten, für jeden Kandidat berechnet man den Ähnlichkeitsgrad. Dieser ist ein normalisierter numerischer Wert zwischen Null und Eins. Diese werden der Größe nach geordnet, um den besten Matchkandidaten herauszufinden (wie in [BCV99], [DDL00], [CDD01]).

6.1.1 Granularität von Match (elementlevel vs. strukturlevel)

Man unterscheidet zwei Hauptvarianten für die Granularität von Match, Elementlevel und Strukturlevel Matching. Für jedes Element des ersten Schemas, bestimmt der Element-Level-Abgleich die Matchingelemente aus dem zweitem Schema. Im einfachsten Fall sind nur Elemente aus dem besten Level der Granularität berücksichtigt, dies nennt man Atomlevel, wie die Attribute in einem XML Schema oder Spalten in einem relationalen Schema. Ein Teilschema ist in Figur 4 dargestellt, „Address.ZIP ≅ CustomerAddress.PostalCode“ ist ein Beispiel für solch ein Atomlevel Match.

S1 elements	S2 elements	
Address	CustomerAddress	full structural match of
Street	Street	Address and CustomerAddress
City	City	
State	USState	
ZIP	PostalCode	
AccountOwner	Customer	partial structural match of AccountOwner and
Name	Cname	Customer
Address	CAddress	
Birthdate	CPhone	
TaxExempt		

Fig. 4: voller vs. partieller struktureller Match

Andererseits verweisen Strukturlevel Matching auf Matchingkombinationen von Elementen, die sich in einer gemeinsamen Struktur befinden. Dadurch ist eine Fallgruppierung möglich, abhängig davon wie komplett und genau ein Match in seiner Struktur erfordert wird. Im idealen Fall sind alle Komponenten der Strukturen in den zwei Schematas vollständig verknüpft. Eine Alternative ist, das nur einige von den Komponenten von dem Match erfordert werden (z.B. partieller Strukturmatch). Beispiele zu diesen zwei Fällen sind in Figur 4 gezeigt. Manchmal braucht man den partiellen Match, damit Subschematas von verschiedenen Bereichen gegenübergestellt werden können. Zum Beispiel, in der zweiten Reihe der Figur 4, AccountOwner kann von einer Finanzdatenbank kommen, obwohl Kunde von einer Verkaufsdatenbank kommt. Für komplexere Fälle kann die Effektivität von Strukturmatching erhöht werden, indem bekannte Äquivalenzmuster berücksichtigt werden. Diese Muster können in einer Bibliothek enthalten sein. Ein einfaches Muster ist in Figur 5 dargestellt. Dabei werden zwei Strukturen in einer is-a Hierarchie zu einer einfachen Struktur verbunden. Die Subklasse vom ersten Schema ist auf ein Booleanattribut vom zweiten Schema abgebildet. Ein anderes gut bekanntes Muster besteht aus

Fig. 5: Äquvalente Muster

zwei Strukturen, die miteinander durch eine referentielle Beziehung verbunden sind. Dies ist einer einfachen Struktur (Join von den zwei Strukturen) gleichwertig. Elementlevel Matching ist nicht beschränkt auf das Atomlevel, es kann aber angewandt werden auf eine grobkörnige Struktur, damit sind höher (nicht-atomare) gestufte Elemente gemeint. Beispiele für höhergstufte Granularitäten sind Aktenablage, Objekte, Klassen, relationale Tabellen und XML Elemente. Im Gegensatz zu einem Strukturlevel Matcher, wie ein Elementlevel Ansatz betrachtet man höhergestufte Elemente in Isolation, d.h. man ignoriert seine Substruktur und Komponenten. Ein Elementlevel Matching kann durch Algorithmen realisiert werden, diese sind so ähnlich wie der relationale Joinprozess. Abhängig vom Matchertyp, wobei der Matchabgleich auf Eigenschaften wie Name, Beschreibung oder Datentyp vom Schemaelement basiert. Dabei gilt für jedes Element vom Schema S1, das alle Elemente vom Schema S2 mit dem gleichen oder ähnlichen Wert der Matcheigenschaft erkannt sind. Eine allgemeine Implementation ist, ähnlich des geschachtelten Joinprozesses, indem man jedes Element von S1 mit jedem Element von S2 vergleicht und eine Änlichkeitsmetrick durch Elementpaare bestimmt. Für einen Matchkandidaten werden nur solche Kombinationen von Ähnlichkeitswerte über einer bestimmten Grenze berücksichtigt. Für spezielle Fälle sind leistungsfähigere Implementationen möglich, z.B. Equi-Joins. Die Implementation ähnlich einem Join ist für Hybridmatcher nutzbar, also dort wo man mehrere Eigenschaften zur selben Zeit berücksichtigt (z.B. Name und Datentyp).

6.1.2 Matchkardinalitäten

Ein S1 (oder S2) Element kann keinmal, einmal oder mehrmals als Mappingelement im Matchergebnis zwischen den beiden Eingabeschemata S1 und S2 teilnehmen. Außerdem können sich ein oder mehrere S1 Elemente mit einem oder mehreren S2 Elementen vermischen. Man kennt hier die allgemeinen Beziehungskardinalitäten, 1:1 und die satzorientierten Fälle 1:n, n:1 und n:m zwischen den Matchingelementen. Beide mit Rücksicht auf verschiedene Mappingelemente (globale Kardinalitäten) und mit Rücksicht auf ein einzelnes Mappingelement (lokale Kardinalität). Elementlevel Matching ist typischerweise eingeschränkt auf lokale Kardinalitäten 1:1, n:1 und 1:n. Üblicherweise erfordern n:m Mappingelemente die strukturelle Einbettung von Schemaelementen und deshalb benötigen sie einen Strukturlevel Matching. Figur 6 zeigt Beispiele für die vier lokalen Kardinalitätsfälle für einzelne Mappingelemente. In Reihe 1 ist die Verschmelzung von einer 1:1 Beziehung dargestellt. Frühere Arbeiten haben sich hauptsächlich mit dieser Beziehung beschäftigt, da es bei den anderen Fällen schwierig ist automatische Festlegungen für Mappingausdrücke zu erstellen. Vermischt man mehrere S1 (oder S2) Elemente zur selben Zeit, so sieht man das Ausdrücke benutzt werden um die Zusammenhängigkeit dieser Elemente darzustellen. Beispielsweise Reihe 3 erläutert wie FirstName und LastName aus Name gewonnen wurden. Ein anderes Beispiel wird in Reihe 4 gezeigt, dort wird ein SQL-Ausdruck benutzt, um Attribute von zwei Tabellen zu vereinen. Es setzt

	Local match cardinalities	S1 element(s)	S2 element(s)	Matching expression
1.	1:1, element level	Price	Amount	Amount = Price
2.	n:1, element-level	Price, Tax	Cost	Cost = Price*(1+Tax/100)
3.	1:n, element-level	Name	FirstName, LastName	FirstName, LastName = Extract (Name, . . .)
4.	n:1 structure-level (n:m element-level)	B.Title, B.PuNo, P.PuNo, P.Name	A.Book, A.Publisher	A.Book, A.Publisher = Select B.Title, P.Name From B, P Where B.PuNo=P.PuNo

Fig. 6: Matchkardinalitäten

sich aus einer m:n Beziehung aus dem Attributlevel (vier S1 Attribute werden mit zwei S2 Attribute verglichen) und einer n:1 Beziehung mit dem Strukturlevel (zwei Tabellen, B und P, werden mit S1 verglichen, wobei Tabelle A in S2 ist) zusammen. Die globalen Kardinalitätsfälle hinsichtlich aller Mappingelemente sind größtenteils orthogonal zu den Fällen mit einzelnen Mappingelementen. Das Beispiel in Reihe 1 ist ein globaler 1:1 Match, wenn kein Element von S1 mit Amount und kein Element von S2 mit Price verknüpft ist. Andererseits, wenn Price in S1 mit anderen S2 Elemente verknüpft ist (z.B. Kosten in Reihe 2) erhält man einen globalen 1:n Match in Kombination mit einem lokalen 1:1 oder 1:n Match. Für die ersten drei Beispiele in Tabelle 3 gilt, das eine S1 Instanz mit einer S2 Instanz verknüpft ist (1:1, Instanz-Level Match). Das Beispiel in Reihe 4 stimmt mit einem n:1 Instanz-Level Match überein, welcher zwei Instanzen kombiniert, eine B und P mit einer A-Instanz. Ein n:m Instanz-Level Verknüpfungsbeispiel ist die Verbindung einzelner Umsatzinstanzen von S2 mit verschiedenen Instanzen vom Gesamtumsatz (monatlich, vierteljährlich, etc.) von S2. Meistens existieren solche Ansätze, die jedes Element von einem Schema abbilden auf Elemente von einem anderen Schema, die eine höhere Ähnlichkeit haben. Dies sind Ergebnisse in den lokalen 1:1 Matches und den globalen 1:1 oder 1:n Mappings.

6.1.3 Linguistische Ansätze

Sprachbasierte- oder liguistische Matcher benutzen Namen und Texte (Wörter oder Sätze) um semantisch ähnliche Schemaelementen zu finden. Die zwei Schemalevel Ansätze, Namensmatching und Beschreibungsmatching, werden nun genauer betrachtet.

Namensmatching Namensbasierte Matchings gleichen Schemaelemente mit gleichem oder ähnlichem Namen an. Die Ähnlichkeit von Namen kann bestimmt werden und ist auf verschiedenen Wegen erfasst:

- Gleichheit von Namen
  Ein wichtiger Unterfall ist die Gleichheit von Namen aus dem gleichem XML Namensraum, wenn gewährleistet wird das der selbe Name tatsächlich von derselben Semantik unterstützt wird.

- Gleichheit von kanonischen Namensrepräsentationen nach Abstammung und anderen Vorverarbeitungen.
  Dies ist wichtig wenn man sich mit speziellen Präfix/Suffix Symbolen (z.b. CName → customer name) beschäftigt.

- Gleichheit von Synonymen. z.B. car ≅ Automobil

- Gleichheit von Hypernyme. Hypernym bedeutet, wenn A ein Hypernym von B ist, dann ist B eine Art von A, z.B. der Hypernym von Eichenholz beinhaltet Baum und Betrieb. Ein konkreteres Beispiel ist: Buch is-a Publikation und Artikel is-a Publikation impliziert Buch ≅ Publikation, Artikel ≅ Publikation und Buch ≅ Artikel

- Ähnlichkeit von Namen, Dies basiert auf allgemeine Substrings, aufbereitet durch Distanzen, Aussprachen, Geräuschbeispiele, z.B. representedBy ≅ representative, ShipTo ≅ Ship2.

