

Schema Matching

Komplexe Informationssysteme

Fabian Panse

panse@informatik.uni-hamburg.de

Universität Hamburg



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG



Schema Matching

Schema Matching – Problemstellung

Schema Matching – Problemstellung

- **Gegeben:** Zwei Datenbankinstanzen mit jeweiligem Schema

Schema Matching – Problemstellung

- **Gegeben:** Zwei Datenbankinstanzen mit jeweiligem Schema
- **Gesucht:** Mapping zwischen den zwei Schemata

Schema Matching – Problemstellung

- **Gegeben:** Zwei Datenbankinstanzen mit jeweiligem Schema
- **Gesucht:** Mapping zwischen den zwei Schemata
- **Vorgehensweise:**

Schema Matching – Problemstellung

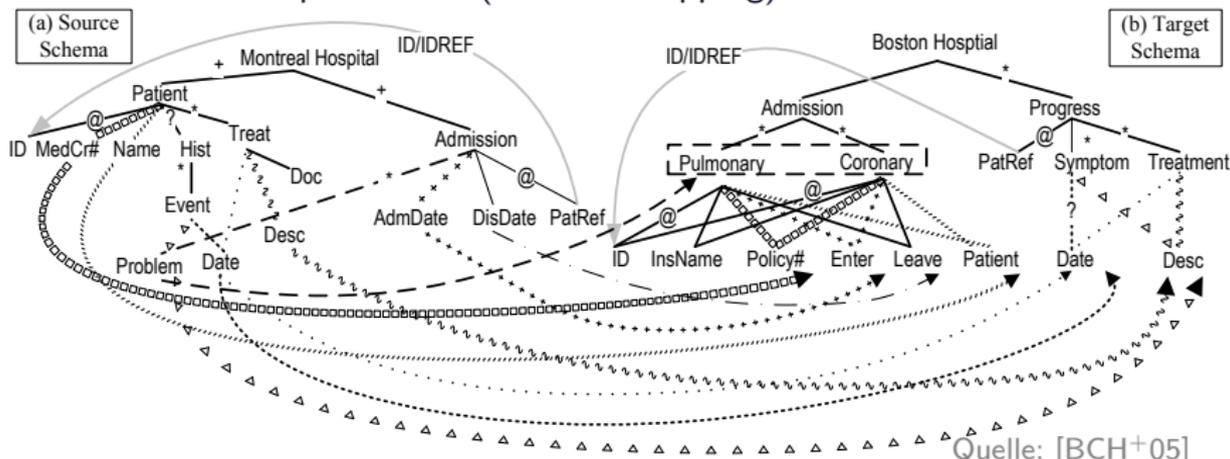
- **Gegeben:** Zwei Datenbankinstanzen mit jeweiligem Schema
- **Gesucht:** Mapping zwischen den zwei Schemata
- **Vorgehensweise:**
 1. Finden von Korrespondenzen zwischen den Elementen beider Schemata (Schema Matching)

Schema Matching – Problemstellung

- **Gegeben:** Zwei Datenbankinstanzen mit jeweiligem Schema
- **Gesucht:** Mapping zwischen den zwei Schemata
- **Vorgehensweise:**
 1. Finden von Korrespondenzen zwischen den Elementen beider Schemata (Schema Matching)
 2. Ableitung eines Mappings von den gefundenen Korrespondenzen (Schema Mapping)

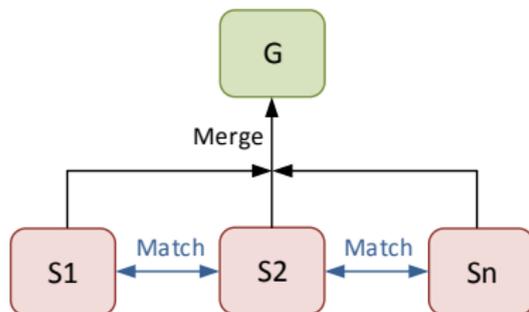
Schema Matching – Problemstellung

- **Gegeben:** Zwei Datenbankinstanzen mit jeweiligem Schema
- **Gesucht:** Mapping zwischen den zwei Schemata
- **Vorgehensweise:**
 1. Finden von Korrespondenzen zwischen den Elementen beider Schemata (Schema Matching)
 2. Ableitung eines Mappings von den gefundenen Korrespondenzen (Schema Mapping)



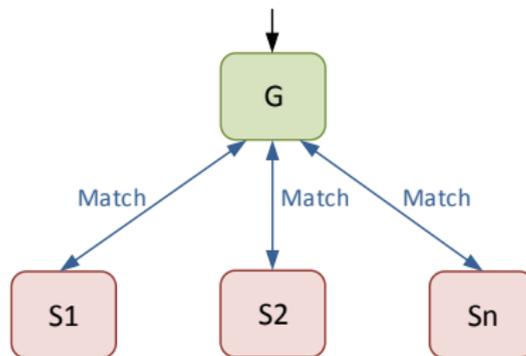
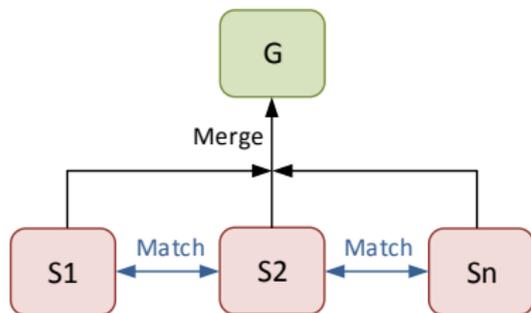
Arten von Schema Matching Szenarien

- **Bottom-Up:**
 - Matching zwischen zwei Quellschemata
 - Globales Schema resultiert aus Integration der Quellschemata
 - Typisch für materialisierte Integration (z.B. Data Warehouse)



Arten von Schema Matching Szenarien

- **Bottom-Up:**
 - Matching zwischen zwei Quellschemata
 - Globales Schema resultiert aus Integration der Quellschemata
 - Typisch für materialisierte Integration (z.B. Data Warehouse)
- **Top-Down:**
 - Matching zwischen globalem und Quellschema
 - Globales Schema ist von außen vorgegeben
 - Typisch für virtuelle Integration



Arten von Schema Korrespondenzen

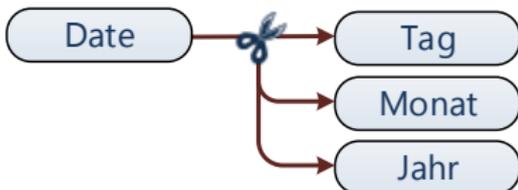
1:1 Korrespondenz



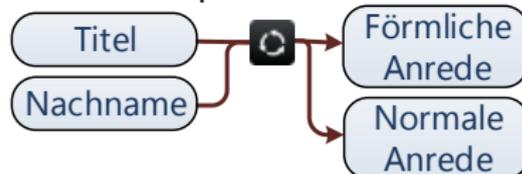
n:1 Korrespondenz



1:n Korrespondenz



n:m Korrespondenz



Quelle: Felix Gessert, Schema Matching - Semiautomatische Matching Verfahren und ihre Algorithmen

Agenda

- 1 Schema Matching
 - Problemstellung
 - **Motivation**

- 2 Schema Matching Systeme
 - Label-based Matcher
 - Instance-based Matcher
 - Structure-based Matcher
 - Combiner
 - Match Selector
 - Constraint Enforcer
 - n:m Korrespondenzen

Schema Matching – Motivation

- Große Schemas
 - > 100 Tabellen, viele Attribute
 - Bildschirm nicht lang genug

Schema Matching – Motivation

- Große Schemas
 - > 100 Tabellen, viele Attribute
 - Bildschirm nicht lang genug
- Unübersichtliche Schemas
 - Tiefe Schachtelungen
 - Fremdschlüssel
 - Bildschirm nicht breit genug
 - XML Schema

Schema Matching – Motivation

- Große Schemas
 - > 100 Tabellen, viele Attribute
 - Bildschirm nicht lang genug
- Unübersichtliche Schemas
 - Tiefe Schachtelungen
 - Fremdschlüssel
 - Bildschirm nicht breit genug
 - XML Schema
- Fremde Schemas
 - Unbekannte Synonyme

Schema Matching – Motivation

- Große Schemas
 - > 100 Tabellen, viele Attribute
 - Bildschirm nicht lang genug
- Unübersichtliche Schemas
 - Tiefe Schachtelungen
 - Fremdschlüssel
 - Bildschirm nicht breit genug
 - XML Schema
- Fremde Schemas
 - Unbekannte Synonyme
- Irreführende Schemas
 - Unbekannte Homonyme

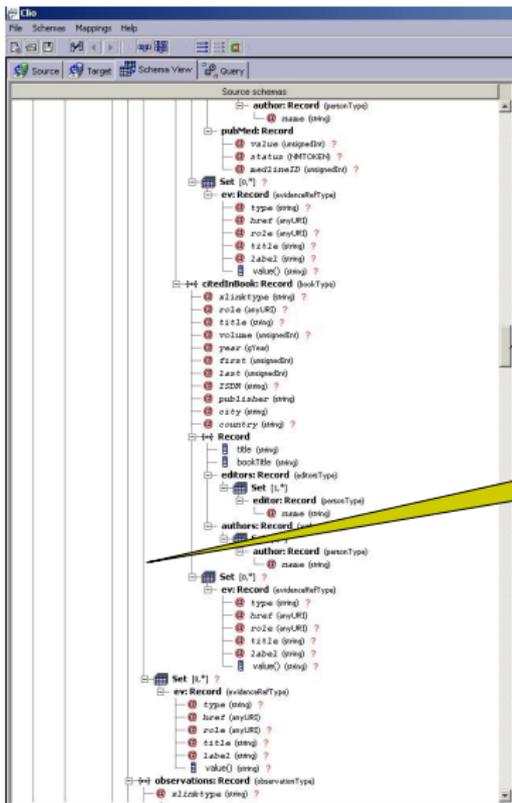
Schema Matching – Motivation

- Große Schemas
 - > 100 Tabellen, viele Attribute
 - Bildschirm nicht lang genug
- Unübersichtliche Schemas
 - Tiefe Schachtelungen
 - Fremdschlüssel
 - Bildschirm nicht breit genug
 - XML Schema
- Fremde Schemas
 - Unbekannte Synonyme
- Irreführende Schemas
 - Unbekannte Homonyme
- Fremdsprachliche Schemas

Schema Matching – Motivation

- Große Schemas
 - > 100 Tabellen, viele Attribute
 - Bildschirm nicht lang genug
- Unübersichtliche Schemas
 - Tiefe Schachtelungen
 - Fremdschlüssel
 - Bildschirm nicht breit genug
 - XML Schema
- Fremde Schemas
 - Unbekannte Synonyme
- Irreführende Schemas
 - Unbekannte Homonyme
- Fremdsprachliche Schemas
- Kryptische Schemas
 - Attributnamen < 8 Zeichen
 - Tabellennamen < 8 Zeichen

Komplexe Schemata



Man beachte die Scrollbar!

Man beachte die Schachtelungstiefe!

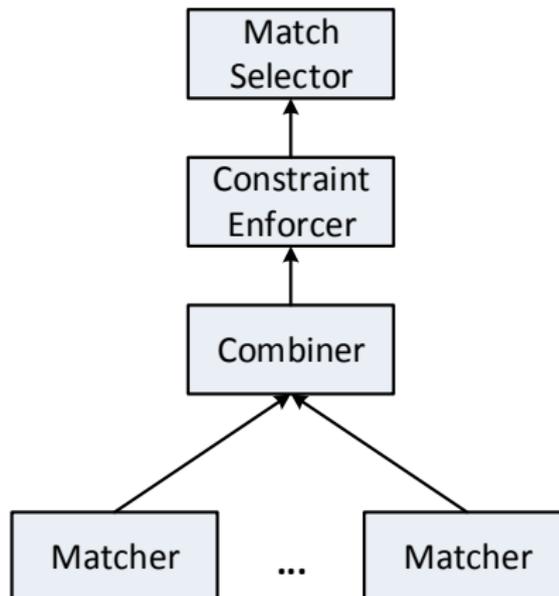
Schema Matching Systeme

Agenda

- 1 Schema Matching
 - Problemstellung
 - Motivation

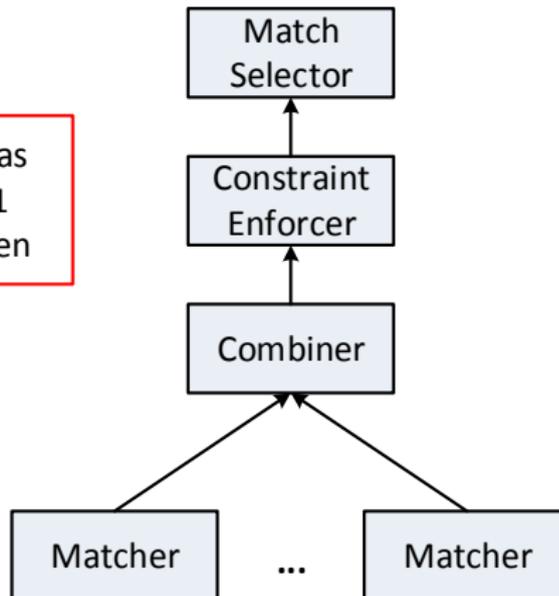
- 2 Schema Matching Systeme
 - Label-based Matcher
 - Instance-based Matcher
 - Structure-based Matcher
 - Combiner
 - Match Selector
 - Constraint Enforcer
 - n:m Korrespondenzen

