Informationsintegration

Schema Matching

Ulf Leser

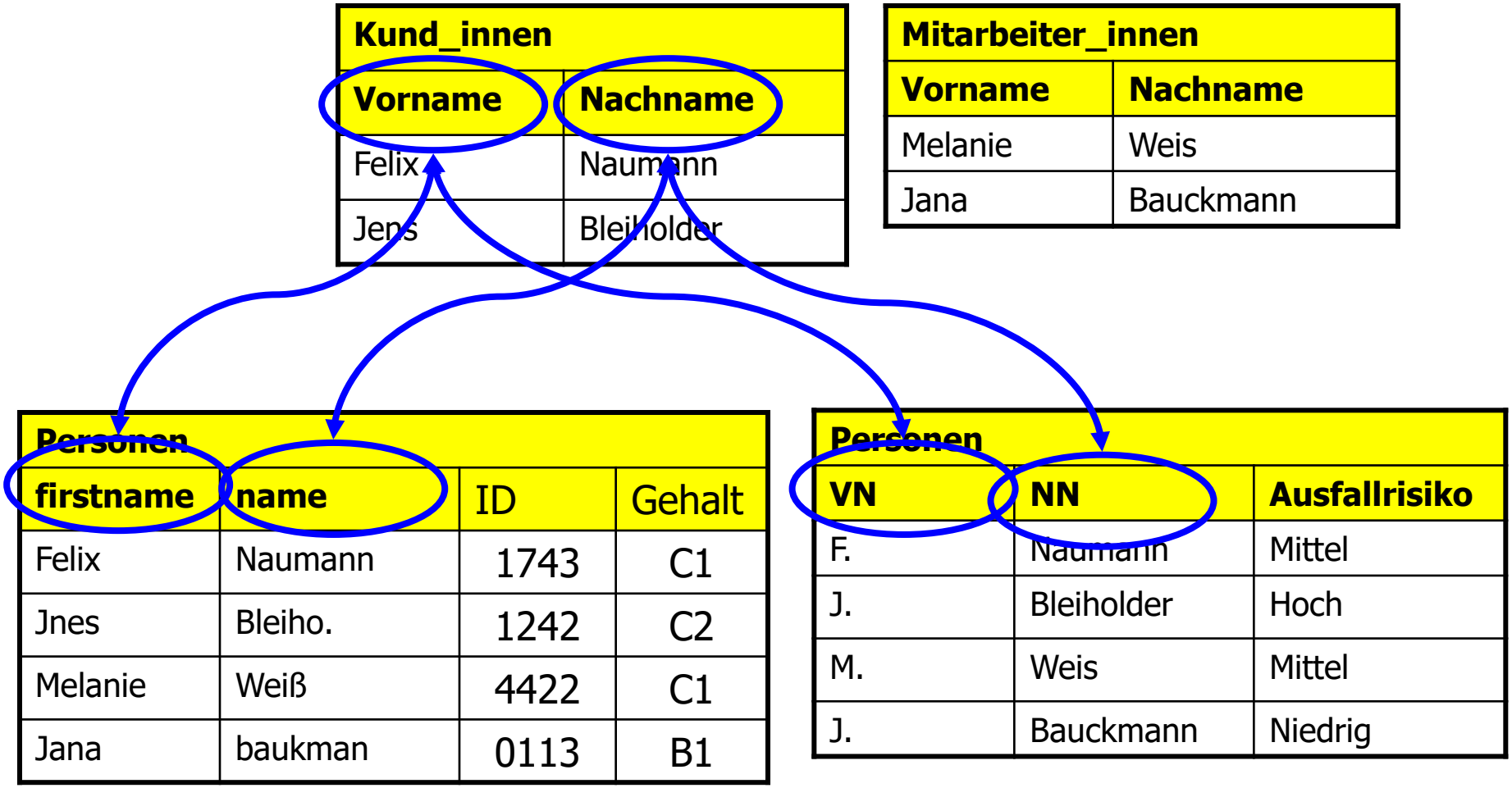
Inhalt dieser Vorlesung

- Schema Matching
- Matching Attributes
  - Labelbasiert
  - Instanzbasiert
- Matching schemata
  - Global matching: Gale-Shapely
  - From attributed to relations
  - Matching schemata
- Varianten