- vom Benutzer zur Verfügung gestellte Namensmatcher z.B. reportsTo ≅ Manager

Gewonnene Synonyme und Hypernyme erfordern das Thesauri oder Wörterbücher benutzt werden. Grundsätzlich können der natürlichen Sprache Wörterbücher nützlich sein, vielleicht sogar mehrsprachige Wörterbücher (z.B. Englisch-Deutsch) um mit Eingabeschemas von verschiedenen Sprachen fertig zu werden. Namensmatching kann benutzt werden für gebiets- oder unternehmensspezifische Wörterbücher. Für eine is-a Taxonomie gilt, das sie gewöhnlich Namen, Synonyme, Beschreibungen von Schemaelementen und Abkürzungen enthalten. Diese spezifischen Wörterbücher erfordern einen wesentlichen Aufwand. Besonders für Schema mit relativ flacher Struktur ist dies wichtig, denn Wörterbücher stellen den größten Wert für Matchinghinweise dar. Homonyms sind gleiche oder ähnliche Namen, dies gilt für verschiedene Elemente. Daraus folgt, das Homonyme einen Matchingalgorithmus irreführen können. Homonyme können Teil der natürlichen Sprache sein, wie „stud“, dies bedeutet eine Halterung oder männliches Pferd. Sie können aber auch verschiedene Gebiete meinen, wie „line“ bedeutet eine Businesskette oder Einzelposten von einem Auftrag. Ein Namensmatcher kann die Anzahl der falschen Matchkandidaten reduzieren durch die Gewinnung von Fehlerinformation, die durch den Benutzer oder Wörterbücher erhalten werden. Ein Matcher kann einen Warnhinweis bekommen, wenn ein Name mehrere Bedeutungen hat. Eine automatisierte Benutzung von Fehlerinformationen ist beim Benutzen von Kontextinformationen möglich. Solch eine Technik verwischt die Trennung zwischen den linguistikbasierten und strukturbasierten Techniken. Namensbasiertes Matching ist möglich für Elemente mit verschiedenen Granularitätsleveln. Es wird genutzt für globale Levels, z.B. für ein lowerlevel Schemaelement, das den Namen von einem Schemaelement berücksichtigt (z.B. zu finden das author.name ≅ AuthorName). Dies ist ähnlich einer kontextbasierten Mehrdeutigkeit von Homonymen. Namensbasiertes Matching ist nicht beschränkt auf das Erkennen von 1:1 Matches. Es kann mehrere wichtige Matches für ein gegebenes Schemaelement erkennen, z.B. „phone“ kann angepasst werden auf „home phone“ und „office phone“. Namensmatching kann durch Elementlevel Matching gesteuert werden. In dem Fall von Synonymen und Hypernymen schließt ein Join ähnlicher Prozess ein Wörterbuch D aus einer weiteren Eingabe ein. Wenn man an eine beziehungsähnliche Präsentation denkt, wie

S1 (name, ...) → eine Reihe vom S1 Schemaelement
S2 (name, ...) → eine Reihe vom S2 Schemaelement
D (name1, name2, similarity) → Ähnlichkeitsstand für [name1, name2]
zwischen 0..1

so kann man eine Liste von Matchkandidaten durch folgenden drei Wege Join Operation generieren.

Select S1.name, S2.name, D.similarity
From S1, S2, D
Where (S1.name = D.name1) and (D.name2 = S2.name)
and (D.similarity > threshold)

Man nimmt an das D alle relevanten Paare von der Umwandlung über ähnliche Namen enthält, z.B. wenn A-B-0.9 und B-C-0.8 in D sind, dann würde man erwarten das D auch B-A-0.9, C-B-0.8 enthält und möglicherweise A-C-Σ, C-A-Σ. Intuitiv würde man den Ähnlichkeitswert Σ von .9 x .8 = .72 erwarten, aber dies ist abhängig vom Ähnlichkeitstyp, die Benutzung von Homonyme und vielleicht andere Faktoren.

Beschreibungsmatching Schema enthalten Bemerkungen in natürlicher Sprache, um die geplanten Semantiken von Schemaelementen auszudrücken. Diese Bemerkungen können auch liguistisch ausgewertet werden, um die Ähnlichkeit zwischen Schemaelementen auszuwerten. Beispielsweise würde es helfen die folgenden Matchelemente, durch eine linguistische Analyse von den Bemerkungen, verbunden mit jedem Schemaelement zu erkennen:

S1: empn → Angestelltenname
S2: name → Name des Angestellten

Diese linguistische Analyse könnte einfach durch entfernen von Schlüsselwörter von der Beschreibung sein, welche benutzt werden für Synonyme Vergleiche, wie Namen.

6.1.4 Constrainedbasierte Ansätze

Oft enthalten Schemata Beschränkungen zum Definieren von Datentypen und Wertebereichen, Eindeutigkeiten, Beziehungstypen und Kardinalitäten, etc. Wenn beide Eingabeschematas solche Information enthalten, kann es von einem Matcher zum Bestimmen der Ähnlichkeit von Schemaelementen [LNE89] benutzt werden. Etwa kann die Ähnlichkeit basieren auf der Äquivalenz von Datentypen und Bereichen, von Schlüsselcharakteristiken (1:1 Beziehungen) oder auf is-a Beziehungen. Die Realisierung ist oft ausgeführt wie in 5.1.1.(Granularität von Matcher) beschrieben, mit einem joinähnlichen Elementlevel Matching. Hier benutzt man die Datentypen, Strukturen und Beschränkungen in den Vergleichen. Gleiche Datentypen und Namensbeschränkungen (String ≅ varchar, Primärschlüssel ≅ unique) können durch eine spezielle Synonymtabelle zur Verfügung gestellt werden. In

S1 elements	S2 elements
Employee	Personnel
EmpNo – int, primary key	Pno - int, unique
EmpName – varchar (50)	Pname – string
DeptNo – int, references Department	Dept - string
Salary - dec (15,2)	Born - date
Birthdate – date	
Department	
DeptNo – int, primary key	
DeptName – varchar (40)	

Fig. 7: Constrainbasierte Matchings

Figur 7 deuten die Typ- und Schlüsselinformationen an, das Born an Birthdate angepaßt und Pno auf EmpNo oder DeptNo angepaßt wird. Das übrig bleibende S2 Element Pname und Dept sind Strings und deshalb wahrscheinlich angepaßt an EmpName oder DeptName. Wie das eben genannte Beispiel demonstriert, benutzt man Beschränkungsinformationen, nur führt dies oft zu fehlerhaften n:m Matches (Matchcluster). Dennoch hilft der Ansatz die Anzahl der Matchkandidaten zu begrenzen und diese mit anderen Matcher (z.B. Namensmatcher) zu kombinieren. Zuverlässige strukturelle Informationen, wie eine interne-Schema Referenz (Fremdschlüssel) und benachbarte relevante Information (part-of Beziehungen) können als Beschränkungen interpretiert werden. Diese Informationen erklären welche Elemente zum selben higher-level Schemaelement gehören, transitiv durch mehrstufige Strukturen. Solche Beschränkungen werden als Strukturen interpretiert und durch das Benutzen von strukturelle Matching Ansätze gewonnen. Ein Matching kann die Topologie der Strukturen als Quelle aus verschiedenen Elementtypen (z.B. für Attribute, Tabellen / Elemente oder Bereiche) und auch verschiedenen Typen von strukturellen Verbindungen (z.B. part-of oder verwendete Beziehungen) berücksichtigen. Viele Schemastrukturen sind hierarchisch und basieren auf dem Gestalten von einigen „Abwehrbeziehungen“. Wenn ein Match ausgeführt wird, basierend auf hierarchischen Strukturen, dann kann ein Algorithmus die Struktur entweder top-down oder bottom-up durchlaufen. Ein top-down Algorithmus ist gewöhnlich rechenintensiver als bottom-up, weil Matcher auf höheren Ebenen der Schemastruktur die Auswahl für Matchings auf einer feineren Strukturebene zu den Kombinationen mit Matching-Vorfahren abgrenzen. Ebenfalls kann ein top-down Algorithmus irreführend sein, wenn top-level Schemastrukturen sehr verschieden sind, eben wenn feiner strukturierte Elemente angepasst sind. Als Kontrast vergleicht ein bottom-up Algorithmus, alle Kombinationen von feiner strukturierten Elemente und findet folglich Matcher von diesem Level, sogar wenn sich Zwischen- und höhere Levelstrukturen gravierend unterscheiden. Auf Figur 7 beziehend, die vorher erkannten atomlevel Matches sind nicht ausreichend, um zum richtigem Match S1 zu S2 zu kommen, weil man einen Join zwischen S1.Employee und S1.Department zum Erhalten von S2.Personnel nötig ist. Dies kann automatisch durch die Beobachtung, dass Komponenten von S2.Personal sowohl zu Komponenten von S1.Employee als auch zu S1.Department zueinander korrespondieren und dass S1.Employee und S1.Department miteinander über den Fremdschlüssel DeptNo in Employee, welcher auf Deparment referenziert, verbunden sind. Dies erlaubt uns das richtige n:m SQL-ähnlichen Verknüpfungsmapping zu erstellen. Dies stellt ein folgendes Beispiel dar.

S2.Personnel (Pno, Pname, Dept, born) ≅
Select S1.Employee.EmpNo, S1.Employee.EmpName, S1.Department.DeptName,
    S1.Employee.Birthdate
From S1.Employee, S1.Department
Where (S1.Employee.DeptNo = S1.Department.DeptNo)

Einige Folgerungen braucht, man um zu wissen ob ein Join hinzugefügt werden soll. Diese Folgerung kann durch Mapping des Problems erledigt werden. Dabei werden die erforderten Joins in das universelle Relationsmodell [KKFG84] verlegt.

6.1.5 Wiederverwendung von Schema und Mappinginformation

Ein Weg um Hilfsinformationen in ein Eingabeschema einzufügen, ist das Benutzen von Thesauri, Wörterbücher und vom Benutzer erstellte Matches oder Informationen zur

Fehleranpassung. Ein anderer Weg ist die Effektivität von Matchs zu verbessern, dies beruht auf der Wiederverwendung von allgemeinen Schemakomponenten und vorhandenen Mappings. Reuse-orientierte Ansätze sind viel versprechend, seit man erwartet das viele Schematas aufeinander abgestimmt werden müssen und das Schematas oft sehr ähnlich sind zu anderen und früheren Schematas. Beispielsweise im E-Commerce werden Substrukturen oft wiederholt, innerhalb verschiedener Nachrichtenformate (z.B. Adress- und Namensfeld). Das Benutzen von Namen in XML Namensräumen oder spezifischen Wörterbüchern ist immer reuse-orientiert. Eine allgemeine Vorgehensweise ist nicht nur die Wiederverwendung global definierter Namen, sondern auch von ganzen Schemafragmenten, die Eigenschaften wie Datentypen, Schlüssel und Beschränkungen beinhalten. Dies ist besonders lohnend für häufig benutzte Elemente, wie Adresse, Kunde, Angestellter, erworbene Position und Rechnung, welche in einer Schemabibliothek geführt werden. Da es äußerst unwahrscheinlich ist, dass die gesamte Welt sich an einem Schema orientiert, definiert man diese für ein Unternehmen, deren Handelspartner, relevante Grundbausteine oder ähnliche Organisationen um die Anzahl der Freiheitsgrade einzuschränken. Schemaeditoren sollten auf diese Bibliotheken zugreifen um die Wiederverwendbarkeit von vordefinierten Schemafragmenten und definierten Termen zu unterstützen. Die Elemente die auf diesem Weg wiederverwendet werden, sollten die ID oder ihre Ursprungsbibliothek enthalten, z.B. via XML Namensräume, so kann die Implementation von Matchs einfacher erkannt werden und Schemafragmente und Namen die von der gleichen Bibliothek kommen angleichen. Ein weiterer generischer Ansatz ist es existierende Mappings zu benutzen. Dabei verwendet man früher erhaltene elementlevel Matches, welche einfach zum Thesaurus hinzugefügt werden können, man nutzt also ganze Strukturen, diese sind nützlich wenn man verschiedene aber gleichartige Schematas zum selben Zielschema auftreten kann, wenn eine neue Quelle in ein Data-Warehouse oder digitalen Bibliothek integriert wird. Ein Beispiel für die Wiederverwendung ist in Figur 8 erläutert. Die Matchergebnisse