Aufbau von Schema Matching Systemen

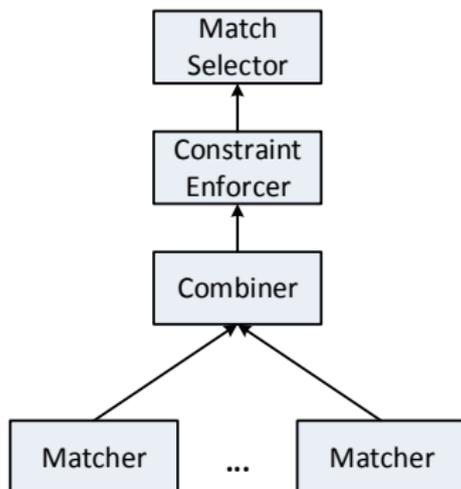


Aufbau von Schema Matching Systemen

Ausgelegt für das
Finden von 1:1
Korrespondenzen



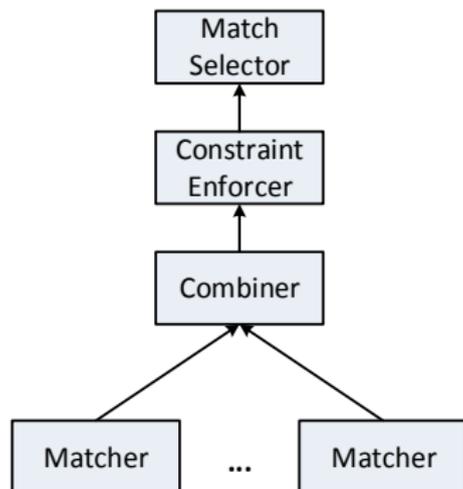
Aufbau von Schema Matching Systemen



Matchers

- **Input:** Schemata \mathcal{A} und \mathcal{B}
- **Output:** Ähnlichkeitsmatrix (eine Ähnlichkeit pro Paar $\{(a, b) \mid a \in \mathcal{A}, b \in \mathcal{B}\}$)
- Verschiedene Arten von Matchern die alle auf anderen Ideen basieren (vergleichbar mit Ähnlichkeitsmaßen für Stringwerte)

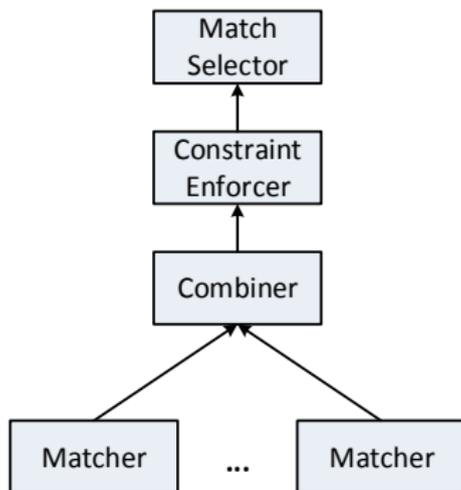
Aufbau von Schema Matching Systemen



Combiner

- **Input:** eine Menge an Ähnlichkeitsmatrizen
- **Output:** eine einzige Ähnlichkeitsmatrix
- Kombiniert die Ergebnisse der einzelnen Matcher zu einem einzelnen
- Breites Spektrum:
 - primitiv (Durchschnittsberechnung)
 - komplex (gelernt oder handgefertigte Skripte)

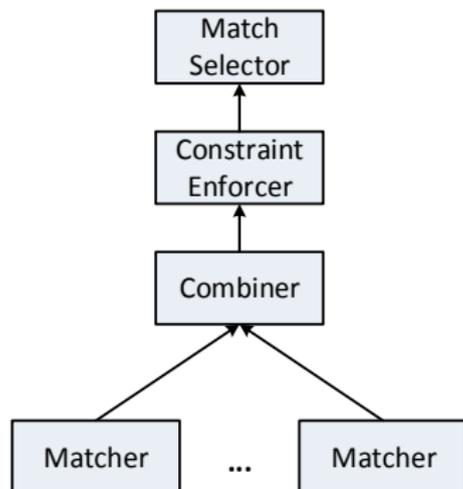
Aufbau von Schema Matching Systemen



Constraint Enforcer

- **Input:** eine Ähnlichkeitsmatrix, eine Menge an Constraints
- **Output:** eine Ähnlichkeitsmatrix
- Einführung von Kontextwissen durch Ändern der Ähnlichkeitsmatrix

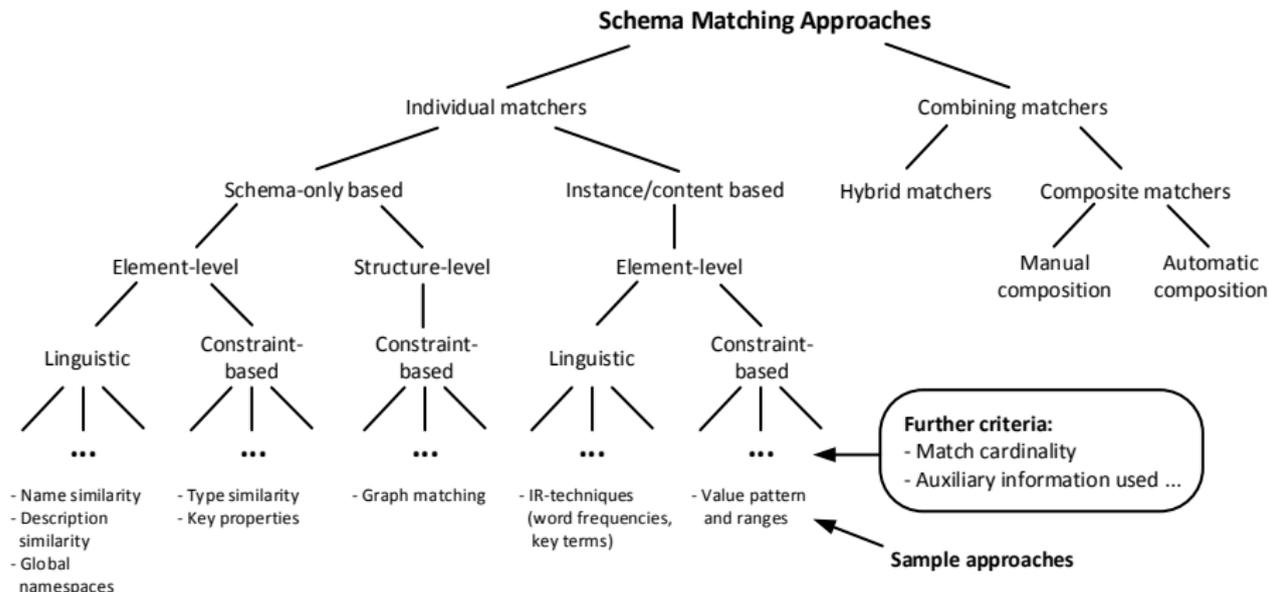
Aufbau von Schema Matching Systemen



Match Selector

- **Input:** eine Ähnlichkeitsmatrix
- **Output:** eine Menge an Korrespondenzen
- Benutzt die paarweise berechneten Ähnlichkeiten um eine (aufeinander abgestimmte) Menge an Korrespondenzen zu berechnen
- Einfacher Ansatz: Thresholding
- Komplexer Ansatz: Optimierungsproblem auf einem gewichteten bipartiten Graphen

Klassifizierung von Schema Matchern [RB01]



Schema Matching – Klassifikation

Schema Matching basierend auf

- Namen der Schemaelemente (*label-based*)
- Darunterliegende Daten (*instance-based*)
- Struktur des Schemas (*structure-based*)
- Mischformen

Agenda

- 1 Schema Matching
 - Problemstellung
 - Motivation

- 2 Schema Matching Systeme
 - **Label-based Matcher**
 - Instance-based Matcher
 - Structure-based Matcher
 - Combiner
 - Match Selector
 - Constraint Enforcer
 - n:m Korrespondenzen

Schema Matching – Label-based

- Gegeben: Zwei Schemata mit Elementmengen \mathcal{A} und \mathcal{B}

Schema Matching – Label-based

- Gegeben: Zwei Schemata mit Elementmengen \mathcal{A} und \mathcal{B}
- Kernidee:
 - Bilde Kreuzprodukt aller Elemente aus \mathcal{A} und \mathcal{B}
 - Für jedes Paar: vergleiche Ähnlichkeit bzgl. Elementnamen (Label) anhand Ähnlichkeitsmaß (z.B. *Edit distance* für Zeichenketten)
 - Ähnlichste Paare sind Matches

Schema Matching – Label-based

- Gegeben: Zwei Schemata mit Elementmengen \mathcal{A} und \mathcal{B}
- Kernidee:
 - Bilde Kreuzprodukt aller Elemente aus \mathcal{A} und \mathcal{B}
 - Für jedes Paar: vergleiche Ähnlichkeit bzgl. Elementnamen (Label) anhand Ähnlichkeitsmaß (z.B. *Edit distance* für Zeichenketten)
 - Ähnlichste Paare sind Matches
- Probleme:
 - Effizienz
 - Auswahl der besten Matches (globales Matching)
 - Synonyme und Homonyme werden nicht erkannt

Schema Matching – Label-based

Erhöhung der Matching Qualität durch Normalisierung

Schema Matching – Label-based

Erhöhung der Matching Qualität durch Normalisierung

- Zerlegung der Label nach bestimmten Trennzeichen wie Großbuchstaben, Nummern oder speziellen Symbolen
Beispiel: *saleLocID* wird zerlegt in *sale*, *Loc* und *ID*

Schema Matching – Label-based

Erhöhung der Matching Qualität durch Normalisierung

- Zerlegung der Label nach bestimmten Trennzeichen wie Großbuchstaben, Nummern oder speziellen Symbolen
Beispiel: *saleLocID* wird zerlegt in *sale*, *Loc* und *ID*
- Ausschreibung von Abkürzungen oder Akronymen
Beispiel: *Loc* wird erweitert zu *Location*

Schema Matching – Label-based

Erhöhung der Matching Qualität durch Normalisierung

- Zerlegung der Label nach bestimmten Trennzeichen wie Großbuchstaben, Nummern oder speziellen Symbolen
Beispiel: *saleLocID* wird zerlegt in *sale*, *Loc* und *ID*
- Ausschreibung von Abkürzungen oder Akronymen
Beispiel: *Loc* wird erweitert zu *Location*
- Entfernen von Artikeln, Präpositionen und Konjunktionen

Schema Matching – Label-based

Erhöhung der Matching Qualität durch Normalisierung

- Zerlegung der Label nach bestimmten Trennzeichen wie Großbuchstaben, Nummern oder speziellen Symbolen
Beispiel: *saleLocID* wird zerlegt in *sale*, *Loc* und *ID*
- Ausschreibung von Abkürzungen oder Akronymen
Beispiel: *Loc* wird erweitert zu *Location*
- Entfernen von Artikeln, Präpositionen und Konjunktionen
- Transformation von Wörtern in ihre Stammformen

Label-based (Beispiel)

DVD-Vendor

Movies(id, title, year)

Products(mid, releaseDate, releaseCompany, basePrice, rating, saleLocID)

Locations(lid, name, taxRate)

AGGREGATOR

Items(name, releaseInfo, classification, price)

label-based matcher

name = \langle name: 1, title: 0.2 \rangle

releaseInfo = \langle releaseDate: 0.5, releaseCompany: 0.5 \rangle

price = \langle basePrice: 0.8 \rangle

Agenda

- 1 Schema Matching
 - Problemstellung
 - Motivation

- 2 Schema Matching Systeme
 - Label-based Matcher
 - **Instance-based Matcher**
 - Structure-based Matcher
 - Combiner
 - Match Selector
 - Constraint Enforcer
 - n:m Korrespondenzen

Schema Matching – Instance-based

- Gegeben: Zwei Schemata mit Attributmengen \mathcal{A} und \mathcal{B} (jeweils mit darunterliegenden Daten)

Schema Matching – Instance-based

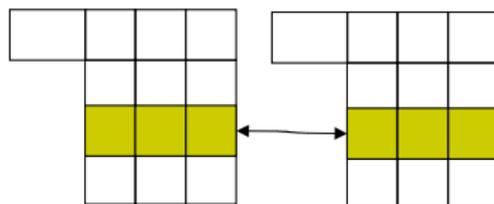
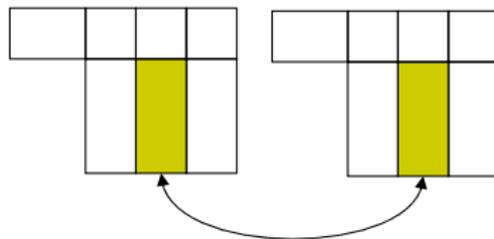
- Gegeben: Zwei Schemata mit Attributmengen \mathcal{A} und \mathcal{B} (jeweils mit darunterliegenden Daten)
- Kernidee:
 - Für jedes Attribut: extrahiere interessante Eigenschaften der Daten (z.B. Buchstabenverteilung, Länge)
 - Bilde Kreuzprodukt aller Attribute aus \mathcal{A} und \mathcal{B}
 - Für jedes Paar: vergleiche Ähnlichkeit bzgl. der Eigenschaften

Schema Matching – Instance-based

- Gegeben: Zwei Schemata mit Attributmengen \mathcal{A} und \mathcal{B} (jeweils mit darunterliegenden Daten)
- Kernidee:
 - Für jedes Attribut: extrahiere interessante Eigenschaften der Daten (z.B. Buchstabenverteilung, Länge)
 - Bilde Kreuzprodukt aller Attribute aus \mathcal{A} und \mathcal{B}
 - Für jedes Paar: vergleiche Ähnlichkeit bzgl. der Eigenschaften
- Probleme:
 - Auswahl der Eigenschaften
 - Menge der Daten: Sampling?
 - Vergleichsmethode (z.B. Naive Bayes)
 - Gewichtung (Maschinelles Lernen)

Instance-based Verfahren

- Konventionelle Lösung: Vertikal
 - Vergleich von Spalten
 - *Attribute classification*
 - Beispiel: [NHT⁺02]
- Andere Lösung: Horizontal
 - Vergleich von Tupeln
 - *Duplicate detection*
(trotz fehlender
Attribut-Korrespondenzen)
 - Attribut-Matching auf Basis der
Duplikate
 - Beispiel: [BN05]



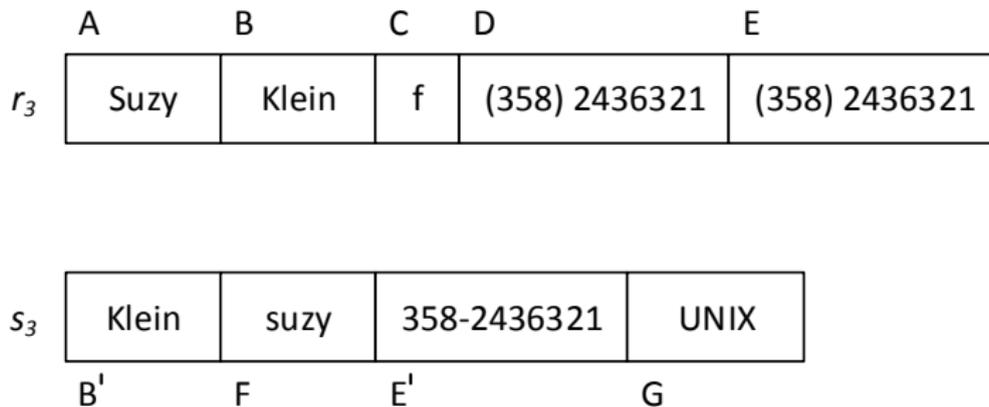
Instance-based (Beispiel - Vertikal)

SCHEMA S		
current-showing	address	phone
Lord of the Rings	Madison WI	(608) 695 2311
	Mountain View CA	(650) 277 1358

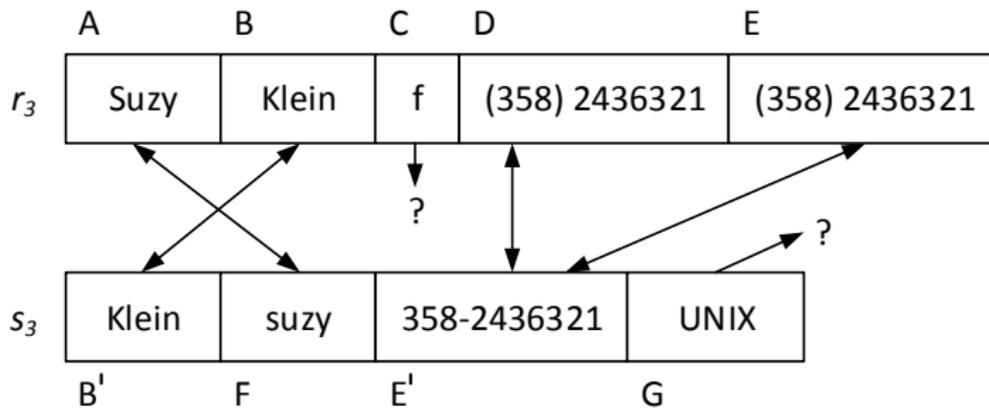
SCHEMA T		
name	location	phone
...	Milwaukee WI	...
	Palo Alto CA	
	Philadelphia PA	

- Korrespondenz zwischen *address* und *location* wird anhand der vorhandenen Attributwerte gefunden

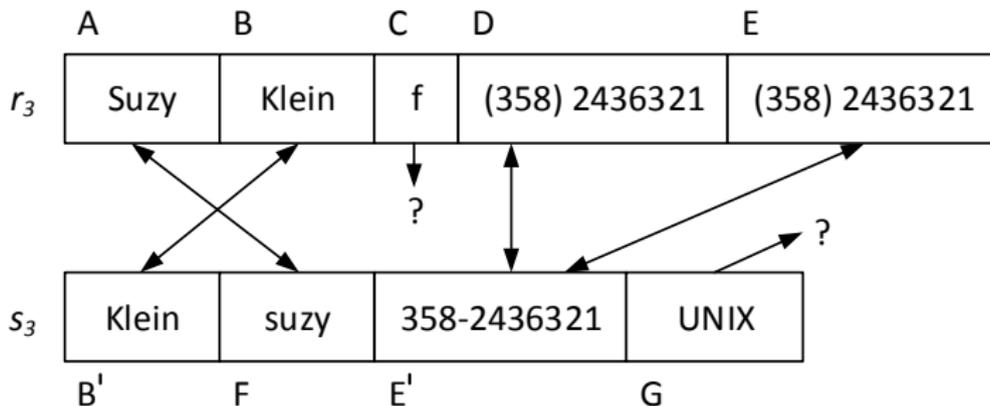
Instance-based (Beispiel - Horizontal) [BN05]



Instance-based (Beispiel - Horizontal) [BN05]



Instance-based (Beispiel - Horizontal) [BN05]



- Haben die Attribute C und G keine Korrespondenz?
- Korrespondiert Attribut E' mit D oder E?

Instance-based (Vertikale Verfahren)

Instance-based (Vertikale Verfahren)

- Recognizers
 - verwenden Kontextinformationen

Instance-based (Vertikale Verfahren)

- Recognizers
 - verwenden Kontextinformationen
- Overlap Matcher
 - bestimmen den Anteil gemeinsamer Datenwerte

Instance-based (Vertikale Verfahren)

- Recognizers
 - verwenden Kontextinformationen
- Overlap Matcher
 - bestimmen den Anteil gemeinsamer Datenwerte
- Klassifizierer
 - lernende Verfahren

Recognizer

Recognizer

- Benutzung von Wörterbüchern oder Regeln um Datenwerte bestimmter Attribute zu erkennen

Recognizer

- Benutzung von Wörterbüchern oder Regeln um Datenwerte bestimmter Attribute zu erkennen
- Attribute für die Recognizer nützlich sind:

Recognizer

- Benutzung von Wörterbüchern oder Regeln um Datenwerte bestimmter Attribute zu erkennen
- Attribute für die Recognizer nützlich sind:
 - Namen von Städten, Ländern oder Staaten
 - Personennamen
 - Farben, Bewertungen (e.g., -, o, +, etc.), Telefonnummern, Postleitzahlen
 - Genome, Proteine

Recognizer

- Benutzung von Wörterbüchern oder Regeln um Datenwerte bestimmter Attribute zu erkennen
 - Attribute für die Recognizer nützlich sind:
 - Namen von Städten, Ländern oder Staaten
 - Personennamen
 - Farben, Bewertungen (e.g., -, o, +, etc.), Telefonnummern, Postleitzahlen
 - Genome, Proteine
 - Vergleich eines Schemaelementes mit einer Menge an Konzepten
- ⇒ Instanzdaten des globalen Schemas nicht erforderlich

Overlap Matcher

Overlap Matcher

- Überlappung von Elementinstanzen bestimmen

Overlap Matcher

- Überlappung von Elementinstanzen bestimmen
- Nutzung mengenbasierter Ähnlichkeitsmaße (e.g. Jaccard)

Overlap Matcher

- Überlappung von Elementinstanzen bestimmen
- Nutzung mengenbasierter Ähnlichkeitsmaße (e.g. Jaccard)
- hauptsächlich nützlich für Attribute mit endlichen Wertebereichen

Overlap Matcher

- Überlappung von Elementinstanzen bestimmen
- Nutzung mengenbasierter Ähnlichkeitsmaße (e.g. Jaccard)
- hauptsächlich nützlich für Attribute mit endlichen Wertebereichen
- Problem: Die globale Sicht ist virtuell (keine Instanz vorhanden)

Klassifizierer

Klassifizierer

- **Idee:** Lerne binären Klassifizierer auf Elemente eines Schemas und nutze sie um die Elemente eines anderen Schemas zu klassifizieren

Klassifizierer

- **Idee:** Lerne binären Klassifizierer auf Elemente eines Schemas und nutze sie um die Elemente eines anderen Schemas zu klassifizieren
- Verschiedene Klassifizierungsmethoden geeignet: Naive Bayes, Entscheidungsbäume oder Support-Vector Machines

Klassifizierer

- **Idee:** Lerne binären Klassifizierer auf Elemente eines Schemas und nutze sie um die Elemente eines anderen Schemas zu klassifizieren
 - Verschiedene Klassifizierungsmethoden geeignet: Naive Bayes, Entscheidungsbäume oder Support-Vector Machines
 - Trainingsdaten
 - Positive Beispiele: Werte des betrachteten Elementes
 - Negative Beispiele: Werte von anderen Elementen
- pause
- Probleme
 - Alle negativen Beispiele den positiven zu unähnlich (Personennamen $\tilde{f}\tilde{A}\frac{1}{4}r$ Telefonnummern)
- ⇒ Klassifizierer wird trivial

Klassifizierer

Klassifizierer

- Vergleich zweier Schemaelemente anhand der Klassifizierung des zweiten Elementes mit dem Klassifizierer des ersten Elementes
 - Berechnung eines normierten *Confidence*-Wertes per Elementinstanz
 - Aggregation aller *Confidence*-Werte als Ähnlichkeit der Elemente

Klassifizierer

- Vergleich zweier Schemaelemente anhand der Klassifizierung des zweiten Elementes mit dem Klassifizierer des ersten Elementes
 - Berechnung eines normierten *Confidence*-Wertes per Elementinstanz
 - Aggregation aller *Confidence*-Werte als Ähnlichkeit der Elemente
- Unidirektional vs. bidirektional

Klassifizierer

- Vergleich zweier Schemaelemente anhand der Klassifizierung des zweiten Elementes mit dem Klassifizierer des ersten Elementes
 - Berechnung eines normierten *Confidence*-Wertes per Elementinstanz
 - Aggregation aller *Confidence*-Werte als Ähnlichkeit der Elemente
- Unidirektional vs. bidirektional
- Alle Quellschemata werden gegen das globale Schema gematched

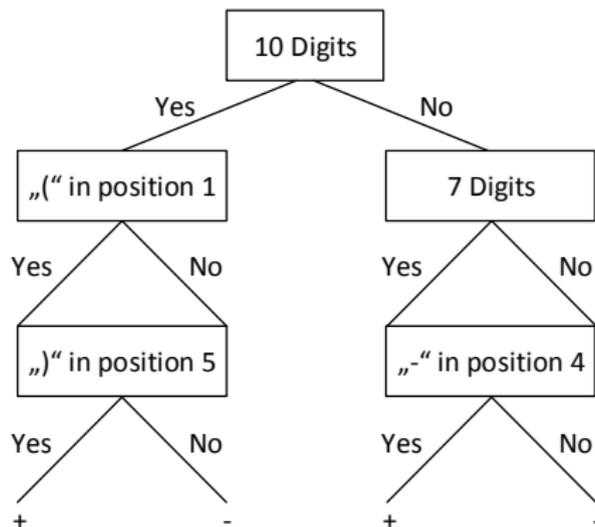
Klassifizierer

- Vergleich zweier Schemaelemente anhand der Klassifizierung des zweiten Elementes mit dem Klassifizierer des ersten Elementes
 - Berechnung eines normierten *Confidence*-Wertes per Elementinstanz
 - Aggregation aller *Confidence*-Werte als Ähnlichkeit der Elemente
 - Unidirektional vs. bidirektional
 - Alle Quellschemata werden gegen das globale Schema gematched
- ⇒ Wiederverwendbarkeit wird erhöht wenn Klassifizierer für das globale Schema gelernt werden

Klassifizierer

- Vergleich zweier Schemaelemente anhand der Klassifizierung des zweiten Elementes mit dem Klassifizierer des ersten Elementes
 - Berechnung eines normierten *Confidence*-Wertes per Elementinstanz
 - Aggregation aller *Confidence*-Werte als Ähnlichkeit der Elemente
- Unidirektional vs. bidirektional
- Alle Quellschemata werden gegen das globale Schema gematched
- ⇒ Wiederverwendbarkeit wird erhöht wenn Klassifizierer für das globale Schema gelernt werden
- Problem: Keine Instanz vorhanden

Klassifizierer (Beispiel)



- Entscheidungsbaum für Telefonnummern
- Jeder Eingabewert wird klassifiziert als
 - Telefonnummer (+)
 - keine Telefonnummer (-)

Klassifizierer

Klassifizierer

- Overlap Matcher und Klassifizierer brauchen Instanzdaten

Klassifizierer

- Overlap Matcher und Klassifizierer brauchen Instanzdaten
- **Problem:** Das globale Schema ist virtuell und hat keine Instanz

Klassifizierer

- Overlap Matcher und Klassifizierer brauchen Instanzdaten
- **Problem:** Das globale Schema ist virtuell und hat keine Instanz
- **Lösung:** Man benutzt bereits vorhandene Mappings zu anderen Quellschemata um Trainingsdaten für das globale Schema zu gewinnen

Klassifizierer

- Overlap Matcher und Klassifizierer brauchen Instanzdaten
 - **Problem:** Das globale Schema ist virtuell und hat keine Instanz
 - **Lösung:** Man benutzt bereits vorhandene Mappings zu anderen Quellschemata um Trainingsdaten für das globale Schema zu gewinnen
- ⇒ Das System lernt aus vergangenen Matchingprozessen um die Qualität und Effizienz zukünftiger Matchingprozesse zu erhöhen

Agenda

- 1 Schema Matching
 - Problemstellung
 - Motivation

- 2 Schema Matching Systeme
 - Label-based Matcher
 - Instance-based Matcher
 - **Structure-based Matcher**
 - Combiner
 - Match Selector
 - Constraint Enforcer
 - n:m Korrespondenzen