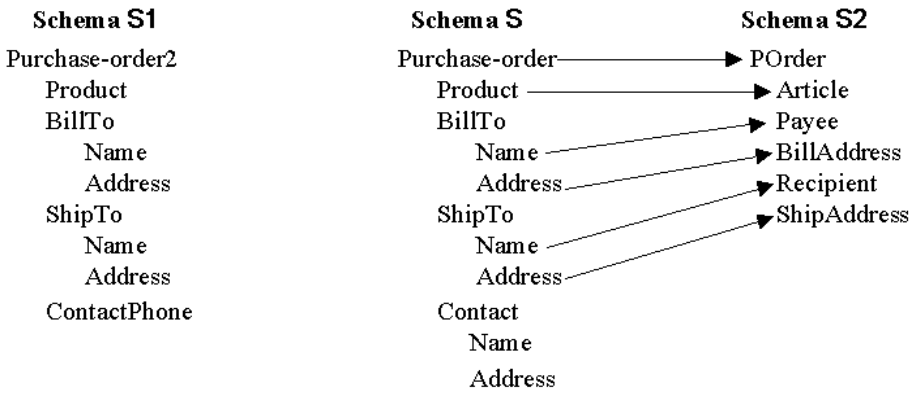

Fig. 8: Szenario für das Wiederverwenden von existierenden Mappings

zwischen S und S2 sind durch einen Pfeil dargestellt. Das neu erhaltene Auftragsschema S1 ist ähnlich zu S. Auf diese Weise erreicht man, dass jedes Element oder jede Struktur von S1 ein entsprechendes Element oder eine volle Matching Struktur in S hat. Dazu benutzt man die existierenden Mappings zwischen S und S2. In diesem (idealem) Fall, verwendet man alle Matches wieder; inzwischen ist S2 voll abgedeckt, so das keine zusätzliche Angleichungsarbeit nötig ist. Solch eine Wiederverwendung von früheren Matches kann nur für einige Teile von einem neuen Schema möglich sein. es ist ein Matchproblem in sich. Das Hauptproplem darin besteht, welcher Teil eines neuen Schemas ähnlich ist zu einem Teil eines früher verbundenen, also ein Matchproblem in sich. Vielmehr sind

Ähnlichkeitsmerkmale, die in einem früheren Ableich bestimmt wurden, wo möglich an den Anwendungsbereich gebunden, so dass deren Wiederverwendung für verwandte Anwendungen eingeschränkt sein kann. Im Extremfall sind keinerlei Schemainformationen gegeben, aber ein Schema kann entweder manuell, oder automatisch durch Instanz-Daten erstellt werden. Ein Beispiel dazu ist die Berücksichtigung von Gehalt und Einkommen in einer Lohn- und Gehaltsanwendung, aber nicht in einer Steuerberichtsanwendung.

6.2 Instanzlevel Anwendungen

Instanzlevel Daten geben eine wichtige Einsicht in den Kontext und die Bedeutung von Schemaelementen. Dies ist besonders wichtig wenn hilfreiche Schemainformationen begrenzt sind, wie für semistrukturierte Daten. Ein extremer Fall ist, wenn kein Schema gegeben ist, aber ein Schema kann von Instanzdaten manuell oder automatisch konstruiert werden (z.B. eine „Datenübersicht“ [GW97] oder ein genäherter Schemagraph [WYW00] kann automatisch von XML Dokumenten konstruiert sein). Gerade wenn wesentliche Schemainformationen verfügbar sind, kann das Benutzen von Instanzlevel Matching sinnvoll sein, um fehlerhafte Interpretationen von Schemainformation abzudecken. Die meisten schon besprochenen Ansätze für Schema-Level-Abgleiche können zum Instanz-Level-Abgleichung verwendet werden. Wie auch immer, einige sind besonders hier zutreffend, z.B.:

- Für Textelemente ist eine sprachliche Charakterisierung begründet auf Information-Retrieval-Techniken der bevorzugte Ansatz, beispielsweise durch Extraktion von Schlüsselwörtern und Themen, basierend auf der relativen Häufigkeit von Wörtern und Wortkombinationen usw. Dies wird in Figur 7 dargestellt, sieht man sich Dept, DeptName und EmpName an, kann man beispielsweise daraus schließen das DeptName ein besserer Matchkandidat für Dept ist als EmpName.

- Für strukturiertere Daten, wie Nummerische und Stringelemente kann man eine constrainbasierte Charakterisierung ansetzen, wie ein numerischer Wertebereich und Mittelwerte oder Charakter-Muster. Diese können von Telefonnummern, Postleitzahlen, geographische Namen, Adressen, ISBNs, SSNs, Dateneingaben oder geld zugehörige Einträge(z.B: basierend auf Währungssymbolen) wieder erkannt werden. In Figur 7 wird gezeigt, das Instanzinformationen helfen können EmpNo zum primären Matchkandidaten für Pno zu machen, z.B. basierend auf ähnlichen Wertebereichen entgegen dem Wertebereich für DeptNo.

Die Hauptanwendung von ausgewerteten Instanzen ist eine genaue Charakterisierung von aktuellen Kontexten der Schemaelementen. Eine Vorgehensweise ist, die Beschreibung zum Verbessern des Schemalevel Matchers zu benutzen, z.B. kann ein Contraintbasierter Matcher genauer die entsprechenden basierten Datentypen bestimmen. Beispielsweise basierend auf den erkannten Wertebereichen und Charakter-Mustern, dadurch wird die Effektivität von Matches verbessert. Diese beschreibt den Inhalt von beiden Eingabeschemata und den Abgleich der Schemata miteinander. Ein weiterer Ansatz ist das Ausführen Instanzlevel Matchings auf sich selber. Zuerst werden die Instanzen von S1 ausgewertet, um die Inhalte von S1 Elementen zu erkennen. Danach werden die S2 Instanzen Schritt für Schritt an die Charakterisierung von S1 Elementen angeglichen. Die durch die Instanzen erzeugten Ergebnisse müssen vermischt und zum Schemalevel abstrahiert sein, um eine Rangliste von Matchkandidaten in S1 für jedes Schemalevel Element in S2 zu

erzeugen. Verschiedene Ansätze haben solche Instanzmatchings oder Klassifikationen vorschlagen, dies dient zur Bestimmung, neuraler Netze und maschinell lernenden Techniken [BM01], [DDL00], [DDH01], [LC00], [LCL00]. Instanzlevel Matching kann auch durch nutzen von Hilfsinformation ausgeführt werden, z.B: frühere Mappings erhalten einen Abgleich durch verschiedene Schematas. Dieser Ansatz ist besonders hilfreich für den Abgleich von Textelementen, durch das zur Verfügung stellen von Matchkandidaten für individuelle Schlüsselwörter, z.B. eine frühere Analyse kann zeigen, das das Schlüsselwort „Microsoft“ häufig auftritt für die Schemaelemente „CompanyName“, „Hersteller“, etc. Für eine neuen Matchaufgabe kann dies benutzt werden, um „CompanyName“ in S1 als einen Matchkandidaten für X zu erzeugen, sogar wenn „Microsoft“ selten in der Instanz von S1 auftritt. Dies ist sogar dann der Fall, wenn ein S2 Schemaelement X häufig den Term „Microsoft“ enthält. Die Hauptansätze für Instanzlevel Matching arbeitet in erster Linie um Elementlevel Matcher zu finden. Gefundene Matcher für Mengen von Schemaelementen oder Strukturen würden eine Charkterisierung des Inhaltes von diesen Mengen erfordern. Das Hauptproblem ist die große Anzahl der möglichen Kombinationen von Schemaelementen, die durch die Instanzen ausgewertet werden müssen.

6.3 Kombination verschiedener Matcher

Für einen Matcher, der nur einen Ansatz benutzt, ist es unwahrscheinlich das er viele gute Matchkandidaten erschafft. Es gibt zwei Möglichkeiten: ein Hybridmatcher, welcher viele Matchingkriterien kombiniert und einen zusammengesetzten Matcher, dieser kombiniert die Ergebnisse von einzeln abgearbeiteten Matchern. Kombiniert man mehrere Matchingansätze, so ist es oft möglich diese simultan oder in einer spezifischen Position zu bewerten.

6.3.1 Hybridmatcher

Hybridmatcher werden direkt kombiniert mit verschiedenen Matchingansätzen um Matchingkandidaten zu bestimmen, die auf mehreren Kriterien oder Informationsquellen basieren. Sie sollten bessere Matchkandidaten zur Verfügung stellen und bessere Leistungen erzielen, als die separate Abarbeitung von mehreren Matchern. Effektivität kann verfeinert werden, weil schwache Matchkandidaten nur einen von verschiedenen Kriterien abgleichen, deshalb können sie früher heraus gefiltert werden und weil komplexe Matcher die gemeinsame Überlegung von mehreren Kriterien erfordern. Beispielsweise die Benutzung von Schlüsseln, Datentypen und Namen in Figur 4. Strukturlevel Matching ist auch vorteilhaft, da es mit anderen Ansätzen wie Namensmatching verbunden ist. Ein Weg um Struktur- mit Elementlevel-Matching zu kombinieren, ist einen Algorithmus zu benutzen, um ein Teilmapping zu erstellen und einen anderen um das Mapping fertig zustellen. Ein Hybridmatcher ist meist besser anzuwenden als die Abarbeitung von mehreren Matchern. Dabei wird die Anzahl der übergebenen Schema verringert, z.B. kann man mit elementlevel angepaßte Hybridmatcher mehrere Kriterien zur selben Zeit von jedem S2 Element kontinuierlich mit dem nächsten S2 Element testen.

6.3.2 Verbundmatcher

Auf der anderen Seite, kann man auch einen Verbundmatcher benutzen , dieser kombiniert die Ergebnisse von verschiedenen einzelnen getesteten Matcher, einschließlich Hybridmatcher. Ein Hybridmatcher benutzt typischerweise fest verknüpfte Kombinationen von ein-

zelnen Matchingtechniken, diese werden simultan oder in einer starren Folge abgearbeitet. Im Gegensatz dazu, erlaubt uns ein Verbundmatcher aus einem Verzeichnis von mehreren Matchern auszuwählen, z.B. Anwendungsgebiete oder Schemasprachen. Beispielsweise könnte man eine Lernmaschine dazu benutzen einzelne Matcher zu kombinieren, wie in [DDH01] für Instanzlevel Matcher und in [EJX01] für eine Kombination von Instanz- und Schemalevel-Matcher. Außerdem sollte ein Verbundmatcher eine flexible Folge von Matchern erlauben, so das sie entweder simultan oder sequentiell abgearbeitet werden können. Ein weiterer Fall ist, wenn das Matchergebnis von einem ersten Matcher verwendet und durch einen zweiten Matcher erweitert wird, um eine iterative Verbesserung von Matchergebnissen zu erhalten. Die Auswahl von Matchern, die Bestimmung ihrer Abarbeitungsreihenfolge und die Kombination von einzelnen Matchergebnissen, kann entweder automatisch durch Implementation von Matches oder Clients, z.B. Werkzeuge oder manuell durch einen Benutzer erfolgen. Ein automatischer Ansatz kann die Anzahl der Beeinflussungen durch den Nutzer reduzieren, aber es ist schwierig solch eine generische Lösung zu bekommen, da diese an verschiedenen Anwendungsgebieten angepaßt werden kann. Manchmal kann dieser Ansatz auch durch Anpassungsparameter bestimmt werden. Alternativ dazu kann ein Benutzer direkt den Matcher der durchgeführt werden soll, seine Durchführungsreihenfolge und wie seine Ergebnisse kombiniert werden auswählen. Solch ein manueller Ansatz ist einfacher zu implementieren und wird durch den Benutzer kontrolliert. Die Beeinflussung durch den Benutzer ist in einigen Fällen nötig, weil die Implementation von Matches nur Matchkandidaten bestimmen kann, die der Benutzer annimmt, zurückweist oder ändert. Komplexe Matchaufgaben befassen sich damit, eine Implementation von Matcher für interaktive Entwicklungen von Matchergebnissen mit mehreren Benutzerinteraktionen zu erhalten. Ein Verbundmatch-Ansatz unterstützt die sequentielle Abarbeitung von Matchern. Die vom Benutzer vorgegebenen Matcher werden als ein spezieller Matcher berücksichtigt, dies wird von automatischen Matchern zur Verfügung gestellt. Fest steht, das Matcher vorgegebene Matcheingaben haben, die vom Benutzer eingegeben wurden und diese Eingaben können nicht geändert werden, aber man konzentriert sich auf die unverknüpften Teile von Eingabeschematas.

6.4 Corpusbasiertes Schemamatsching [MBDH05]

Corpus-basiertes-Matching ist ein Ansatz, der eine Sammlung von Schemen und Mappings diverser Anwendungsbereiche einsetzt, um die Robustheit von Schemamatch-Algorithmen zu verbessern. Ein Korpus bietet ein Warenlager von alternativen Repräsentationen von Konzepten einer Domäne an und kann deshalb für die unterschiedlichsten Zwecke verwendet werden. Der Korpus ist eine Sammlung von Schemen und Mappings zwischen zwei Paaren von Schemen. Jedes Schema enthält seine Elemente (wie Relationen und Attributnamen), deren Datentypen, Beziehungen zwischen Elementen, Instanzen von Beispieldaten und sonstiges Wissen, welches dazu verwendet werden kann, es mit anderen Schemen zu verknüpfen. Schemen im Korpus sind lose miteinander verbunden, sie gehören annähernd zu einer Domäne, müssen aber nicht zwingend aufeinander abgebildet werden. Angenommen man hat ein Element s aus dem Schema S, welches nicht im Korpus enthalten ist, dann findet man andere Elemente, beispielsweise e und f, im Korpus, die alternative Repräsentationen des selben zugrunde liegenden Konzeptes sind. Die Elemente e und f werden sich von s in gewisser Weise unterscheiden, da sie zu unterschiedlichen Schemen gehören. Zum Beispiel kann e den selben Datentyp wie s besitzen, aber sich im Namen unterscheiden und wird folglich zu einem wesentlich generellen Namensmodell von s beitragen. Im Gegensatz dazu kann f von ähnlichen Namen und Datenstruktur sein,

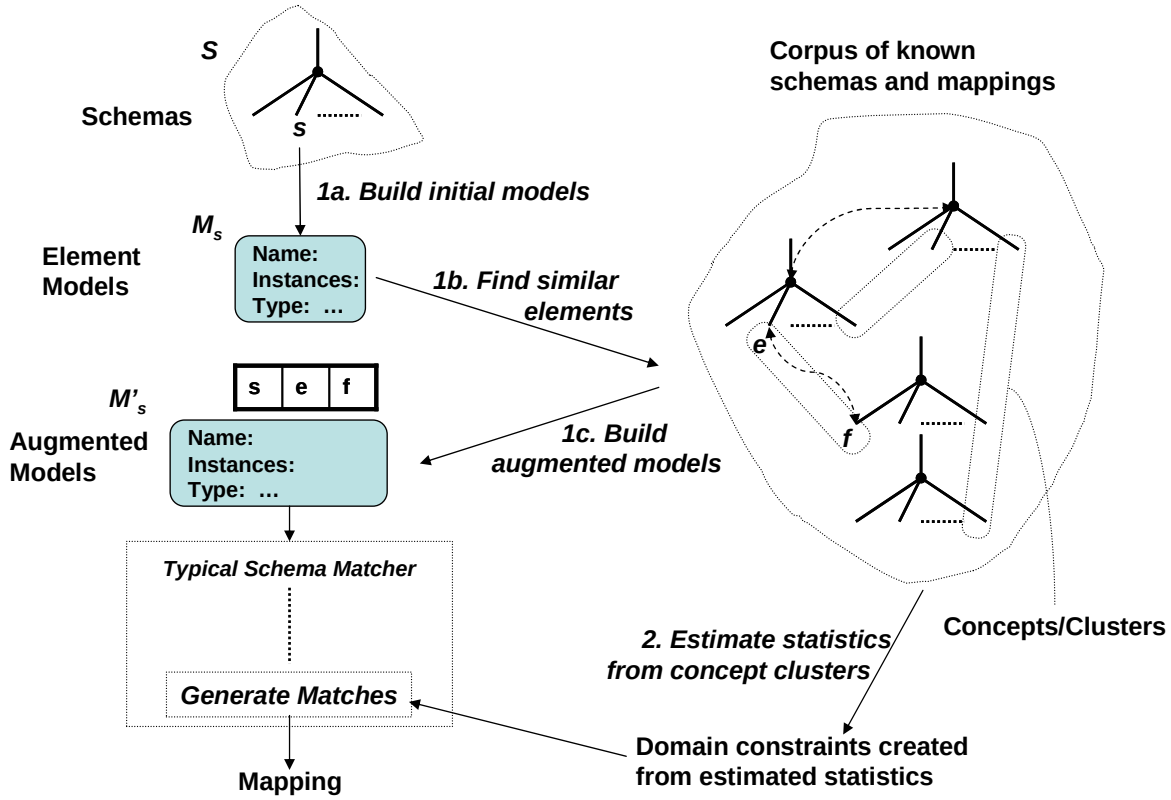

Fig. 9: Beschreibung der Vergrößerungsmethode

aber in unterschiedliche Beziehungen zu anderen Elementen auf diesem Gebiet haben und somit zu einem besseren Beziehungsmodell beisteuern. Die Anreicherungsmethode erstellt ein Modell M'_s für s, welches das Wissen über e und f beinhaltet und benutzt M'_s im Matching-Prozess anstelle von M_s. Figur 9 stellt dies näher dar. Eine weitere Anwendung des Korpus ist die Aufstellung diverser Statistiken über Elemente und Beziehungen in einer Domäne. Dies liefert uns ein besseres Verständnis der einzelnen Domänen.

7 Beispielansätze aus der Literatur

7.1 Prototypschemamatcher

		SemInt [LC94, LC00, LCL00]	**LSD [DDL00, DDH01]**	**SKAT [MWJ99, MWK00]**	**TranScm [MZ98]**	**DIKE [PSU98a,b, PSTU99]**	**ARTEMIS [CDD01, BCC*00]**	**CUPID [MBR01]**
Schema types		relational, files	XML	XML, IDL, text	SGML, OO	ER	relational, OO, ER	XML, relational
Metadata representation		unspecified (attribute-based)	XML schema trees	graph-based OO data model	labeled graph	graph	hybrid relational / OO data model	extended ER
Match granularity		element-level: attributes (attribute clusters)	element and structure-level	element/ structure-level: attributes / classes	element-level	element/ structure-level: entities / relationships / attributes	element/ structure-level: entities / relationships / attributes	element and structure-level
Match cardinality		1:1	1:1	1:1 and n:1	1:1	1:1	1:1	1:1 and n:1
Schema-level match	Name-based	-	name equality / synonyms	name equality; synonyms; homonyms; hypernyms	name equality; synonyms; homonyms; hypernyms	name equality; synonyms; hypernyms	name equality; synonyms; hypernyms	name equality, synonyms, hypernyms, homonyms, abbreviations
	Constraint-based	several criteria: data type, length, key info, ...	-	is-a (inclusion); relationship cardinalities	is-a (inclusion); relationship cardinalities	domain compatibility	domain compatibility. In MOMIS, uses keys, foreign keys, is-a, aggregation	data type and domain compatibility, referential constraints
	Structure matching	-	XML classifier for matching non-leaf elements [DDH01]	similarity w.r.t. "related" elements	similarity w.r.t. "related" elements	matching of neighborhood	matching of neighborhood	matching subtrees, weighted by leaves
Instance-level matchers	Text-oriented	-	Whirl [Co98], Bayesian learners	-	-	-	-	-
	Constraint-oriented	character / numerical data pattern, value distribution, averages	list of valid domain values	-	-	-	-	-
Reuse / auxiliary information used		-	comparison with training matches; lookup for valid domain values	reuse of general matching rules	-	provision of some synonyms + inclusions with similarity probabilities	thesauri	thesauri, glossaries
Combination of matchers		hybrid	composite matcher with automatic combination of matcher results	hybrid	hybrid matchers; fixed order of matchers	hybrid	hybrid	hybrid

Fig. 10: Charakteristiken von genannten Schemamatchansätzen

Figur 10 und 11 zeigen 7 Prototypimplementationen, deren Klassifikationskriterien in Kapitel 5 gegeben wurden. Die Tabelle gibt uns an, welche Teile der Lösungsräume sich auf welche Implementationen beziehen. Es werden also die Hilfsschematypen, das innere Metadatenrepräsentationsformate, die manuell durchführbaren Aufgaben und die Anwendungsgebiete vorgeschrieben. Die Tabelle zeigt das alle Systeme mehrere Matchingkriterien erfüllen, sechs in der Form von einem Hybridmatcher und nur einer, LSD, durch einen

	SemInt [LC94, LC00, LCL00]	**LSD [DDL00, DDH01]**	**SKAT [MWJ99, MWK00]**	**TranScm [MZ98]**	**DIKE [PSU98a,b, PSTU99]**	**ARTEMIS [CDD01, BCC*00]**	**CUPID [MBR01]**
Manual work / user input	selection of match criteria (optional); selection of matching attributes from attribute clusters	user-supplied matches for training sources; user can specify tuning parameters and integrity constraints to guide selection of match candidates [DDH01]	match / mismatch rules + iterative refinement	resolving multiple matches, adding new matching rules	resolving structural conflicts (preprocessing)	user can adjust weights in match calculations	user can adjust threshold weights
Application area	data integration	data integration with pre-defined global schema	ontology composition for data integration / interoperability	data translation	schema integration	schema integration	data translation, but intended to be generic
Remarks	neural networks; C implementation		"algorithms" implicitly represented by rules	rules implemented in Java	algorithms to calculate new synonyms, homonyms, similarity metrics	also embedded in the MOMIS mediator, with extensions	

Fig. 11: Fortsetzung der Charakteristiken von genannten Schemamatchansätzen

Verbundmatcheransatz. Eine flexible Reihenfolge von verschiedenen Matchern wurde bisher noch nicht unterstützt. Die meisten Systeme stellen beide, Stukturlevel und Elementlevel Matching, in einzelnen Namens- und constrainedbasierten Matching zur Verfügung. Wie auch immer, nur zwei von sieben Systemen berücksichtigen Instanzdaten und alle Systeme betrachten lokale 1:1 Matches, aber zwei Systeme unterstützen globale n:1 Matches. Die meisten Prototypen wurden für ein spezifisches Anwendungsgebiet entwickelt, meistens durch Daten und Schemaintegration. Viele Systeme unterstützen mehrere Schematypen, eine Zeit lang war LSD beschränkt auf XML und DIKE auf ER-Sources. Alle Systeme erlauben den Benutzer erzeugte Matchergebnisse zu bestätigen und aber diese erforderten eine manuelle Nacharbeit der Instrumente vom System, z.B. durch zur Verfügung stellen von früheren Matchwissen oder Parametereinstellungen ähnlicher Grenzen. Die Hauptform der Hilfsinformation und deren Wiederverwendung werden unterstützt durch die Vorschriften vom Thesauri, Glosaren und die Angabe von spezifischem Matchwissen. Wiederverwendbarkeit von früheren Matchergebnissen wird noch nicht unterstützt.