Schema Matching – Structure-based

- Gegeben: Zwei Schemata mit Attributmengen \mathcal{A} und \mathcal{B}
- Kernidee:
 - Ausnutzen der (komplexen) Struktur der Schemata
 - Hierarchieebene
 - Elementtyp (Attribut, Relation, ...)
 - Nachbarschaftsbeziehungen

Beispiel: Similarity Flooding [MGMR02]

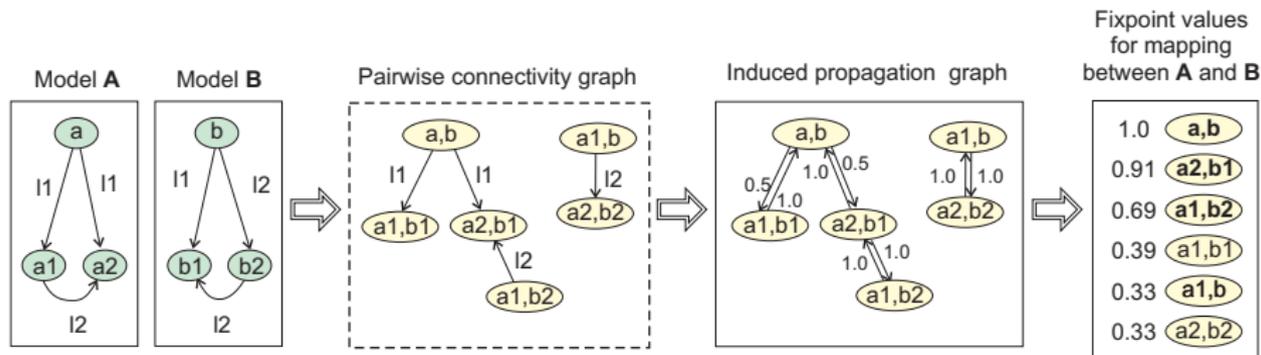
- Gegeben initiale Ähnlichkeit zwischen Schemaelementen (z.B. durch *edit distance* oder Instanzanalyse)

Beispiel: Similarity Flooding [MGMR02]

- Gegeben initiale Ähnlichkeit zwischen Schemaelementen (z.B. durch *edit distance* oder Instanzanalyse)
- Lasse Ähnlichkeiten abfärben auf die Nachbarn
 - Nachbarn sind durch Struktur definiert
 - Sind alle Nachbarn von x und y ähnlich zueinander, sind (vielleicht) auch x und y ein *Match*

Beispiel: Similarity Flooding [MGMR02]

- Gegeben initiale Ähnlichkeit zwischen Schemaelementen (z.B. durch *edit distance* oder Instanzanalyse)
- Lasse Ähnlichkeiten abfärben auf die Nachbarn
 - Nachbarn sind durch Struktur definiert
 - Sind alle Nachbarn von x und y ähnlich zueinander, sind (vielleicht) auch x und y ein *Match*
- Analogie: Man flutet das Netzwerk der Ähnlichkeiten bis ein Gleichgewicht erreicht ist



Anfrage-basierte Matcher

- Vergleich von Schemaelementen basierend auf der Art und Weise wie sie in Anfragen verwendet werden

Anfrage-basierte Matcher

- Vergleich von Schemaelementen basierend auf der Art und Weise wie sie in Anfragen verwendet werden
- Erfordert Vergleich von semantisch ähnlichen Anfragen

Anfrage-basierte Matcher

- Vergleich von Schemaelementen basierend auf der Art und Weise wie sie in Anfragen verwendet werden
- Erfordert Vergleich von semantisch ähnlichen Anfragen
- Problem: Welche Anfragen sind semantisch ähnlich?

Anfrage-basierte Matcher

- Vergleich von Schemaelementen basierend auf der Art und Weise wie sie in Anfragen verwendet werden
- Erfordert Vergleich von semantisch ähnlichen Anfragen
- Problem: Welche Anfragen sind semantisch ähnlich?
- Auswahl basierend auf:
 - Kontextwissen wie z.B. Kenntnisse der Schnittstellen
 - Statistischen Mustern wie z.B. Häufigkeit oder Größe der Ergebnismenge

Anfrage-basierte Matcher

- Vergleich von Schemaelementen basierend auf der Art und Weise wie sie in Anfragen verwendet werden
- Erfordert Vergleich von semantisch ähnlichen Anfragen
- Problem: Welche Anfragen sind semantisch ähnlich?
- Auswahl basierend auf:
 - Kontextwissen wie z.B. Kenntnisse der Schnittstellen
 - Statistischen Mustern wie z.B. Häufigkeit oder Größe der Ergebnismenge
- Erfordert Zugriff auf Anfragen (und Anfragestatistiken) der einzelnen Quellen (Anfragen des globalen Schemas sind bekannt)

Anfrage-basierte Matcher (Beispiel 1)

Anfrage-basiertes Matching

```
SELECT Name, Alter, Ort
FROM   Person
WHERE  EMail = ...
```

```
SELECT Fname, Lname, DoB, Residence
FROM   Person
WHERE  EMail = ...
```

Indiz für Korrespondenz zwischen

Anfrage-basierte Matcher (Beispiel 1)

Anfrage-basiertes Matching

```
SELECT Name, Alter, Ort  
FROM Person  
WHERE EMail = ...
```

```
SELECT Fname, Lname, DoB, Residence  
FROM Person  
WHERE EMail = ...
```

Indiz für Korrespondenz zwischen

- Name und {FName, LName}

Anfrage-basierte Matcher (Beispiel 1)

Anfrage-basiertes Matching

```
SELECT Name, Alter, Ort
FROM Person
WHERE EMail = ...

SELECT FName, LName, DoB, Residence
FROM Person
WHERE EMail = ...
```

Indiz für Korrespondenz zwischen

- Name und {FName, LName}
- Alter und DoB

Anfrage-basierte Matcher (Beispiel 1)

Anfrage-basiertes Matching

```
SELECT Name, Alter, Ort
FROM Person
WHERE EMail = ...

SELECT Fname, Lname, DoB, Residence
FROM Person
WHERE EMail = ...
```

Indiz für Korrespondenz zwischen

- Name und {FName, LName}
- Alter und DoB
- Ort und Residence

Anfrage-basierte Matcher (Beispiel 2)

Anfrage-basiertes Matching

```
SELECT Phone
FROM Person
WHERE Residence = 'Hamburg'
```

```
SELECT Vorname, Nachname
FROM Person
WHERE Ort = 'Hamburg'
```

Indiz für Korrespondenz zwischen

Anfrage-basierte Matcher (Beispiel 2)

Anfrage-basiertes Matching

```
SELECT Phone                                SELECT Vorname, Nachname
FROM   Person                                FROM   Person
WHERE  Residence = 'Hamburg'                WHERE  Ort = 'Hamburg'
```



Indiz für Korrespondenz zwischen

- Residence und Ort

Anfrage-basierte Matcher (Beispiel 3)

Anfrage-basiertes Matching

```
SELECT Phone
FROM Person
WHERE Firstname = 'Klaus'
AND Surname = 'Meier'
```

```
SELECT Phone
FROM Person
WHERE Name = 'Klaus Meier'
```

Indiz für Korrespondenz zwischen

Anfrage-basierte Matcher (Beispiel 3)

Anfrage-basiertes Matching

```
SELECT Phone                                SELECT Phone
FROM   Person                                FROM   Person
WHERE  Firstname = 'Klaus'          WHERE  Name = 'Klaus Meier'
AND    Surname = 'Meier'
```

Indiz für Korrespondenz zwischen

- {Firstname, Surname} und Name

Anfrage-basierte Matcher (Beispiel 4)

Anfrage-basiertes Matching

```
SELECT f.Title, a2.Name           SELECT f.Title, a.Name
FROM   Film f, Act a1, Actor a2   FROM   Film f, Plays p, Actor a
WHERE  a1.Film = f.FID           WHERE  p.FID = f.FID
AND    a1.Actor = a2.AID        AND    p.AID = a.AID
```

Annahme: FK zwischen den einzelnen Tabellen nicht existent
oder nicht bekannt

Indiz für Korrespondenz zwischen

Anfrage-basierte Matcher (Beispiel 4)

Anfrage-basiertes Matching

SELECT f.Title, a2.Name		SELECT f.Title, a.Name
FROM Film f, Act a1, Actor a2		FROM Film f, Plays p, Actor a
WHERE a1.Film = f.FID		WHERE p.FID = f.FID
AND a1.Actor = a2.AID		AND p.AID = a.AID

Annahme: FK zwischen den einzelnen Tabellen nicht existent oder nicht bekannt

Indiz für Korrespondenz zwischen

- Act und Plays (Tabellen-Ebene)
- Act.Film und Plays.FID (Attribut-Ebene)
- Act.Actor und Plays.AID (Attribut-Ebene)

Anfrage-basierte Matcher (Beispiel 5)

Anfrage-basiertes Matching

```
SELECT Titel, Länge  
FROM Film  
WHERE Genre = 'Horror'
```

```
SELECT Titel, Länge  
FROM Horrorfilm
```

Indiz für Korrespondenz zwischen

Anfrage-basierte Matcher (Beispiel 5)

Anfrage-basiertes Matching

```
SELECT Titel, Länge
FROM Film
WHERE Genre = 'Horror'
```

```
SELECT Titel, Länge
FROM Horrorfilm
```

Indiz für Korrespondenz zwischen

- {Film, Film.Genre='Horror'} und Horrorfilm

Agenda

- ① Schema Matching
 - Problemstellung
 - Motivation

- ② Schema Matching Systeme
 - Label-based Matcher
 - Instance-based Matcher
 - Structure-based Matcher
 - **Combiner**
 - Match Selector
 - Constraint Enforcer
 - n:m Korrespondenzen

Kombinierung von Match Vorhersagen

realestate.com

listed-price	contact-name	contact-phone	office	comments
\$250K	James Smith	(305) 729 0831	(305) 616 1822	Fantastic house
\$320K	Mike Doan	(617) 253 1429	(617) 112 2315	Great location
.....

homes.com

sold-at	contact-agent	extra-info
\$350K	(206) 634 9435	Beautiful yard
\$230K	(617) 335 4243	Close to Seattle

Kombinierung von Match Vorhersagen

realestate.com

listed-price	contact-name	contact-phone	office	comments
\$250K	James Smith	(305) 729 0831	(305) 616 1822	Fantastic house
\$320K	Mike Doan	(617) 253 1429	(617) 112 2315	Great location
.....

homes.com

sold-at	contact-agent	extra-info
\$350K	(206) 634 9435	Beautiful yard
\$230K	(617) 335 4243	Close to Seattle

- Label-based Matching: *contact-agent* matched entweder mit *contact-name* oder *contact-phone*

Kombinierung von Match Vorhersagen

realestate.com

listed-price	contact-name	contact-phone	office	comments
\$250K	James Smith	(305) 729 0831	(305) 616 1822	Fantastic house
\$320K	Mike Doan	(617) 253 1429	(617) 112 2315	Great location
.....

homes.com

sold-at	contact-agent	extra-info
\$350K	(206) 634 9435	Beautiful yard
\$230K	(617) 335 4243	Close to Seattle

- Label-based Matching: *contact-agent* matched entweder mit *contact-name* oder *contact-phone*
- Instance-based Matching: *contact-agent* matched entweder mit *contact-phone* oder *office*

Kombinierung von Match Vorhersagen

realestate.com

listed-price	contact-name	contact-phone	office	comments
\$250K	James Smith	(305) 729 0831	(305) 616 1822	Fantastic house
\$320K	Mike Doan	(617) 253 1429	(617) 112 2315	Great location
.....

homes.com

sold-at	contact-agent	extra-info
\$350K	(206) 634 9435	Beautiful yard
\$230K	(617) 335 4243	Close to Seattle

- Label-based Matching: *contact-agent* matched entweder mit *contact-name* oder *contact-phone*
- Instance-based Matching: *contact-agent* matched entweder mit *contact-phone* oder *office*
- Beides: *contact-agent* matched mit *contact-phone*

Kombinierung von Match Vorhersagen

Kombinierung von Match Vorhersagen

- Aggregation mehrerer Ähnlichkeitsmatrizen

Kombinierung von Match Vorhersagen

- Aggregation mehrerer Ähnlichkeitsmatrizen
- Einfache Aggregationsmethoden
 - **Durchschnitt:** Wir vertrauen allen Matchern gleich
 - **Maximum:** Vertrauen in positive Signale einzelner Matcher
 - **Minimum:** Vertrauen in negative Signale einzelner Matcher

Kombinierung von Match Vorhersagen

- Aggregation mehrerer Ähnlichkeitsmatrizen
- Einfache Aggregationsmethoden
 - **Durchschnitt:** Wir vertrauen allen Matchern gleich
 - **Maximum:** Vertrauen in positive Signale einzelner Matcher
 - **Minimum:** Vertrauen in negative Signale einzelner Matcher
- Handgeschriebene Skripte

Kombinierung von Match Vorhersagen

- Aggregation mehrerer Ähnlichkeitsmatrizen
- Einfache Aggregationsmethoden
 - **Durchschnitt:** Wir vertrauen allen Matchern gleich
 - **Maximum:** Vertrauen in positive Signale einzelner Matcher
 - **Minimum:** Vertrauen in negative Signale einzelner Matcher
- Handgeschriebene Skripte
 - Manchmal gibt es Vorkenntnisse welche Matcher in welchen Domänen zuverlässig sind

Kombinierung von Match Vorhersagen

- Aggregation mehrerer Ähnlichkeitsmatrizen
- Einfache Aggregationsmethoden
 - **Durchschnitt:** Wir vertrauen allen Matchern gleich
 - **Maximum:** Vertrauen in positive Signale einzelner Matcher
 - **Minimum:** Vertrauen in negative Signale einzelner Matcher
- Handgeschriebene Skripte
 - Manchmal gibt es Vorkenntnisse welche Matcher in welchen Domänen zuverlässig sind
 - Durch Skripte sind beliebige Kombinationen von Aggregationen und Kontextinformationen möglich
 - **Beispiel:** Benutze Matcher x wenn es sich um einen Vergleich mit dem Attribut *Adresse* handelt

Kombinierung von Match Vorhersagen

- Aggregation mehrerer Ähnlichkeitsmatrizen
- Einfache Aggregationsmethoden
 - **Durchschnitt:** Wir vertrauen allen Matchern gleich
 - **Maximum:** Vertrauen in positive Signale einzelner Matcher
 - **Minimum:** Vertrauen in negative Signale einzelner Matcher
- Handgeschriebene Skripte
 - Manchmal gibt es Vorkenntnisse welche Matcher in welchen Domänen zuverlässig sind
 - Durch Skripte sind beliebige Kombinationen von Aggregationen und Kontextinformationen möglich
 - **Beispiel:** Benutze Matcher x wenn es sich um einen Vergleich mit dem Attribut *Adresse* handelt
- Gewichtete Aggregation
 - Gewichte von Domänenexperten spezifiziert
 - Lernverfahren um elementspezifischer Gewichte zu bestimmen

Agenda

- ① Schema Matching
 - Problemstellung
 - Motivation

- ② Schema Matching Systeme
 - Label-based Matcher
 - Instance-based Matcher
 - Structure-based Matcher
 - Combiner
 - **Match Selector**
 - Constraint Enforcer
 - n:m Korrespondenzen

Match Selector

Match Selector

- Bildet die Ähnlichkeitsmatrix auf eine Menge an Korrespondenzen ab (sog. *Match Combination*)

Match Selector

- Bildet die Ähnlichkeitsmatrix auf eine Menge an Korrespondenzen ab (sog. *Match Combination*)
- Einfache Methode: Wähle alle Elementpaare deren Ähnlichkeit größer θ ist

Match Selector

- Bildet die Ähnlichkeitsmatrix auf eine Menge an Korrespondenzen ab (sog. *Match Combination*)
- Einfache Methode: Wähle alle Elementpaare deren Ähnlichkeit größer θ ist
- Problem: Nicht konsistent, da ein Element von Schema \mathcal{A} mit mehreren Elementen von Schema \mathcal{B} gematched werden kann (keine 1:n Korrespondenz!)

Match Selector

- Bildet die Ähnlichkeitsmatrix auf eine Menge an Korrespondenzen ab (sog. *Match Combination*)
- Einfache Methode: Wähle alle Elementpaare deren Ähnlichkeit größer θ ist
- Problem: Nicht konsistent, da ein Element von Schema \mathcal{A} mit mehreren Elementen von Schema \mathcal{B} gematched werden kann (keine 1:n Korrespondenz!)
- Bessere aber auch aufwendigere Verfahren:
Stable Marriage und *Maximum Weighted Matching*

Match Selector

- Bildet die Ähnlichkeitsmatrix auf eine Menge an Korrespondenzen ab (sog. *Match Combination*)
- Einfache Methode: Wähle alle Elementpaare deren Ähnlichkeit größer θ ist
- Problem: Nicht konsistent, da ein Element von Schema \mathcal{A} mit mehreren Elementen von Schema \mathcal{B} gematched werden kann (keine 1:n Korrespondenz!)
- Bessere aber auch aufwendigere Verfahren:
Stable Marriage und *Maximum Weighted Matching*
- Manchmal ist es sinnvoll mehrere Match Combinations auszugeben und dem Nutzer das Auflösen von verbleibenden Unsicherheiten überlässt

Stable Marriage

- Gegeben: n Frauen (Elemente in Schema \mathcal{A}) und m Männer (Elemente in Schema \mathcal{B})

Stable Marriage

- Gegeben: n Frauen (Elemente in Schema \mathcal{A}) und m Männer (Elemente in Schema \mathcal{B})
- Monogamie:
Je eine Frau kann nur mit je einem Mann verheiratet sein (nur 1:1 Matches)

Stable Marriage

- Gegeben: n Frauen (Elemente in Schema \mathcal{A}) und m Männer (Elemente in Schema \mathcal{B})
- Monogamie:
Je eine Frau kann nur mit je einem Mann verheiratet sein (nur 1:1 Matches)
- Jede Frau hat eine Rangliste der Männer und umgekehrt.
Beim Schema Matching:
 - Rangliste basiert auf den Ähnlichkeiten der Schemaelemente
 - Trotz symmetrischer Ähnlichkeitsmatrix, Ranglisten können sich stark unterscheiden

Stable Marriage

- Gegeben: n Frauen (Elemente in Schema \mathcal{A}) und m Männer (Elemente in Schema \mathcal{B})
- Monogamie:
Je eine Frau kann nur mit je einem Mann verheiratet sein (nur 1:1 Matches)
- Jede Frau hat eine Rangliste der Männer und umgekehrt.
Beim Schema Matching:
 - Rangliste basiert auf den Ähnlichkeiten der Schemaelemente
 - Trotz symmetrischer Ähnlichkeitsmatrix, Ranglisten können sich stark unterscheiden
- Gesucht: Paarung (globales Matching), so dass niemals gilt:
 - f_1 heiratet m_1 , f_2 heiratet m_2
 - aber f_1 bevorzugt m_2 und m_2 bevorzugt f_1 (instabile Paarung)

Stable Marriage

Männer (1-4)	Frauen (A-D)
1: B, D , A, C	A : 2, 1, 4 , 3
2: C , A, D, B	B : 4, 3 , 1, 2
3: B , C, A, D	C : 1, 4, 3, 2
4: D, A , C, B	D : 2, 1 , 4, 3

Stable Marriage

Männer (1-4)	Frauen (A-D)
1: B, D, A, C	A: 2, 1, 4, 3
2: C, A, D, B	B: 4, 3, 1, 2
3: B, C, A, D	C: 1, 4, 3, 2
4: D, A, C, B	D: 2, 1, 4, 3

- 1 stellt Antrag an B, sie willigt ein: (1, B)

Stable Marriage

Männer (1-4)	Frauen (A-D)
1: B, D, A, C	A: 2, 1, 4, 3
2: C, A, D, B	B: 4, 3, 1, 2
3: B, C, A, D	C: 1, 4, 3, 2
4: D, A, C, B	D: 2, 1, 4, 3

- 1 stellt Antrag an B, sie willigt ein: (1, B)
- 2 stellt Antrag an C, sie willigt ein: (1, B) (2, C)

Stable Marriage

Männer (1-4)	Frauen (A-D)
1: B, D, A, C	A: 2, 1, 4, 3
2: C, A, D, B	B: 4, 3, 1, 2
3: B, C, A, D	C: 1, 4, 3, 2
4: D, A, C, B	D: 2, 1, 4, 3

- 1 stellt Antrag an B, sie willigt ein: (1, B)
- 2 stellt Antrag an C, sie willigt ein: (1, B) (2, C)
- 3 stellt Antrag an B, sie willigt ein und verlässt 1: (2, C) (3, B)

Stable Marriage

Männer (1-4)	Frauen (A-D)
1: B, D, A, C	A: 2, 1, 4, 3
2: C, A, D, B	B: 4, 3, 1, 2
3: B, C, A, D	C: 1, 4, 3, 2
4: D, A, C, B	D: 2, 1, 4, 3

- 1 stellt Antrag an B, sie willigt ein: (1, B)
- 2 stellt Antrag an C, sie willigt ein: (1, B) (2, C)
- 3 stellt Antrag an B, sie willigt ein und verlässt 1: (2, C) (3, B)
- 1 stellt Antrag an D, sie willigt ein: (1, D) (2, C) (3, B)

Stable Marriage

Männer (1-4)	Frauen (A-D)
1: B, D, A, C	A: 2, 1, 4, 3
2: C, A, D, B	B: 4, 3, 1, 2
3: B, C, A, D	C: 1, 4, 3, 2
4: D, A, C, B	D: 2, 1, 4, 3

- 1 stellt Antrag an B, sie willigt ein: (1, B)
- 2 stellt Antrag an C, sie willigt ein: (1, B) (2, C)
- 3 stellt Antrag an B, sie willigt ein und verlässt 1: (2, C) (3, B)
- 1 stellt Antrag an D, sie willigt ein: (1, D) (2, C) (3, B)
- 4 stellt Antrag an D, sie lehnt ab: (1, D) (2, C) (3, B)

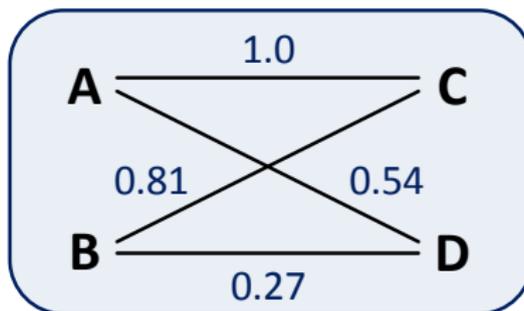
Stable Marriage

Männer (1-4)	Frauen (A-D)
1: B, D, A, C	A: 2, 1, 4, 3
2: C, A, D, B	B: 4, 3, 1, 2
3: B, C, A, D	C: 1, 4, 3, 2
4: D, A, C, B	D: 2, 1, 4, 3

- 1 stellt Antrag an B, sie willigt ein: (1, B)
- 2 stellt Antrag an C, sie willigt ein: (1, B) (2, C)
- 3 stellt Antrag an B, sie willigt ein und verlässt 1: (2, C) (3, B)
- 1 stellt Antrag an D, sie willigt ein: (1, D) (2, C) (3, B)
- 4 stellt Antrag an D, sie lehnt ab: (1, D) (2, C) (3, B)
- 4 stellt Antrag an A, sie willigt ein: (1, D) (2, C) (3, B) (4, A)

Maximum Weighted Matching

- Alternative zu *Stable Marriage*
- Suche Matching mit maximalem Gewicht in bipartiten Graphen
 - Bipartit:
 - Knoten in zwei Klassen (Quelle & Ziel)
 - Kanten nur zwischen Knoten verschiedener Klassen (Korrespondenzen)
 - Maximiere Summe der einzelnen Gewichte/Ähnlichkeiten



Maximum Weighted Matching (Wdh.)

Gegeben: Ein (gewichteter) Graph $G = (V, E, w)$

- Ein Matching M von G ist eine Menge an Kanten die sich keine Knoten teilen (die Kanten sind paarweise nicht adjazent)

Maximum Weighted Matching (Wdh.)

Gegeben: Ein (gewichteter) Graph $G = (V, E, w)$

- Ein Matching M von G ist eine Menge an Kanten die sich keine Knoten teilen (die Kanten sind paarweise nicht adjazent)
- Ein Maximal Matching M von G ist ein Matching das nicht erweitert werden kann (d.h. das Hinzufügen einer weiteren Kante von G bedeutet, dass M kein Matching mehr ist)

Maximum Weighted Matching (Wdh.)

Gegeben: Ein (gewichteter) Graph $G = (V, E, w)$

- Ein Matching M von G ist eine Menge an Kanten die sich keine Knoten teilen (die Kanten sind paarweise nicht adjazent)
- Ein Maximal Matching M von G ist ein Matching das nicht erweitert werden kann (d.h. das Hinzufügen einer weiteren Kante von G bedeutet, dass M kein Matching mehr ist)
- Ein Maximum Matching M von G ist ein Matching das (unter allen mgl. Matchings) die maximale Anzahl an Kanten besitzt

Maximum Weighted Matching (Wdh.)

Gegeben: Ein (gewichteter) Graph $G = (V, E, w)$

- Ein Matching M von G ist eine Menge an Kanten die sich keine Knoten teilen (die Kanten sind paarweise nicht adjazent)
- Ein Maximal Matching M von G ist ein Matching das nicht erweitert werden kann (d.h. das Hinzufügen einer weiteren Kante von G bedeutet, dass M kein Matching mehr ist)
- Ein Maximum Matching M von G ist ein Matching das (unter allen mgl. Matchings) die maximale Anzahl an Kanten besitzt
- Ein Maximum Weighted Matching M von G ist ein Matching bei dem die Gesamtsumme aller Kantengewichte maximal ist

Maximum Weighted Matching

Gegeben: Ein (gewichteter) Graph $G = (V, E, w)$

- Finden des Maximum Weighted Matching M von G ist ein Zuordnungsproblem
- **bipartite Graphen:** Hungarian algorithm (Komplexität $\mathcal{O}(V^2E)$ oder $\mathcal{O}(V^2 \log V + VE)$)
- **generell:** Edmonds' algorithm (Komplexität $\mathcal{O}(V^2E)$)

Maximum Weighted Matching

Gegeben: Ein (gewichteter) Graph $G = (V, E, w)$

- Finden des Maximum Weighted Matching M von G ist ein Zuordnungsproblem
- **bipartite Graphen:** Hungarian algorithm (Komplexität $\mathcal{O}(V^2E)$ oder $\mathcal{O}(V^2 \log V + VE)$)
- **generell:** Edmonds' algorithm (Komplexität $\mathcal{O}(V^2E)$)

Anmerkungen:

- Das Maximum Weighted Matching soll nur Korrespondenzen mit großer Glaubwürdigkeit enthalten
- ⇒ Alle Korrespondenzen deren Glaubwürdigkeit kleiner als ein Schwellwert ist, bekommen die Glaubwürdigkeit 0 (in hybriden String Matching Verfahren ähnlich)

Agenda

- 1 Schema Matching
 - Problemstellung
 - Motivation

- 2 Schema Matching Systeme
 - Label-based Matcher
 - Instance-based Matcher
 - Structure-based Matcher
 - Combiner
 - Match Selector
 - **Constraint Enforcer**
 - n:m Korrespondenzen