7.2 SemInt - Northwestern Univ.

Der SemInt Matchprototyp [LC00], [LCL00] erschafft ein Mapping zwischen verschiedenen Attributen von zwei Schemata, bei denen z.B. die Matchkardinalität 1:1 ist. Es nutzt 15 konstrainbasierte und 5 kontentbasierte Matchingkriterien aus. Die Schemalevel-Beschränkung benutzt die Information, die von einer relationalen DBMS verfügbar ist. Instanzdaten werden benutzt um diese Information zur Verfügung zu stellen und aktuelle Wertebeschreibungen zu verbessern, wie numerische Mittelwerte etc. Für jedes Kriterium benutzt das System eine Funktion, um jeden möglichen Wert auf das Intervall von [0...1] zu mappen. Man benutzt diese Funktion, SemInt um für jedes Attribut eine Matchsignatur zu bestimmen, welche aus den Werten im Intervall von [0...1] für N Matchingkriterien besteht. Inzwischen korrespondieren Signaturen mit Punkten im n-dimensionalen Raum. Die euklidische Distanz zwischen Signaturen kann benutzt werden als ein Maß für den

Grad der Ähnlichkeit und auf diese Weise für die Bestimmung einer Reihenfolgeliste von Matchkandidaten. In diesem Hauptansatz benutzt SemInt neurale Netze zum Bestimmen von Matchkandidaten. Clustering ist automatisiert durch Zuweisung aller Attribute mit einer Distanz unter einem Grenzwert zu dem gleichem Cluster. Das neurale Netz ist mit den Signaturen vom Clusterzentren ausgebildet. Die Signaturen von Attributen vom zweitem Schema werden in das neurale Netz überführt, um den besten Attribut-Cluster vom ersten Schema zu bestimmen. Die Autoren fanden Aufgrund ihrer Experimente heraus, dass der einfache Matchansatz auf der euklidischen Distanz basiert, welcher fast alle identischen Attribute findet. Unterdessen sind neurale Netze besser zur Bestimmung weniger ähnlichen Attributen dieser Matches. Wie auch immer, der neurale Netz Ansatz hat umfangreiche Darstellungsprobleme für größere Schemaleistungen [CHR97]. Um die Effizienz dieses Ansatzes zu verbessern, erkennt ein Match nur Attribut-Cluster, dabei wählt der Benutzer die Matchingattribute vom Cluster aus. SemInt repräsentiert einen leistungsstarken und flexiblen Ansatz von Hybridmatchern, dabei können mehrere Matchkriterien ausgewählt und ausgewertet werden. Dies ist ein Kontrast zu anderen Hybridmatchern, die verschiedene Kriterien in einer stark zusammenhängenden Art und Weise nutzen. SemInt unterstützt kein namensbasiertes oder Graphmatching, bei diesen ist es schwierig ein brauchbares Mapping im [0..1] Intervall zu bestimmen.

7.3 LSD Univ. of Washington

Das LSD (lernende Quellbeschreibungen) System benutzt lernende Maschinentechniken um eine neue Datenquelle anzugleichen gegenüber einem früher bestimmten globalen Schema, welcher ein 1:1 atomarlevel Mapping bildet [DDL00], [DDH01]. Es repräsentiert ein Verbundmatch Schema mit einer automatischen Kombination von Matchergebnissen. In Ergänzung zu einem Namensmatcher der verschiedene Instanzlevel-Matcher benutzt, sind diese außerhalb eines Vorprozessschrittes ausgebildet. Ein vom Kunden vorgegebenes Mapping für eine Datenquelle zu dem globalen Schema, der Vorprozessschritt sieht aus wie eine Instanz für diese Datenquelle um den Lernenden auszubilden. Dadurch entdeckt man charakteristische Instanzmuster und Matchingregeln. Diese Muster und Regeln können dann angewandt werden um andere Datenquellen zu dem globalen Schema anzugleichen. Eine neue Datenquelle ist dadurch gegeben, das jeder Matcher eine Liste von Matchkandidaten bestimmt. Ein globaler Matcher der die selbe maschinenlernende Technologie benutzt, mischt die Liste in eine kombinierte Liste von Matchkandidaten für jedes Schemaelement. Dieser ist ausgebildet für Schematas, für die gilt, das ein vom Benutzer geliefertes Mapping bekannt ist, dadurch ist es lernfähig und kann viele Bedeutungen zu jedem Komponentmatcher geben. Neue Komponentmatcher können hinzugefügt werden um die globale Matcherrichtigkeit zu verfeinern. Der Ansatz ist primär instanzorientiert und kann Schemainformation gewinnen. Ein Lernender kann selbst beschreibende Eingaben benutzen, wie XML und macht daraus eine Matchingentscheidung durch die Konzentration auf die Schematags, die die Dateninstanz Werte ignorieren. LSD hat auch Erweiterungen, bei denen die vom Benutzer vorgegebenen Gebiete einschränkend auf das globale Schema beachtet werden. Dabei werden einige früher erkannte Matchkandidaten eliminiert, um die Richtigkeit des Matches zu verbessern [DDH01].

7.4 SKAT - Stanford University

Der SKAT (Semantisches Wissensartikulations Tool) Prototyp folgt einem regelbasierten Ansatz, um semi-automatisierte Matcher zwischen zwei Ontologien [MWJ99] zu bestimmen. Regeln sind logisch so formuliert, das sie Match- und Fehleranpassungsbeziehungen ausdrücken. Dafür werden Methoden so definiert, das sie neue Matches ableiten. Diese Beschreibung behandelt Namensmatcher und einfache strukturelle Matcher, die auf der is-a Hierarchie basieren, es wird aber angegeben welche Details implementiert sind. SKAT wird ohne die ONION Architektur für ontologische Integration [MWK00] benutzt. ONION bedeutet, es werden Ontologien in ein graphbasiertes objektorientiertes Datenbankmodell umgewandelt. Matchingregeln zwischen Ontologien werden benutzt, um eine „Aussprache-Ontologie“ aufzubauen, welche die „Kreuzung“ von Quellontologien erfasst. Matching basiert stark auf einer is-a Beziehung zwischen den Aussprache-Ontologien und Quellontologien. Die Aussprache-Ontologie wird für Anfragen und zum Hinzufügen von Quellen benutzen.

7.5 TransScm - Tel Aviv Univ.

Der TransScm Prototyp [MZ98] benutzt Schemamatching, um eine automatische Datenübersetzung zwischen Schemainstanzen abzuleiten. Eingabeschemata werden in Labelgraphen umgeformt, die interne Schemarepräsentation. Kanten in diesem Schemagraph repräsentieren Einzelteile der Beziehungen. Alle anderen Schemainformationen (Name, Wahlmöglichkeiten, Anzahl der Kinder, etc.) werden als Eigenschaften von Kanten repräsentiert. Das Matching wandelt Knoten für Knoten um, 1:1, beginnend mit der Spitze, Dies setzt einen hohen Grad an Ähnlichkeit zwischen den Schematas voraus. Man nutzt verschiedene Matcher, diese werden in einer festen Reihenfolge abgetestet. Jeder Matcher ist eine „Regel“, die in Java implementiert wird. Sie erfordern das der Abgleich durch exakt einen Matcher pro Knotenpaar bestimmt ist. Wenn kein Abgleich gefunden wurde oder wenn ein Matcher mehrere Matchkandidaten hat, dann ist ein Eingriff vom Benutzer erforderlich, z.B. um eine neue Regel zur Verfügung zu stellen oder einen Matchkandidaten auszuwählen. Die Matcher berücksichtigen typischerweise mehrere Kriterien und können auf diese Weise Hybridansätze darstellen. Ein Beispiel dafür ist, das ein Matcher die Eigenschaften von Namen und die Anzahl der Kinder testet. Knotenmatching kann abhängig von einem partiellen oder vollen Abgleich von dem Knoten-Nachfolger sein.

7.6 ARTEMIS - Univ. of Milano & MOMIS - Univ. of Modena

ARTEMIS ist ein Schemaintegrations-Werkzeug [CDD01]. Es berechnet als erstes die „Verwandschaften“ zwischen den Attributen im Bereich von 0 und 1, dies ist ein matchähnlicher Schritt. Anschließend stellt das Tool die Schemaintegration von Clusteringattributen, welche auf diesen Verwandschaften basieren, fertig. Anschließend werden die Sichten, basierend auf den Clustern erstellt. Der Algorithmus wird auf einen hybriden relationalen OO-Modell betrieben, dieses beinhaltet den Namen, Datentypen und Kardinalitäten von Attributen und Zielobjekttypen, die auf andere Objekte zurückgreifen. Es berechnet Matches durch eine gewichtete Summe von Namen, Datentyp- und strukturale Verwandschaft. Namensverwandschaft basiert auf generische und hauptspezifische Thesauri, dies ist jede Zusammenarbeit von zwei Namen durch Synonyme, Hypernyme oder normale Beziehungen, mit einer festen Verwandschaft für jeden Assoziationstyp. Datentypverwandschaft basiert auf generischen Tabellen von Datentypen Verträglichkeiten. Strukturale

Verwandschaft von zwei Objekten basiert auf der Ähnlichkeit von Beziehungen, die von diesen Objekten ausgehen. ARTEMIS benutzt ein Bestandteil von heterogenen Datenbank Vermittlern, MOMIS genannt (Mediator envirOment for Multiple Information Sources) [BCV99], [BCVB01]. MOMIS integriert unabhängig entwickelte Schemata in ein virtuelles globales Schema auf Basis von objektorientierten Datenmodell, welches ein relationales, objektorientiertes und semistrukturiertes Quellschema benutzt. MOMIS verläßt sich auf ARTEMIS, dem lexikalischen WordNet System, und dem beschreibenden logisch basierten Schlußfolgerungswerkzeug ODB-Tool, um ein integriertes virtuelles Schema zu erstellen. Es bietet einen Anfrageprozessor (mit Optimierung) für Anfragen an die heterogene Datenquelle an.

7.7 Cupid - Microsoft Forschung

Cupid ist ein Hybridmatcher und basiert auf Element- als auch Strukturelevel Matching [MBR01]. Es ist generisch durch Datenmodelle bestimmt und wird angewandt auf XML und relationalen Beispielen. Es benutzt Zusatzinformationen als Quelle für Synonyme, Abkürzungen und Akronyme. Wie bei DIKE beinhaltet jeder Eintrag in diesen Zusatzquellen einen Plausibilitätsfaktor vom Bereich [0,1]. Anders als DIKE kann Cupid Einträge aus normalen Wörtern (z.B. Rechnung ist ein Synonym für Faktura) gewinnen, ohne das eine exakte Matchmischung von Elementnamen erforderlich ist (z.B: RechnungFür oder $Faktura\_Adresse$). Dieser Algorithmus besteht aus 3 Phasen. Die erste Phase stellt linguistische Elementlevel Matching fest und katalogisiert Elemente basierend auf Namen, Datentypen und Gebiete (erstellen von Cupid Hybride). Es analysiert Verbindungselemente wie Namen in Zeichen, die auf Begrenzungssymbole basieren(z.B. $Produkt\_ID$ wird zu {Produkt, Id}). Es klassifiziert diese basierend auf ihre Datentypen und linguistischen Inhalt. Dann errechnet es einen linguistisch ähnlichen Faktor zwischen Datentyp- und linguistischen Inhalt kompatiblen Paaren von Namen, basiert auf Substringmatching und Zusatzquellen. Die zweite Phase transformiert das Orginalschema in einen Baum und dann in ein bottom-up Strukturmatching, Ergebnisse in einer strukturalen Ähnlichkeit zwischen den Elementpaaren. Diese Transformation verschlüsselt referentielle Beschränkungen in Strukturen die dann aufeinander abgestimmt sind, wie in anderen Strukturen (erstellen von Cupid Beschränkungsbasierten). Die Ähnlichkeit von zwei Elementen zur Rootstruktur basiert auf ihre linguistische Ähnlichkeit und der Ähnlichkeit von ihren Blattsätzen. Wenn die Ähnlichkeit eine Grenze übersteigt, dann wird ihre Ähnlichkeit zwischen Blattpaaren verstärkt. Diese Konzentration auf Blattpaare basiert auf die Hypothese, das der Inhalt der Information in Blättern repräsentiert wird und diese Blätter haben weniger Variationen zwischen Schematas als interne Strukturen. Phase zwei ist abgeschlossen durch Errechnung eines gewichteten Mittelwertes von linguistischen und strukturellen Ähnlichkeiten von Elementpaaren. Die dritte Phase benutzt diese gewichteten Mittelwerte um ein Mapping auszuwählen. Diese Phase wird als anwendungsabahängig betrachtet und wird nicht im Algorithmus betont. Es gibt Experimente bei denen Cupid zu DIKE und MOMIS in verschiedenen Schemabeispielen gegenübergestellt wurden.

7.8 Ähnlichkeitsflooding - Stanford Univ. und Univ. Leipzig

In [MGR02] stellt Melnik einen graphischen Matching Algorithmus, den man Ähnlichkeitsflooding (SF) nennt, vor und erforscht seine Brauchbarkeit für Schemamatching. Der Ansatz konvertiert Schematas in direkt beschriftete Graphen und benutzt Fixpunktberechnung um die Matcher zwischen korrespondierenden Knoten von Graphen zu bestim-

men. Er produziert einen 1:1 lokalen, m:n globalen Mapping zwischen Schemaelementen. Der SF Algorithmus ist implementiert, als ein Bediener in einem Prototyp von generischen Schemamanipulationswerkzeuge. Im Zusatz von strukturalen SF Matchern unterstützt das Werkzeug Bediener als einen Namensmatcher, Schemakonvertierer und Filter die mit Skripten kombiniert sein können. Ein typisches Matchskript beginnt mit der Konvertierung der zwei Eingabeschematas in die interne Graphenpräsentation. Danach benutzt der Namensmatcher das eingehende Anfangs-Elementlevel-Mapping, welches in das strukturale SF Matcher eingegeben wird. Im letzten Schritt werden verschiedene Filter auf das Selektieren relevanter Subsets von Matchergebnissen angewandt, diese werden durch den strukturalen Matcher erstellt. Das Werkzeug akzeptiert verschiedene Eingabeformate, wie SQL DDL, XML und RDF.

7.9 Delta - MITRE

Delta repräsentiert einen einfachen Ansatz für bestimmte Attributübereinstimmungen von benutzter Attributbeschreibung [BHFW95]. Alle verfügbaren Matadaten über ein Attribut (z.B. Textbeschreibung, Attributname und Typinformation) sind gruppiert und konvertiert in einen einfachen Textstring, welcher ein Dokument von einem Volltext-Informatiosgewinnungstool erhält. Das IR Tool wird interpretiert, als ein Dokument von einer Anfrage. Dokumente von anderen Schemas mit Matchingattributen sind bestimmt und geordnet. Die Auswahl von Matchern von der Ergebnisliste ist vom Benutzer erlaubt. Der Ansatz ist einfach zu implementieren, aber er hängt von der Verfügbarkeit und den Ausdrücken von Textbeschreibungen für Attribute ab. Clifton vergleicht die experimentellen Matchergebnisse, die durch Delta und dem SemInt Tool ermittelt wurden und schlägt vor die zwei Ansätze zu kombinieren, daraus würde ein Verbundmatcher resultieren.

7.10 Tess - Univ. of Massachusetts

Tess ist ein System, um Schemaevolution abzudecken[Le00]. Tess nimmt eine Definition von alten und neuen Typen und produziert ein Programm zur Transformierung der Daten. Diese sind konform zum alten und zu einem neuen Datentyp. Um dies zu schaffen, muss ein Mapping vom alten zum neuen Typ automatisch erzeugt werden, dabei wird ein Schemalevel Matchingalgorithmus benutzt. Wie TransScm setzt es einen hohen Ähnlichkeitsgrad zwischen Schemata voraus. Es identifiziert Typpaare als Matchkandidaten und gleicht den rekursiven Trie in ihrer Substruktur in einem top-down Gestalt an. Zwei Elemente sind Matchkandidaten, wenn sie den gleichen Namen haben, Paare von Subelemente dieses Matches sind (z.B. sind sie vom selben Typ) oder den selben Typkonstruktor benutzen (in Reihenfolgen der Vorzügen, wo meistens der Namensmatching bevorzugt ist). Die Rekonstruktion wird mit skalaren Subelementen erreicht. Tess durchläuft beides, Strukturlevel- und Elementlevel-Matching.

7.11 COMA - System for combining match algorithms

7.11.1 Allgemeines zu COMA

Dieses generische Schemamatching System wurde in der Universität Leipzig entwickelt [RH02]. Es ist eine Plattform, die eine erweiterbare Bibliothek von Matchalgorithmen zur Verfügung stellt und auf der mehrere Matcher auf einem flexiblen Weg kombiniert wurden. Man verfügt über ein großes Spektrum an verschiedenen Matchern, dabei zielt ein neuartiger Ansatz auf das Wieder benutzen von Ergebnissen aus früheren Matchoperationen

und getrennten Mechanismen zum Kombinieren von Ergebnissen von Matcherausführungen. Neue Matchalgorithmen können in die Bibliothek integriert werden und anschließend mit anderen Matchern kombiniert, benutzt werden. Man benutzt COMA als eine Auswertungsplattform von systemunabhängigen Untersuchungen und Vergleichen von verschiedenen Matchern und Kombinationsstrategien. In dem Design von COMA beachtet man, das allgemeine voll-automatische Lösungen von Matchproblemen nicht möglich sind aufgrund des hohen Grades der semantischen Heterogenität zwischen Schematas. Man erlaubt deshalb einen interaktiven und iterativen Matchprozess, welcher dem Nutzer eine Rückmeldung gibt, z.B. durch zur Verfügung stellen von Matchkorespondenzen, bestätigen oder zurückweisen verdächtiger Matches. Die Flexibilität von COMA ist, das man DBMS-basierte Speicher für Ablageschemata benutzen kann und komplexe Matchergebnisse für eine spätere Benutzung. Man stellt Schematas durch gerichtete azyklische Graphen dar. In Figur 12 sind solche Graphbeispiele dargestellt. Schemaelemente werden

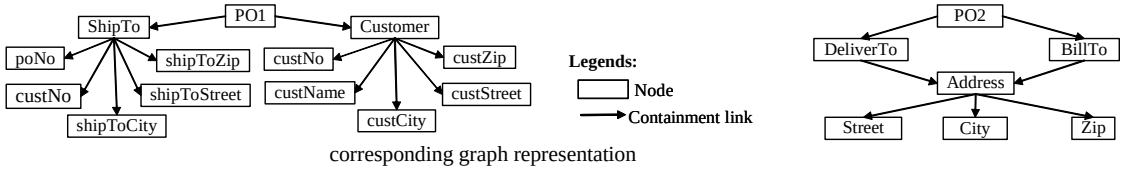

Fig. 12: externe und interne Schemarepräsentation

durch Graphknoten repräsentiert, diese sind durch direkte Verknüpfung von verschiedenen Typen, z.B. für erhaltenen und referentielle Beziehungen, miteinander verbunden. Figur

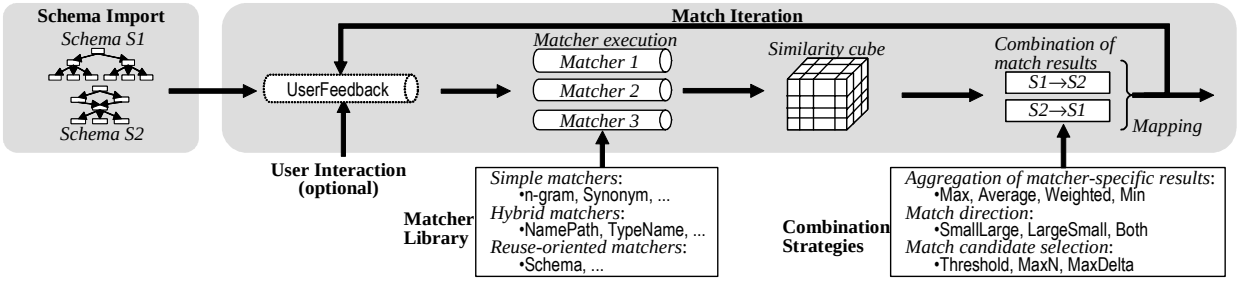

Fig. 13: Matchprozessing in COMA

13 zeigt einen Matchprozess von zwei Eingabeschematas S1 und S2 in COMA. Matchprozesse nehmen entweder Platz in einem oder mehreren Iterationen ein, abhängig davon ob eine automatische oder interaktive Festsetzung von Matchkandidaten ausgeführt wird. Jede Matchiteration besteht aus drei Phasen: eine optionale Benuzterrückmeldungsphase, die Abarbeitung von verschiedenen Matchern und die Kombination von individuellen Matchergebnissen. Im interaktiven Modus kann der Benutzer interaktiv mit COMA für jede Iteration die Matchstrategie spezifizieren (Auswahl von Matchern, Strategien von kombinierten, individuellen Matchergebnisse), Match oder Mismatch Beziehungen definieren und akzeptieren oder Matchkandidaten vorgeschlagener vergangener Iterationen zu unterdrücken. Der interaktive Ansatz ist nützlich zum Testen und zum Vergleichen verschiedener Matchstrategien für spezifische Schematas und Verbessern der Matchergebnisse. Im automatischen Modus besteht der Matchprozess aus einer Singlematchiteration, für diese wird eine Defaultstrategie angewendet oder die Strategie ist spezifiziert durch Eingabeparameter. Dieser Modus ist besonders nützlich für Anwendungen, die bereits

ihre geeigneten Matchstrategien wissen oder ihr eigenes Benutzerinteraktions-Interface implementieren. Ein Hauptschritt während einer Matchiteration ist die Abarbeitung von mehreren unabhängigen Matchern, die aus der Matcherbibliothek ausgewählt wurden. Jeder Match bestimmt eine obere Mittelklasse der Matchergebnisse, bestehend aus einem ähnlichen Wert zwischen 0 und 1 für jede Kombination von S1 und S2 der Schemaelemente. Das Ergebnis der Matchabarbeitungsphase mit k Matchern, m S1 Elemente und n S2 Elemente führt in einem k x m x n Würfel der ähnlichen Werte, welche in einer Quelle für spätere Kombinationen oder Auswahlschritte abgelegt sind. Figur 14 zeigt ein Beispiel für den Ähnlichkeitswürfel für das erworbene Ordnungschemata von Figur 13. Der End-

Matcher	*PO1* Elements	*PO2* Elements	Sim
Type-Name	*PO1.ShipTo.shipToCity*	*PO2.DeliverTo.Address.City*	0.65
	PO1.ShipTo.shipToStreet		0.3
	PO1.Customer.custCity		0.80
Name-Path	*PO1.ShipTo.shipToCity*	*PO2.DeliverTo.Address.City*	0.78
	PO1.ShipTo.shipToStreet		0.73
	PO1.Customer.custCity		0.53

Fig. 14: errechnete Ähnlichkeitswerte für PO1 und PO2

schritt in einer Matchiteration ist das Ableiten der kombinierten Matchergebnisse von den individuellen Matchergebnissen, welche in einem Ähnlichkeitswürfel abgelegt sind. Dieser Endschritt wird nochmal in zwei Subschritte unterteilt: Sammlung von matchspezifischen Ergebnissen und Auswahl der Matchkandidaten.

7.11.2 Matchbibliothek

Figur 15 gibt uns eine Übersicht über die implementierten und getesteten Matchern unter COMA. Es charakterisiert die Schemasorten und die Zusatzinformationen.