Einbinden von Integritätsbedingungen

Einbinden von Integritätsbedingungen

- Domänenwissen kann helfen die Glaubhaftigkeit einiger Korrespondenzen zu erhöhen/verringern

Einbinden von Integritätsbedingungen

- Domänenwissen kann helfen die Glaubhaftigkeit einiger Korrespondenzen zu erhöhen/verringern
- Modellierung von Domänenwissen mit Hilfe von Integritätsbedingungen

Einbinden von Integritätsbedingungen

- Domänenwissen kann helfen die Glaubhaftigkeit einiger Korrespondenzen zu erhöhen/verringern
- Modellierung von Domänenwissen mit Hilfe von Integritätsbedingungen
- Glaubhaftigkeit einer Korrespondenz erhöht sich, wenn sie viele/alle Integritätsbedingungen erfüllt

Einbinden von Integritätsbedingungen

- Domänenwissen kann helfen die Glaubhaftigkeit einiger Korrespondenzen zu erhöhen/verringern
- Modellierung von Domänenwissen mit Hilfe von Integritätsbedingungen
- Glaubhaftigkeit einer Korrespondenz erhöht sich, wenn sie viele/alle Integritätsbedingungen erfüllt
- Glaubhaftigkeit einer Korrespondenz verringert sich, wenn sie einige Integritätsbedingungen verletzt

Einbinden von Integritätsbedingungen

- Domänenwissen kann helfen die Glaubhaftigkeit einiger Korrespondenzen zu erhöhen/verringern
- Modellierung von Domänenwissen mit Hilfe von Integritätsbedingungen
- Glaubhaftigkeit einer Korrespondenz erhöht sich, wenn sie viele/alle Integritätsbedingungen erfüllt
- Glaubhaftigkeit einer Korrespondenz verringert sich, wenn sie einige Integritätsbedingungen verletzt
- Ändern der Glaubhaftigkeiten von Korrespondenzen durch Modifikation der Ähnlichkeitsmatrix

Beispiel: Modellierung von Domänenwissen [DHI12]

average-combiner

name = \langle name: 0.6, title: 0.5 \rangle

releaseInfo = \langle releaseDate: 0.6, releaseCompany: 0.25 \rangle

classification = \langle rating: 0.3 \rangle

price = \langle basePrice: 0.5 \rangle

Beispiel: Modellierung von Domänenwissen [DHI12]

average-combiner

name = \langle name: 0.6, title: 0.5 \rangle

releaseInfo = \langle releaseDate: 0.6, releaseCompany: 0.25 \rangle

classification = \langle rating: 0.3 \rangle

price = \langle basePrice: 0.5 \rangle

- **Domänenwissen:** die meisten Filmtitel (AGGREGATOR.name) bestehen aus vier Wörtern oder mehr

Beispiel: Modellierung von Domänenwissen [DHI12]

average-combiner

name = \langle name: 0.6, title: 0.5 \rangle

releaseInfo = \langle releaseDate: 0.6, releaseCompany: 0.25 \rangle

classification = \langle rating: 0.3 \rangle

price = \langle basePrice: 0.5 \rangle

- **Domänenwissen:** die meisten Filmtitel (AGGREGATOR.name) bestehen aus vier Wörtern oder mehr
- ⇒ **Bedingung:** „*Ein Attribut A kann AGGREGATOR.name nur matchen, wenn in einem zufällig gewählten Sample von 100 Datenwerten mindestens 10 Werte vier Wörter oder mehr besitzen*“

Arten von Integritätsbedingungen

Arten von Integritätsbedingungen

- Zwei Arten von Integritätsbedingungen

Arten von Integritätsbedingungen

- Zwei Arten von Integritätsbedingungen
- Harte Bedingungen
 - müssen erfüllt sein
 - keine der ausgewählten Korrespondenzen darf sie verletzen

Arten von Integritätsbedingungen

- Zwei Arten von Integritätsbedingungen
- Harte Bedingungen
 - müssen erfüllt sein
 - keine der ausgewählten Korrespondenzen darf sie verletzen
- Weiche Bedingungen
 - sind eher heuristischer Natur
(können unter Umständen verletzt werden)
 - Verletzung dieser Bedingungen soll minimiert werden

Arten von Integritätsbedingungen

- Zwei Arten von Integritätsbedingungen
- Harte Bedingungen
 - müssen erfüllt sein
 - keine der ausgewählten Korrespondenzen darf sie verletzen
- Weiche Bedingungen
 - sind eher heuristischer Natur
(können unter Umständen verletzt werden)
 - Verletzung dieser Bedingungen soll minimiert werden
- Jede Bedingung wird mit einem Kostenwert versehen
 - für harte Bedingungen sind die Kosten unendlich
 - für weiche Bedingungen kann irgendein positiver Kostenwert gewählt werden

Beispiel: Integritätsbedingungen [DHI12]

BOOK-VENDOR

Books(ISBN, publisher, pubCountry, title, review)

Inventory(ISBN, quantity, location)

DISTRIBUTOR

Items(code, name, brand, origin, desc)

InStore(code, availQuant)

	Constraints	Costs
c_1	If $A = \text{Items.code}$, then A is a key	∞
c_2	If $A = \text{Items.desc}$, then any random sample of 100 data instances of A must have an average length of at least 20 words	1.5

Korrespondenz übergreifende Bedingungen

Korrespondenz übergreifende Bedingungen

- Problem: Integritätsbedingungen betreffen oft Kombinationen von Korrespondenzen (anstatt einzelne)

Korrespondenz übergreifende Bedingungen

- Problem: Integritätsbedingungen betreffen oft Kombinationen von Korrespondenzen (anstatt einzelne)
- ⇒ Modellierung einer reduzierten Glaubwürdigkeit in der Ähnlichkeitsmatrix nicht möglich

Korrespondenz übergreifende Bedingungen

- Problem: Integritätsbedingungen betreffen oft Kombinationen von Korrespondenzen (anstatt einzelne)
- ⇒ Modellierung einer reduzierten Glaubwürdigkeit in der Ähnlichkeitsmatrix nicht möglich
- Daher Auswahl der besten Match Combination die alle (harten) Bedingungen erfüllt bzw. die geringsten Kosten hat (⇒ Anwendung nach oder kombiniert mit Match Selector)

Korrespondenz übergreifende Bedingungen

- Problem: Integritätsbedingungen betreffen oft Kombinationen von Korrespondenzen (anstatt einzelne)
- ⇒ Modellierung einer reduzierten Glaubwürdigkeit in der Ähnlichkeitsmatrix nicht möglich
- Daher Auswahl der besten Match Combination die alle (harten) Bedingungen erfüllt bzw. die geringsten Kosten hat (⇒ Anwendung nach oder kombiniert mit Match Selector)
- Frage: Wie bestimmt man die Güte einer Match Combination?

Korrespondenz übergreifende Bedingungen

- Problem: Integritätsbedingungen betreffen oft Kombinationen von Korrespondenzen (anstatt einzelne)
- ⇒ Modellierung einer reduzierten Glaubwürdigkeit in der Ähnlichkeitsmatrix nicht möglich
- Daher Auswahl der besten Match Combination die alle (harten) Bedingungen erfüllt bzw. die geringsten Kosten hat (⇒ Anwendung nach oder kombiniert mit Match Selector)
- Frage: Wie bestimmt man die Güte einer Match Combination?
- Einfache Lösung: Annahme das alle Confidence-Werte unabhängig voneinander bestimmt wurden
- ⇒ Multiplizieren der Confidence-Werte aller Korrespondenzen in der Match Combination

Beispiel: Güte von Match Combinations [DHI12]

average-combiner

name = \langle name: 0.6, title: 0.5 \rangle

releaseInfo = \langle releaseDate: 0.6, releaseCompany: 0.25 \rangle

classification = \langle rating: 0.3 \rangle

price = \langle basePrice: 0.5 \rangle

Beispiel: Güte von Match Combinations [DHI12]

average-combiner

name = \langle name: 0.6, title: 0.5 \rangle

releaseInfo = \langle releaseDate: 0.6, releaseCompany: 0.25 \rangle

classification = \langle rating: 0.3 \rangle

price = \langle basePrice: 0.5 \rangle

- Es gibt vier Match Combinations $M_1 - M_4$
Beispiel: $M_1 = \{ \text{name} = \text{name}, \text{releaseInfo} = \text{releaseDate}, \text{classification} = \text{rating}, \text{price} = \text{basePrice} \}$

Beispiel: Güte von Match Combinations [DHI12]

average-combiner

name = \langle name: 0.6, title: 0.5 \rangle

releaseInfo = \langle releaseDate: 0.6, releaseCompany: 0.25 \rangle

classification = \langle rating: 0.3 \rangle

price = \langle basePrice: 0.5 \rangle

- Es gibt vier Match Combinations $M_1 - M_4$
Beispiel: $M_1 = \{ \text{name} = \text{name}, \text{releaseInfo} = \text{releaseDate}, \text{classification} = \text{rating}, \text{price} = \text{basePrice} \}$
- Für jede Match Combination M_i berechnen wird die Güte indem wir die individuellen Confidence-Werte multiplizieren
Beispiel: $\text{score}(M_1) = 0.6 \times 0.6 \times 0.3 \times 0.5$

Beispiel: Korrespondenz übergreifende Bedingung [DHI12]

BOOK-VENDOR

Books(ISBN, publisher, pubCountry, title, review)

Inventory(ISBN, quantity, location)

DISTRIBUTOR

Items(code, name, brand, origin, desc)

InStore(code, availQuant)

	Constraints	Costs
c_1	If $A = \text{Items.code}$, then A is a key	∞
c_2	If $A = \text{Items.desc}$, then any random sample of 100 data instances of A must have an average length of at least 20 words	1.5
c_3	If $A_1 = B_1, A_2 = B_2, B_2$ is next to B_1 in the schema, but A_2 is not next to A_1 , then there is no A^* next to A_1 such that $ \text{sim}(A^*, B_2) - \text{sim}(A_2, B_2) \leq t$ for a small pre-specified t	2
c_4	If more than half of the attributes of Table U match those of Table V , then $U = V$	1

Einbinden von Integritätsbedingungen

Einbinden von Integritätsbedingungen

- Der Enforcer muss für jede Match Combination entscheiden können ob diese die gegebenen Bedingungen erfüllt

Einbinden von Integritätsbedingungen

- Der Enforcer muss für jede Match Combination entscheiden können ob diese die gegebenen Bedingungen erfüllt
- **Problem:** Auch wenn eine Bedingung durch die gegebene Instanz nicht verletzt ist, muss nicht gelten, dass die Bedingung nicht durch eine zukünftige Instanz verletzt wird

Einbinden von Integritätsbedingungen

- Der Enforcer muss für jede Match Combination entscheiden können ob diese die gegebenen Bedingungen erfüllt
- **Problem:** Auch wenn eine Bedingung durch die gegebene Instanz nicht verletzt ist, muss nicht gelten, dass die Bedingung nicht durch eine zukünftige Instanz verletzt wird
- **Beispiel:** Wenn alle Werte eines Attributes verschieden sind, muss dies nicht bedeuten, dass dieses Attribut ein Schlüsselkandidat ist

Einbinden von Integritätsbedingungen

- Der Enforcer muss für jede Match Combination entscheiden können ob diese die gegebenen Bedingungen erfüllt
- **Problem:** Auch wenn eine Bedingung durch die gegebene Instanz nicht verletzt ist, muss nicht gelten, dass die Bedingung nicht durch eine zukünftige Instanz verletzt wird
- **Beispiel:** Wenn alle Werte eines Attributes verschieden sind, muss dies nicht bedeuten, dass dieses Attribut ein Schlüsselkandidat ist
- Gegebene Quellinstanzen reichen meistens aber aus um schnell Verletzungen von Integritätsbedingungen zu entdecken

Einbinden von Integritätsbedingungen

Einbinden von Integritätsbedingungen

- Naiver Ansatz:
 - Sortiere alle Match Combinations nach ihrer Güte
 - Iteriere über die sortierte Liste bis eine Match Combination alle Bedingungen erfüllt

Einbinden von Integritätsbedingungen

- Naiver Ansatz:
 - Sortiere alle Match Combinations nach ihrer Güte
 - Iteriere über die sortierte Liste bis eine Match Combination alle Bedingungen erfüllt
- In der Praxis meist zu ineffizient
 - Auswertung von Bedingungen kann komplex sein
 - Anzahl an Match Combinations ist meist sehr groß
(\Rightarrow Die Güte kann nicht für jede berechnet werden)

Einbinden von Integritätsbedingungen

- Naiver Ansatz:
 - Sortiere alle Match Combinations nach ihrer Güte
 - Iteriere über die sortierte Liste bis eine Match Combination alle Bedingungen erfüllt
- In der Praxis meist zu ineffizient
 - Auswertung von Bedingungen kann komplex sein
 - Anzahl an Match Combinations ist meist sehr groß
(\Rightarrow Die Güte kann nicht für jede berechnet werden)
- Zudem: Keine Kosten von schwachen Bedingungen berücksichtigt

Einbinden von Integritätsbedingungen

- Naiver Ansatz:
 - Sortiere alle Match Combinations nach ihrer Güte
 - Iteriere über die sortierte Liste bis eine Match Combination alle Bedingungen erfüllt
- In der Praxis meist zu ineffizient
 - Auswertung von Bedingungen kann komplex sein
 - Anzahl an Match Combinations ist meist sehr groß
(\Rightarrow Die Güte kann nicht für jede berechnet werden)
- Zudem: Keine Kosten von schwachen Bedingungen berücksichtigt
- Effizientere Methoden benötigt (z.B. A*-Suche)

A*-Suche

A*-Suche

- Die A*-Suche lässt sich als wachsender Baum visualisieren
 - Wurzel = Startzustand
 - einige Blätter sind Zielzustand
 - Jeder Pfad ist mit Kosten versehen

A*-Suche

- Die A*-Suche lässt sich als wachsender Baum visualisieren
 - Wurzel = Startzustand
 - einige Blätter sind Zielzustand
 - Jeder Pfad ist mit Kosten versehen
- Die A*-Suche findet ausgehend vom Startzustand den Endzustand mit den minimalen Kosten (garantiert)

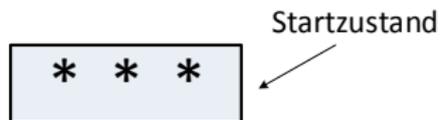
A*-Suche

- Die A*-Suche lässt sich als wachsender Baum visualisieren
 - Wurzel = Startzustand
 - einige Blätter sind Zielzustand
 - Jeder Pfad ist mit Kosten versehen
- Die A*-Suche findet ausgehend vom Startzustand den Endzustand mit den minimalen Kosten (garantiert)
- **Bestensuche:** Solange der aktuell kostengünstigste Zustand kein Endzustand ist:
 - Wähle den aktuell kostengünstigste Zustand
 - Expandiere den gewählten Zustand in eine Menge neuer Zustände

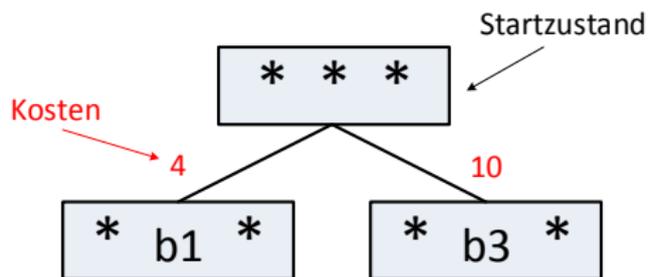
A*-Suche

- Die A*-Suche lässt sich als wachsender Baum visualisieren
 - Wurzel = Startzustand
 - einige Blätter sind Zielzustand
 - Jeder Pfad ist mit Kosten versehen
- Die A*-Suche findet ausgehend vom Startzustand den Endzustand mit den minimalen Kosten (garantiert)
- **Bestensuche:** Solange der aktuell kostengünstigste Zustand kein Endzustand ist:
 - Wähle den aktuell kostengünstigste Zustand
 - Expandiere den gewählten Zustand in eine Menge neuer Zustände
- Kosten eines Zustandes $Z = \text{Kosten des Pfades von Startzustand zu } Z + \text{lower bound für die Kosten des günstigen Pfades von } Z \text{ zu einem Zielzustand}$

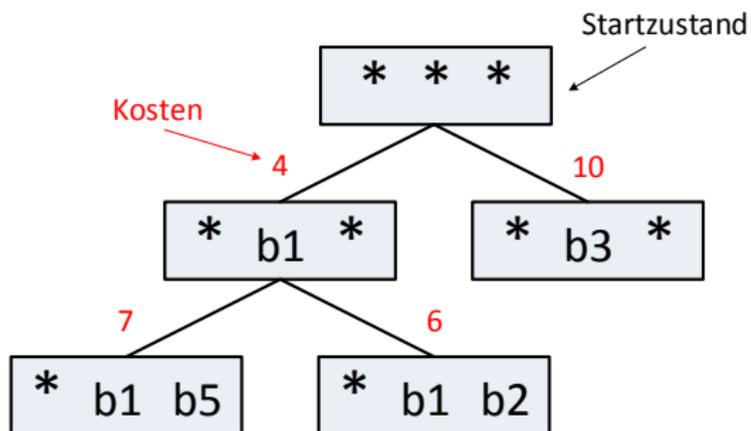
Beispiel: A*-Suche



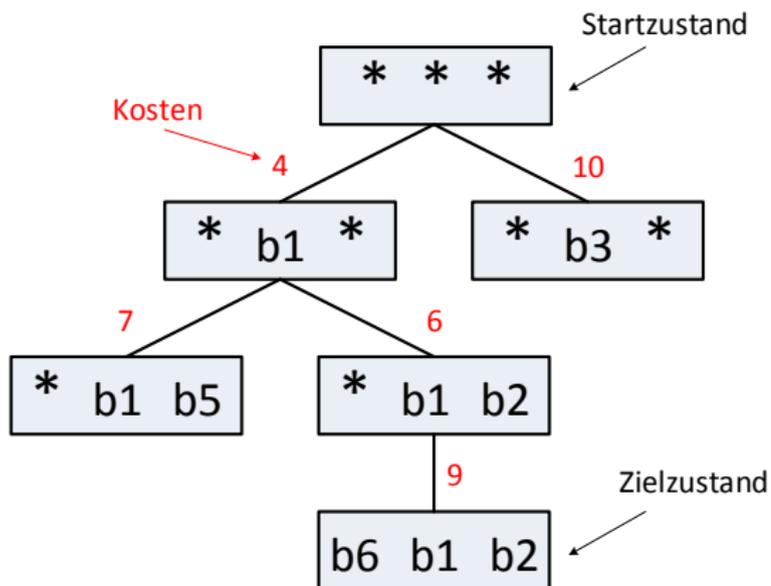
Beispiel: A*-Suche



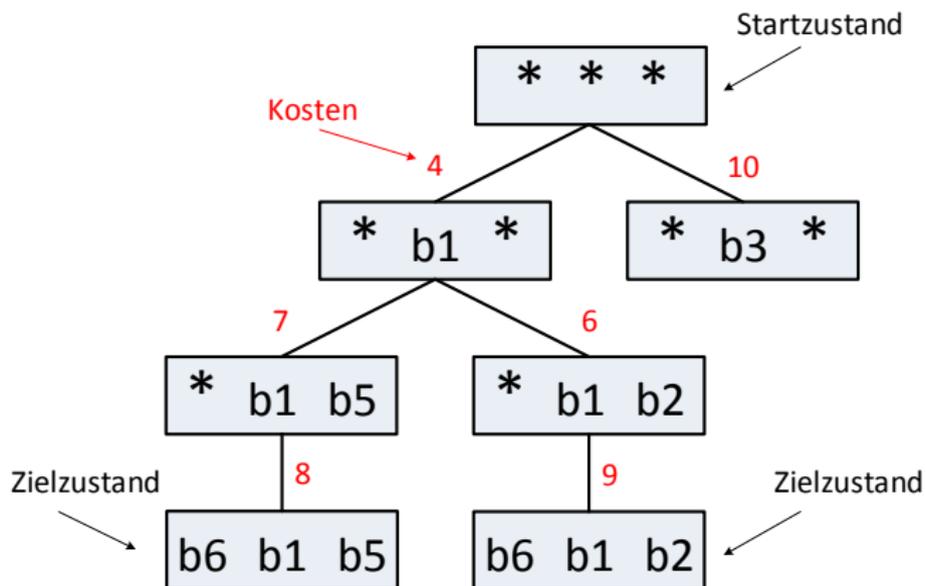
Beispiel: A*-Suche



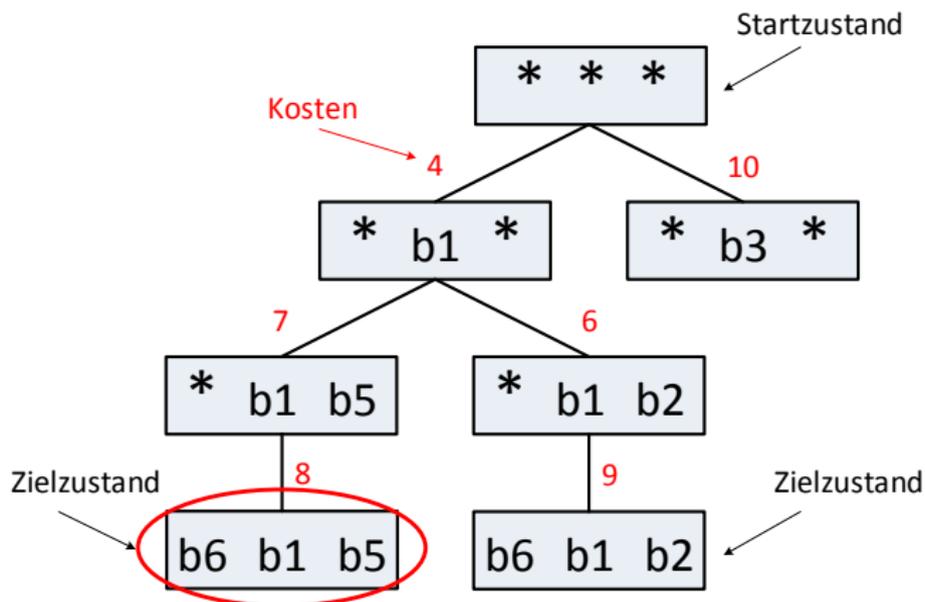
Beispiel: A*-Suche



Beispiel: A*-Suche



Beispiel: A*-Suche



A*-Suche bzgl. Integritätsbedingungen

A*-Suche bzgl. Integritätsbedingungen

- Ziel: A*-Suche um zwei Schemata \mathcal{A} und \mathcal{B} zu matchen
 - \mathcal{A} hat Elemente $\{A_1, \dots, A_n\}$
 - \mathcal{B} hat Elemente $\{B_1, \dots, B_m\}$

A*-Suche bzgl. Integritätsbedingungen

- Ziel: A*-Suche um zwei Schemata \mathcal{A} und \mathcal{B} zu matchen
 - \mathcal{A} hat Elemente $\{A_1, \dots, A_n\}$
 - \mathcal{B} hat Elemente $\{B_1, \dots, B_m\}$
- Jeder Zustand entspricht einem n -tuple
(eine Position pro Element in \mathcal{A})

A*-Suche bzgl. Integritätsbedingungen

- Ziel: A*-Suche um zwei Schemata \mathcal{A} und \mathcal{B} zu matchen
 - \mathcal{A} hat Elemente $\{A_1, \dots, A_n\}$
 - \mathcal{B} hat Elemente $\{B_1, \dots, B_m\}$
- Jeder Zustand entspricht einem n -tuple
(eine Position pro Element in \mathcal{A})
- Wert an Position i ist entweder
 - eine Korrespondenz für A_i (also ein Element in \mathcal{B})
 - einen Nullwert ' \perp '
(es gibt zu A_i kein korrespondierendes Element in \mathcal{B})
 - oder eine Wildcard '*'
(die Korrespondenz ist noch nicht spezifiziert)

A*-Suche bzgl. Integritätsbedingungen

A*-Suche bzgl. Integritätsbedingungen

⇒ Jeder Zustand repräsentiert eine Menge an Match Combinations

A*-Suche bzgl. Integritätsbedingungen

- ⇒ Jeder Zustand repräsentiert eine Menge an Match Combinations
 - Der Startzustand enthält nur Wildcards, d.h. $(*, *, \dots, *)$
(entspricht allen möglichen Match Combinations)

A*-Suche bzgl. Integritätsbedingungen

- ⇒ Jeder Zustand repräsentiert eine Menge an Match Combinations
- Der Startzustand enthält nur Wildcards, d.h. $(*, *, \dots, *)$
(entspricht allen möglichen Match Combinations)
 - Ein Zielzustand enthält keine Wildcards
(entspricht einer einzigen Match Combination)

A*-Suche bzgl. Integritätsbedingungen

- ⇒ Jeder Zustand repräsentiert eine Menge an Match Combinations
- Der Startzustand enthält nur Wildcards, d.h. $(*, *, \dots, *)$
(entspricht allen möglichen Match Combinations)
 - Ein Zielzustand enthält keine Wildcards
(entspricht einer einzigen Match Combination)
 - Expansion eines Zustandes
 - Wähle eine Wildcard an Position i
 - Erzeuge einen Zustand für jede mögliche Korrespondenz von A_i

A*-Suche bzgl. Integritätsbedingungen

- ⇒ Jeder Zustand repräsentiert eine Menge an Match Combinations
- Der Startzustand enthält nur Wildcards, d.h. $(*, *, \dots, *)$
(entspricht allen möglichen Match Combinations)
 - Ein Zielzustand enthält keine Wildcards
(entspricht einer einzigen Match Combination)
 - Expansion eines Zustandes
 - Wähle eine Wildcard an Position i
 - Erzeuge einen Zustand für jede mögliche Korrespondenz von A_i
 - Auswahl der zu expandierenden Wildcard beeinflusst Effizienz

A*-Suche bzgl. Integritätsbedingungen

A*-Suche bzgl. Integritätsbedingungen

- **Grobes Ziel:** Beste Match Combination mit den geringsten (durch verletzte Bedingungen verursachten) Kosten

A*-Suche bzgl. Integritätsbedingungen

- **Grobes Ziel:** Beste Match Combination mit den geringsten (durch verletzte Bedingungen verursachten) Kosten
- **Problem:** A*-Suche berücksichtigt nur Kosten und keine Güte

A*-Suche bzgl. Integritätsbedingungen

- **Grobes Ziel:** Beste Match Combination mit den geringsten (durch verletzte Bedingungen verursachten) Kosten
- **Problem:** A*-Suche berücksichtigt nur Kosten und keine Güte
- Die von verletzten Bedingungen verursachten Kosten spiegeln allerdings auch eine Art von Güte wieder

A*-Suche bzgl. Integritätsbedingungen

- **Grobes Ziel:** Beste Match Combination mit den geringsten (durch verletzte Bedingungen verursachten) Kosten
 - **Problem:** A*-Suche berücksichtigt nur Kosten und keine Güte
 - Die von verletzten Bedingungen verursachten Kosten spiegeln allerdings auch eine Art von Güte wieder
- ⇒ **Kosten eines Pfades:** Kombination von Güte des erreichten Zustandes und Kosten verursacht durch die Bedingungen die durch den Zustand verletzt werden (Wildcards werden ignoriert)