Matcher Type	Matcher	Schema Info	Auxiliary Info
Simple	*Affix*	Element names	-
	n-gram	Element names	-
	Soundex	Element names	-
	EditDistance	Element names	-
	Synonym	Element names	Extern. dictionaries
	DataType	Data types	Data type compatibility table
	UserFeedback	-	User-specified (mis-) matches
Hybrid	*Name*	Element names	-
	NamePath	Names+Paths	-
	TypeName	Data types+Names	-
	Children	Child elements	-
	Leaves	Leaf elements	-
Reuse-oriented	*Schema*	-	Existing schema-level match results

Fig. 15: Implementierte Matcher in der Matchbibliothek

7.11.3 Wiederverwendbarkeit von vorhandenen Matches

Für eine Wiederverwendung benutzt man die Match-Compose-Operation. Bei der Match-Compose-Operation nimmt man zwei gegebene Matchergebnisse, match1: S1 $\leftrightarrow$ S2 und match2: S2 $\leftrightarrow$ S3 gemeinsam genutztes Schema S2, Matchcompose leitet daraus ein neues Matchergebnis ab, match: S1 $\leftrightarrow$ S3 zwischen S1 und S3. Die Operation ist transitiver Natur der Ähnlichkeitsrelation zwischen den Elementen, z.B. wenn a ähnlich zu b und b ähnlich zu c sei, dann ist auch a ähnlich zu c. Ein Ansatz, bei dem die transitive Ähnlichkeit

bestimmt wird, ist das vermehren von individuellen Ähnlichkeitswerten. Dieser Ansatz zersetzt schnell den Ähnlichkeitswert, z.B. contactFirstName $\leftrightarrow^{0.5}$ Name $\leftrightarrow^{0.7}$ firstName stellt die Ähnlichkeit zwischen contactFirstName und firstName mit 0.5 * 0.7 = 0.35 dar. Figur 16 zeigt ein Beispiel für den Match PO1 ↔ PO3 abgeleitet von den zwei

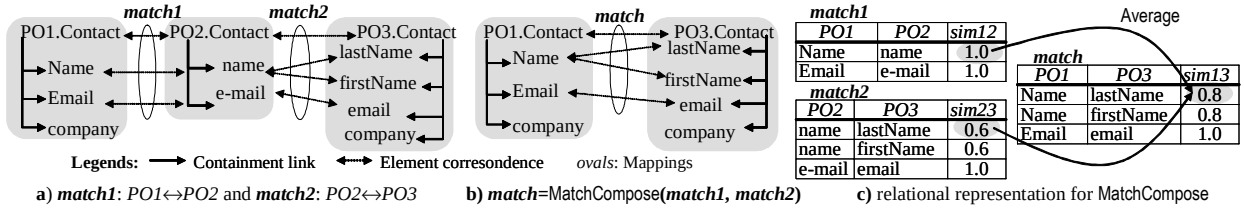

Fig. 16: MatchCompose Beispiel

Matchergebnisse match1: PO1 ↔ PO2 und match2: PO2 ↔ PO3. Die Abbildung 16c stellt die repräsentierenden Tabellen match1, match2 und match dar. In diesen Tabellen ist jedes Tupel in der 1:1 Notation und korrespondiert zwischen Elementen vom Erfüllten Schemata mit deren Ähnlichkeit. MatchCompose ist der natürliche Join zwischen zwei Eingabetabellen.

7.11.4 Kombination von Ähnlichkeitswerten

Um Ähnlichkeitswerte zu kombinieren, nutzt man zwei Fälle: innerhalb der Implementation von Hybridmatchern kombiniert man die Ergebnisse von Komponentenmatcher und im letzten Schritt kombiniert man die Ergebnisse von unabhängigen Matchern, um ein komplettes Matchergebnis zu erhalten. Beide Fällen sind durch eine Reihe von Aggregations- und Selektions-Operationen im Ähnlichkeitswürfel gekennzeichnet, dieser enthält die Ähnlichkeitswerte, die durch einen Satz von Matchern M errechnet wurden. Dies wird in Figur 17 dargestellt. Um ein komplettes Matchergebnis für zwei Eingabeschemata zu erhalten,

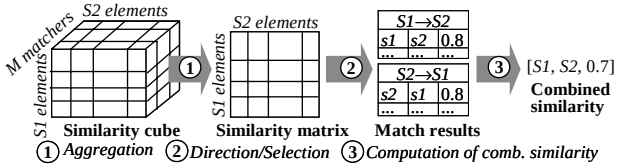

Fig. 17: Kombination von Matchergebnissen

braucht man 3 Schritte, dabei ist Schritt 3 optional.

- Aggregation von matchspezifischen Ergebnissen: Im ersten Subschritt, werden durch mehrere Matcher Ähnlichkeitswerte errechnet und insgesamt als ein kombinierter Ähnlichkeitswert für jedes Paar von Schemaelementen betrachtet. Mit m S1 Elementen und n S2 Elementen erhält man eine m x n Matrix von kombinierten Ähnlichkeitswerten.

- Auswahl von Matchkandidaten: zum Bestimmen der besten Matchkandidaten. Man stuft die Übereinstimmung gemäß ihrer Ähnlichkeitswerte durch Schemaelemente ein und wendet eine Filterstrategie zum Bestimmen der Besten an. Das Ergebnis von diesem Schritt ist ein kombiniertes Matchergebnis mit keinem, einem oder mehreren Matchkandidaten durch Schemaelemente. In dem Fall eines ungerichteten Match wird der Matchkandidat für beide Schemata ermittelt.

- Berechnung von kombinierter Ähnlichkeit: das Matchergebnis vom vorhergehenden Schritt kann sich für die zwei Schematas belaufen auf einen einzelnen Ähnlichkeitswert, dies nennt man Schemaähnlichkeit. Es ist abhängig von den gewählten Matchern und ihren kombinierten Strategien.

Aggregation von matchspezifischen Ergebnissen Man benutzt eine der folgenden Strategien für jedes Elementpaar:

- Max: gibt die maximalen Ähnlichkeitswerte von jedem Matcher zurück. Dies nutzt man in Fällen, wenn sich Ähnlichkeitswerten widersprechen. Matcher können sich gegenseitig ergänzen und es ist eine optimistische Strategie.
- Gewichtet: bestimmt eine gewichtete Summe von Ähnlichkeitswerten von den verschiedenen Matchern und braucht relative Gewichtungen, diese sollten den gewonnenen Gewichtungen der Matcher entsprechen.
- Durchschnitt: ist ein Spezialfall von der gewichteten Strategie und gibt die durchschnittliche Ähnlichkeit über allen Matchern zurück, z.B. Berücksichtigung der gleichen Wichtigkeiten.
- Min: wählt die niedrigsten Ähnlichkeitswerte von beliebigen Matchern aus. Es ist das Gegenstück zu der Max-Strategie, es ist pessimistisch.

Auswahl der Matchkandidaten Für zwei gegebene Schematas S1 und S2, wobei gilt $|S2| \leq |S1|$.

- LargeSmall: man gleicht das größere Schemata S1 gegen das kleinere Zielobjekt S2 ab, z.B. Elemente von S1 sind abgestuft und stehen mit jedem S2 Element in Beziehung.
- SmallLarge: ist das Gegenstück zu LargeSmall.
- Both: berücksichtigt die Ergebnisse von beiden Matchrichtungen LargeSmall und SmallLarge. Ein S1 und ein S2 Element wird nur akzeptiert als ein Matchingpaar, wenn es in beiden Richtungen nachgewiesen wurde.

Berechnung von kombinierter Ähnlichkeit Hierbei nutzt man zwei Strategien, die mit der Ausgabe vom zweitem Schritt arbeiten.

- Mittelwert: die mittlere Ähnlichkeit ist bestimmt durch dividieren der Summe von den Ähnlichkeitswerten von allen Matchkandidaten beider Sätze S1 und S2 durch die gesamte Anzahl der Elemente, $|S1| + |S2|$.
- Würfel: basierend auf den Würfelkoeffizienten [CDF98] und das zurückgeben des Verhältnisses der Anzahl der Elemente, die über die gesamte Anzahl der Satzelemente ab geglichen wurden.

7.12 COMA++

7.12.1 Allgemeines

COMA++ ist ein Schema und Ontologie Matchingwerkzeuge, das den Prototyp COMA erweitert [BVL03]. Es implementiert wichtige Verbesserungen und bietet eine reichhaltige Infrastruktur zum Lösen von real-world Matchproblemen an. Ein graphisches Interface erlaubt eine Anzahl von Benutzerinteraktionen. Das generische Datenmodell unterstützt einheitlich Schematas und Ontologien, die in verschiedenen Sprachen geschrieben sind, z.B. SQL, W3C XSD und OWL Standards. Durch den neuen Ansatz für Ontologien kann man teilweise Scharen von Taxonomien auswerten. In Figur 18 wird das graphische User Interface dargestellt. Die GUI stellt die 5 Hauptteile von COMA++ zur Verfügung, das Repository zum Speicher aller match-relevanten Daten, die Modell und Mappingpools zum Verwalten von Schematas, Ontologien und Mappings im Speicher, der Match Customizer zum Konfigurieren von Matchern und Matchstrategien und die Execution Engine zum Ausführen von Matchoperationen.

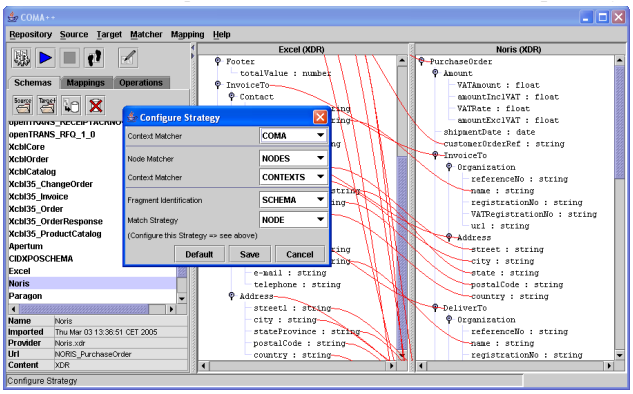

Fig. 18: User Interface für COMA++

Figur 19 zeigt die tiefer liegende Struktur von COMA++.

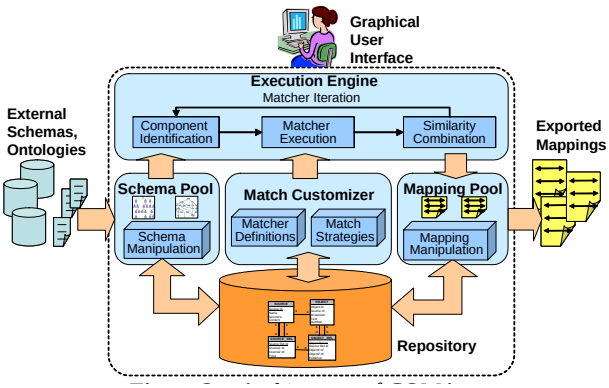

Fig. 19: Architektur von COMA++

7.12.2 Ontologie Unterstützung

COMA++ unterstützt gegenwärtig Matching zwischen Ontologien, die in W3C geschrieben sind. Die OWL Klassen-Hierarchie und Beziehungstypen werden via OWL API gelesen

und zur generischen Modell-Repräsentation in direkten azyklischen Graphen aufgenommen. OWL/RDF Knoten werden durch URI's identifiziert, diese werden als Knotennamen in COMA++ aufgenommen, dabei benutzt man die RDF-Label wenn sie verfügbar sind, ansonsten das letzte URI Fragment. Nachdem man OWL Ontologien importiert hat, kann man sie als Schema sichtbar machen und alle verfügbaren Matcher eintragen, in teilweise verschiedenen Namens- und strukturell Matcher. Der Taxonomiematcher passt zwei Schemata oder Ontologien mit Hilfe von Taxonomien an.

7.13 ETuner

ETuner ist ein automatisch einstellbares Schemamatching System [SLD05], welches sequentiell arbeitet. Man nutzt eTuner zum Abstimmen für vier vor kurzem entwickelte Matching Systeme im real-world Gebiet. Figur 20 beschreibt die Architektur, dabei be-

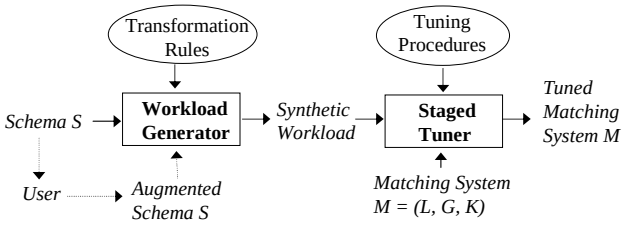

Fig. 20: Architektur von eTuner

steht das Tupel M = (L, G, K) aus folgenden Komponenten. L ist eine Bibliothek von Matchingkomponenten, G ist ein direkter Graph der den Fluß der Abarbeitung der Komponenten von M spezifiziert. Und K steht für eine Sammlung von Kontrollvariablen, die vom Nutzer oder vom System, z.B. eTuner gesetzt sein können. Der eTuner Ansatz besteht aus 2 Modulen, den Workload Generator und dem Staged Tuner. In einem gegebenen Schema S legt der Workload Generator einen Satz von Transaktionsregeln zum Generieren eines synthetischen Workloads an. Der Staged Tuner stimmt ein Matchingsystem unter Benutzung des synthetischen Workload und von tuning Prozeduren ein, anschließend wird es in einem eTuner Reposority gespeichert. Das abgestimmte System M kann nun auf das Matchschema S mit einigen Folgeschemata angewendet werden.