Schema Matching – Weitere Anwendungen

- Herkömmlich: Korrespondenzen finden
- Schlüssel - Fremdschlüssel Beziehungen finden
(ähnliche Attribute innerhalb eines Schemas sind gute Kandidaten)
- Höher-stufige Korrespondenzen finden
(Ähnlichkeiten von Tabellen durch Aggregation der Matches ihrer Attribute)

Agenda

- ① Schema Matching
 - Problemstellung
 - Motivation

- ② Schema Matching Systeme
 - Label-based Matcher
 - Instance-based Matcher
 - Structure-based Matcher
 - Combiner
 - Match Selector
 - Constraint Enforcer
 - n:m Korrespondenzen

n:m Korrespondenzen

- Bisher nur 1:1 Korrespondenzen

n:m Korrespondenzen

- Bisher nur 1:1 Korrespondenzen
- Schemata sind aber oft so heterogen, dass n:m Korrespondenzen notwendig sind

n:m Korrespondenzen

- Bisher nur 1:1 Korrespondenzen
- Schemata sind aber oft so heterogen, dass n:m Korrespondenzen notwendig sind
- Beispiel 1:

SOURCE

Books(title, basePrice, taxRate, quantity, authorFirstName, authorLastName)

TARGET

Items(title, price, inventory, author, genre)

n:m Korrespondenzen

- Bisher nur 1:1 Korrespondenzen
- Schemata sind aber oft so heterogen, dass n:m Korrespondenzen notwendig sind
- Beispiel 1:

SOURCE

Books(title, basePrice, taxRate, quantity, authorFirstName, authorLastName)

TARGET

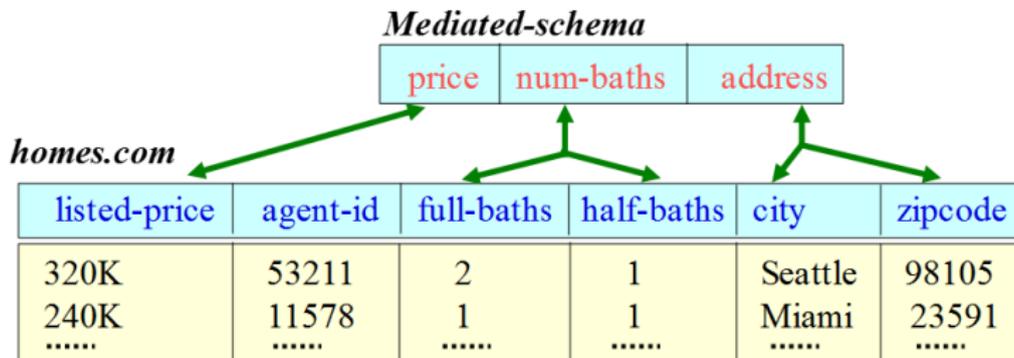
Items(title, price, inventory, author, genre)

n:m Korrespondenzen:

- author = concat(authorFirstName, authorLastName)
- price = basePrice \times (1 + taxRate)

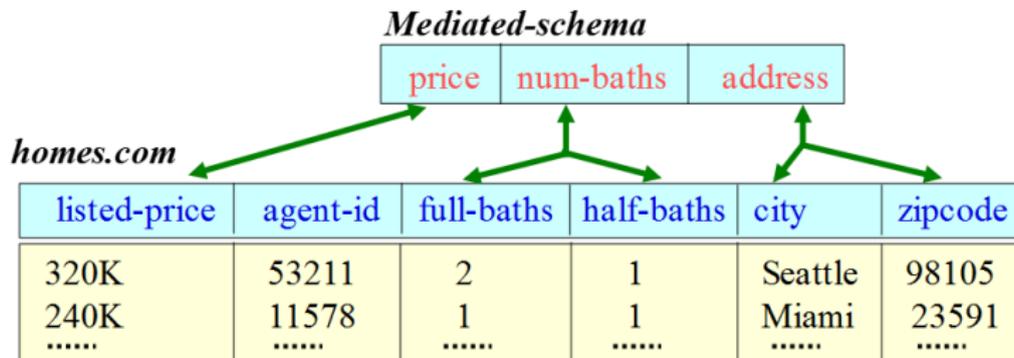
n:m Korrespondenzen (Beispiel)

- Beispiel 2:



n:m Korrespondenzen (Beispiel)

- Beispiel 2:



n:m Korrespondenzen:

- num-baths = full-baths + half-baths
- address = concat(city, zipcode)

n:m Korrespondenzen

- **Problem:** Man kann nicht alle möglichen Kombinationen testen (evtl. unendliche viele)

n:m Korrespondenzen

- **Problem:** Man kann nicht alle möglichen Kombinationen testen (evtl. unendliche viele)
- Verwendung spezieller *Searcher*, die für bestimmte Datentypen und Domänen konzipiert sind
 - **Text searcher:** Konkatenationen von Elementen
 - **Numeric searcher:** Arithmetische Ausdrücke, Verwendung von häufig auftretenden Konstanten (z.B. zur Währungsumrechnung)
 - **Date searcher:** Kombinationen von Monat/Jahr/Tag

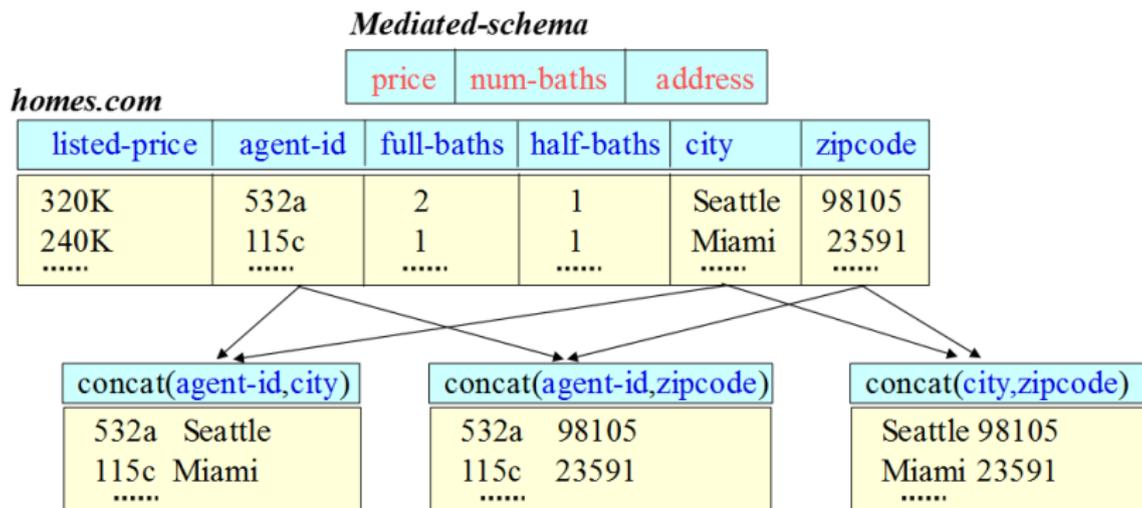
n:m Korrespondenzen

- **Problem:** Man kann nicht alle möglichen Kombinationen testen (evtl. unendliche viele)
- Verwendung spezieller *Searcher*, die für bestimmte Datentypen und Domänen konzipiert sind
 - **Text searcher:** Konkatenationen von Elementen
 - **Numeric searcher:** Arithmetische Ausdrücke, Verwendung von häufig auftretenden Konstanten (z.B. zur Währungsumrechnung)
 - **Date searcher:** Kombinationen von Monat/Jahr/Tag
- Testen von Kombinationen speziell mit Instanzbasierten Matchern sinnvoll

n:m Korrespondenzen

- **Problem:** Man kann nicht alle möglichen Kombinationen testen (evtl. unendliche viele)
- Verwendung spezieller *Searcher*, die für bestimmte Datentypen und Domänen konzipiert sind
 - **Text searcher:** Konkatenationen von Elementen
 - **Numeric searcher:** Arithmetische Ausdrücke, Verwendung von häufig auftretenden Konstanten (z.B. zur Währungsumrechnung)
 - **Date searcher:** Kombinationen von Monat/Jahr/Tag
- Testen von Kombinationen speziell mit Instanzbasierten Matchern sinnvoll
- **A*-Suche:** Auswahl einer möglichen Korrespondenz für disjunkte Elementmengen $\mathcal{A}' \subseteq \mathcal{A}$ anstatt eines einzelnen Elementes von \mathcal{B} pro Element in \mathcal{A}

Searcher (Beispiel)



- Beste Match-Kandidaten für das Attribut *address*:
 - (agent-id,0.7)
 - (concat(agent-id, city),0.75)
 - (concat(city, zipcode),0.9)

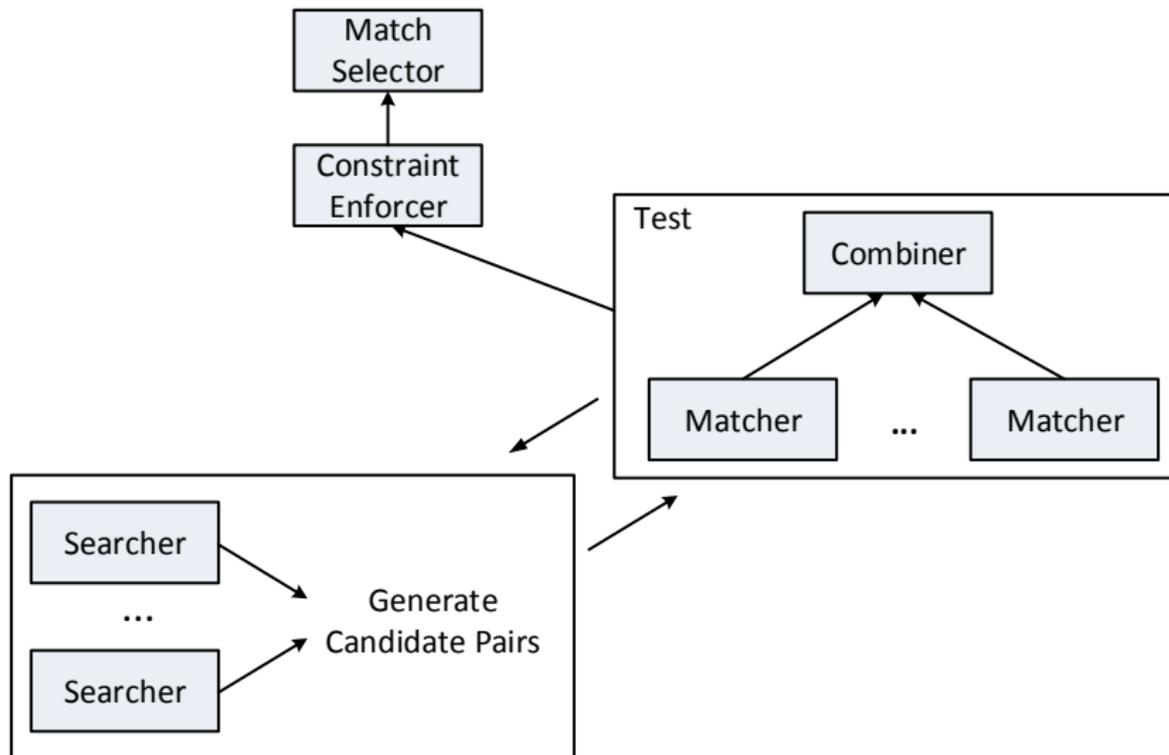
Searcher

Searcher besteht aus drei Komponenten

- **Suchstrategie:** Bestimmt welche Elemente mit welchen Funktionen verknüpft werden und nach welchem Konzept die Suche verläuft (genetische Ansätze denkbar)
- **Evaluation:** Berechnung einer Ähnlichkeit (wie bisher) aber Searcher können bestimmte Matcher diktieren
- **Termination:** Stoppkriterium um die (wohlmöglich sonst endlose) Suche zu beenden

Beispiel: Stoppe die Suche wenn die bis dato maximale gefundene Ähnlichkeit bei den letzten 100 getesteten Kombinationen nicht mehr gestiegen ist

Modifizierte Matching Architektur



Zusammenfassung Schema Matching

- Schema Matching basierend auf
 - Namen der Schemaelemente (*label-based*)
 - Darunterliegende Daten (*instance-based*)
 - Struktur des Schemas (*structure-based*)
- Kombination von verschiedenen Matching Ergebnissen
- Einbinden von Integritätsbedingung
- Auswahl einer finalen Match Combination
- Suche nach n:m Korrespondenzen

Literatur I

- [BCH⁺05] Angela Bonifati, Qing (Elaine) Chang, Terence Ho, Laks V.S. Lakshmanan, and Rachel Pottinger.
HePToX: Marrying XML and heterogeneity in your P2P databases.
In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, 2005.
Demo paper.
- [BN05] Alexander Bilke and Felix Naumann.
Schema matching using duplicates.
In *Proc. of the Int. Conf. on Data Engineering (ICDE)*, 2005.
- [DHI12] Anhai Doan, Alon Halevy, and Zachary Ives.
Principles of Data Integration.
Morgan Kaufmann, 2012.
- [MMR02] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm.
Similarity Flooding: A versatile graph matching algorithm and its application to schema matching.
In *Proc. of the Int. Conf. on Data Engineering (ICDE)*, 2002.

Literatur II

- [NHT⁺02] Felix Naumann, Ching-Tien Ho, Xuqing Tian, Laura Haas, and Nimrod Megiddo.
Attribute classification using feature analysis.
In *Proc. of the Int. Conf. on Data Engineering (ICDE)*, 2002.
- [RB01] E. Rahm and P.A. Bernstein.
survey of approaches to automatic schema matching.
The VLDB Journal, 10(4):334-350, 2001.