8 Beispiel aus der Wirtschaft

8.1 Cognidata

In einem Projektbericht vom März 2004, der Firma Cognidata [CSAP04] geht hervor, dass sie mit der Entwicklung eines prototypischen Matchingtools, das Daten- bzw. Nachrichtenstrukturen halbautomatisch miteinander vergleicht und passende Informationen miteinander verknüpft, von der Firma SAP beauftragt wurde. Auf Basis der Algorithmensammlung von COMA wurde ein lauffähiger Prototyp entwickelt.Nach der ersten Testphase wurde dieser Prototyp als eine Teilkomponente für SAP Exchange Infrastructure spezifiziert. Leider liegen keine weiteren Daten vor, ob dieses Projekt abgeschlossen wurde.

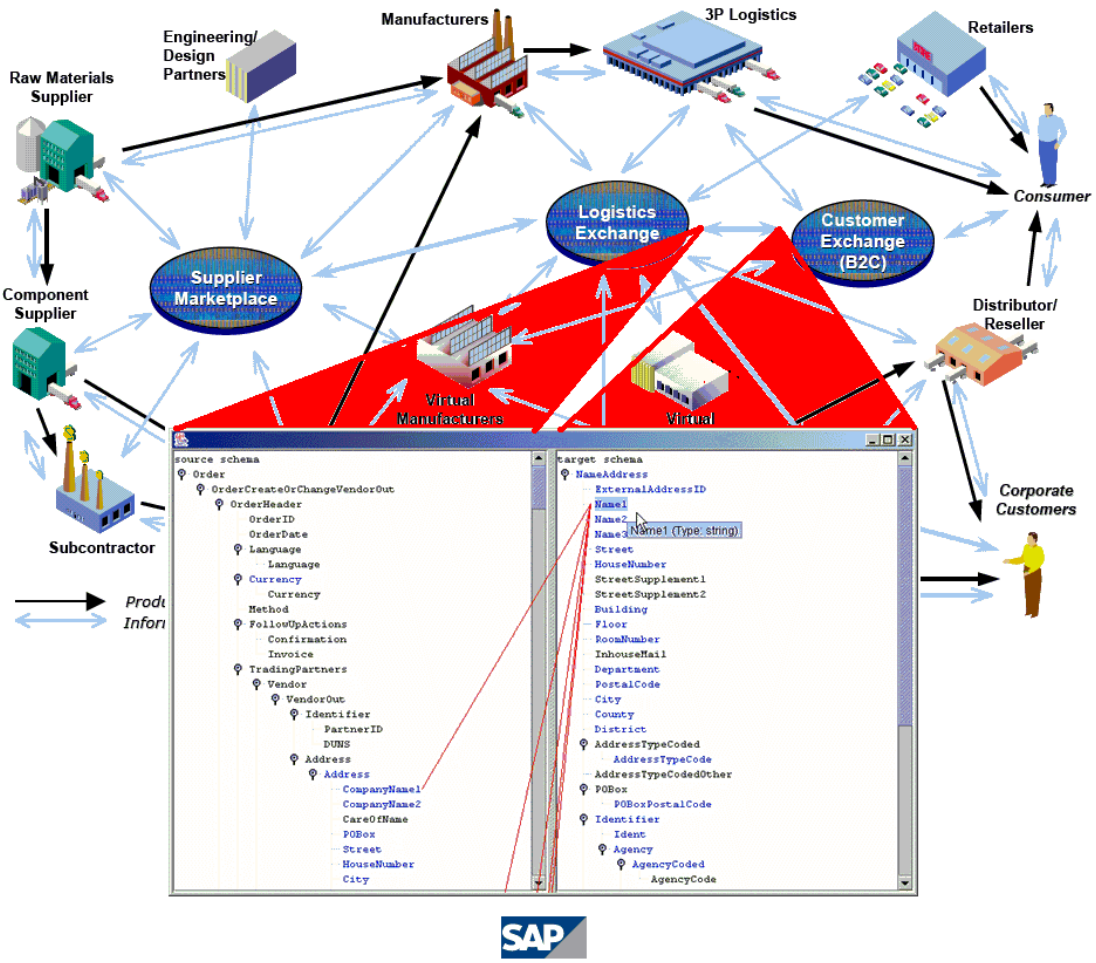

Fig. 21: semi-automatisches Matchingtool

9 Ausblick

Schemamatching ist ein Basisproblem in vielen Datenbank Anwendungsgebieten, wie E-Commerce, Data-Warehouse und semantische Anfrageprozesse. In diesem Dokument, werden einige existierende Ansätze genauer beschrieben. Teilweise diskutiert man zwischen schema- und instancelevel, element- und strukturlevel und sprach- und constraintbasierte Matcher und man diskutiert die Kombination von mehreren Matchern. Man benutzt die Taxonomie zur Charakterisierung und gegenüberstellen einer Art von vorläufigen Matchimplementationen. Man hofft das die Taxonomie nützlich für Programmierer sein wird, welche gebraucht werden um Matchalgorithmen zu implementieren und an effektiveren Schemamatching-Algorithmen zu forschen. Beispielsweise sollte man mehr Achtung auf die Auswertung von instanzlevel Informationen und bei der Wiederbenutzung von Matchs. Frühere Arbeiten an Schemamatching standen oft im Kontext von Anwendungsgebieten. Dieses Problem ist fundamental, aber man arbeitet bereits daran dieses Problem zu lösen. In der Zukunft ist vorgesehen, dass man an akkuraten und quantitativen Ansätzen arbeitet. Solche Ergebnisse könnten uns mitteilen, welche von den existierenden Ansätzen die anderen dominieren. Und könnte uns helfen die Schwächen in den existierenden Ansätzen zu identifizieren, dies deutet Möglichkeiten für die zukünftige Forschung an.

Literatur

[BCV99] Bergamaschi S, Castano S, Vincini M (1999) Semantic integration of semistructured and struktured data sources. ACM SIGMOD Recodt 28(1):54-59

[BCVB01] Bergamaschi S, Castano S, Vincini M, Beneventano D (2001) Semantic integration of hererogeneous information sources. Data Knowl Eng 36(3):215 - 249

[BHFW95] Benkley S, Fandozzi J, Housman E, Woodhouse G (1995) Data element tool-based analysis (DELTA). MITRE Technical Report MTR'95 B147

[BM01] Berlin J, Motro M (2001) Autoplex: automated discovery of content for virtual databases. In: Proc 9th Int Conf On Cooperative Information Systens (CoopIS), Lecture Notes in Computer Science, vol. 2172. Springer, 2001, pp. 108 - 122

[BVL03] Bechhofer S, Volz R, Lord P (2003) Cooking the Semantic Web with the OWL API. International Semantic Web Conference (ISWC)

[BS01] Bell GS, Sethi A (2001) Matching records in a national medical patient index. CACM 44(9): 83 - 88

[CDD01] Castano S, DeAntonellis V, DeCapitani di Ve,ercati S (2001) Global viewing of heterogeneois data sources. IEEE Trans Data Knowl Eng 13(2):277 - 297

[CDF98] Castano S, DeAntonellis V, Fugini MG, Pernici B (1998) Conceptual Schema Analysis: Techniques and Applications. ACM Trans. Database Systems 23(3): 286 - 333

[CHR97] Clifton C, Housman E, Rosenthal A (1997) Experience with a combined approach to attribute-matching across heterogeneous databases. In: Proc 7, IFIP 2.6 Working Conf. Database Semantics

[CSAP04] http://www.cognidata.de (2004)

[DDH01] Doan AH, Domingos P, Halevy A (2001) Reconciling schemas of disparate data sources: a machine-learning approach. In: Proc ACM SIGMOD Conf, pp. 509 - 520

[DDL00] Doan AH, Domingos P, Levy A (2000) Learning source descriptions for data integration. In: Proc WebDB Workshop, pp. 81 - 92

[EJX01] Embley DW, Jackman D, Xu L (2001) Multifaceted exploitation of metadata for attribute match discovery in information integration. In: Proc Int Workshop on Information Integration on the Web, pp. 110 - 117

[GW97] Goldman R, Widom J (1997) Dataguides: enabling query formulation and optimaization in semistructured databases. In: Proc 23th Int Conf On Very Large Data Bases, pp. 436 - 445

[KKFG84] Korth HF, Kuper GM, Feigenbaum J, Van Gelder A, Ullmann JD (1984) System/U: a database system based on the universal relation assumption. ACM TODS 9(3): 331 - 347

[LC00] Li W, Clifton C (2000) SemInt: a tool for identifying attribute correspondences in heterogeneous databases using neural network. Data Knowl Eng 33(1):49 - 84

[LCL00] Li W, Clifton C, Liu S (2000) Database integration using neural network: implementation and experiences. Knowl Inf Syst 2(1): 73 - 96

[Le00] Lerner BS (2000) A model for compound type changes encountered in schema evolution. ACM TODS 25(1):83 - 127

[LNE89] Larson JA, Navathe SB, ElMasri R (1989) A theory of attribute equvalence in databases with application to schema integration.IEEE TransSoftwareEng 16(4):449 - 463

[MBDH05] Madhavan J, Berstein PA, Doan A, Halevy A (2005) Corpus-based Schema Matching. Int. Conf. of Data Engineering (ICDE)

[MBR01] Madhavan J, Bernstein PA, Rahm E (2001) Generic schema matching with Cupid. In: Proc 27th Int Conf On Very Large Databases, pp. 49 - 58

[MGR02] Melnik S, Garcia-Molina H, Rahm E (2002) Similarity flooding - a versatile graph matching algorithm. In: Proc 18th Int Conf Data Eng (to appear)

[MWJ99] Mitra P, Wiederhold G, Jannink J (1999) Semi-automatic integration of knowledge sources. In: Proc of Fusion '99, Sunnyvale, USA

[MWK00] Mitra P, Wiederhold G, Kersten M (2000) A graph-oriented model for articulation of ontology interdependencies. In: Proc Extending DataBase Technologies, Lecture Notes in ComputerScience, vol. 1777. Springer, 2000, pp. 86 - 100

[MZ98] Milo T, Zohar S (1998) Using schema matching to simplify heterogeneous data translation. In: Proc 24th Int Conf On Very Large Data Bases, pp. 122 - 133

[RB01] Rahm E, Bernstein PA (2001) A survey of Approaches to Automatic Schema Matching. VLDB Journal 10(4)

[RH02] Rahm E, Hong-Hai D (2002) COMA - A system for flexible combination of schema matching approaches. In: Proc. of the 28th VLDB Conference, Hong Kong, China

[SLD05] Sayyadian M, Lee Y, Doan AH, Rosenthal AS (2005) Tuning Schema Matching Software using Synthetik Scenarios. In: Proc 31st VLDB Conference, Trondheim, Norway

[WI05] Wikepedia Lexikon 2005 - http://de.wikipedia.org/wiki/Schema_Matching

[WYW00] Wang Q, Yu J, Wong K (2000) Approximate graph schema extraction for semi-structured data. In: Proc Extending DataBase Technologies, Lecture Notes in Computer Science, vol. 1777. Springer, pp. 302 - 3161995) Data element tool-based analysis (DELTA). MITRE Technical Report MTR'95 B147

[WZJS94] Wang JTL, Zhang K, Jeong K, Sasha D (1994) A system for approximate tree matching. IEEE Trans Data Knowl Eng 6(4): 559 - 571

[YMHF01] Yan L, Miller RJ, Haas LM, Fagin R (2001) Data-driven understanding and refinement of schema mappings. In: Proc ACM SIGMOD Conf, pp. 485 - 496