Schema Matching

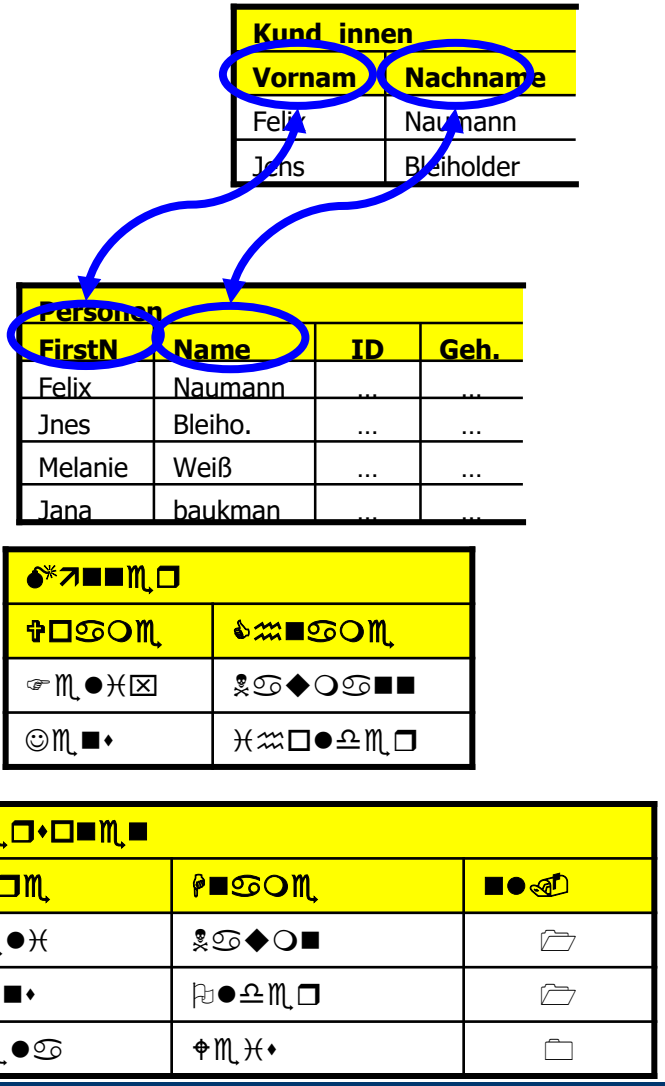
- Korrespondenzen müssen irgendwo herkommen
- Finden von Korrespondenzen ist nicht trivial
  - Große Schemata: Viele Tabellen und Attribute
  - Generische, automatisch erzeugte Namen
  - Keine oder fehlerhaft umgesetzte Namenskonventionen
  - Fehlende Dokumentation
  - „Ungewöhnliche" Schemata: Denormalisierung, Redundanzen etc.
  - Semantische Heterogenität: Synonyme, Homonyme, …
  - Schematische Heterogenität
- Möglichkeiten
  - Manuelle Erstellung: Teuer, fehleranfällig, zeitraubend
  - Schema Matching: Automatisches Finden äquivalenter Elemente

Beispiele


Kund_innen
Vorname
Nachname
Felix
Naumann
Jens
Bleiholder
Mitarbeiter_innen
Vorname
Nachname
Melanie
Weis
Jana
Bauckmann
Personen
firstname
name
ID
Gehalt
Felix
Naumann
1743
C1
Jnes
Bleiho.
1242
C2
Melanie
Weiß
4422
C1
Jana
baukman
0113
B1
Personen
VN
NN
Ausfallrisiko
F.
Naumann
Mittel
J.
Bleiholder
Hoch
M.
Weis
Mittel
J.
Bauckmann
Niedrig

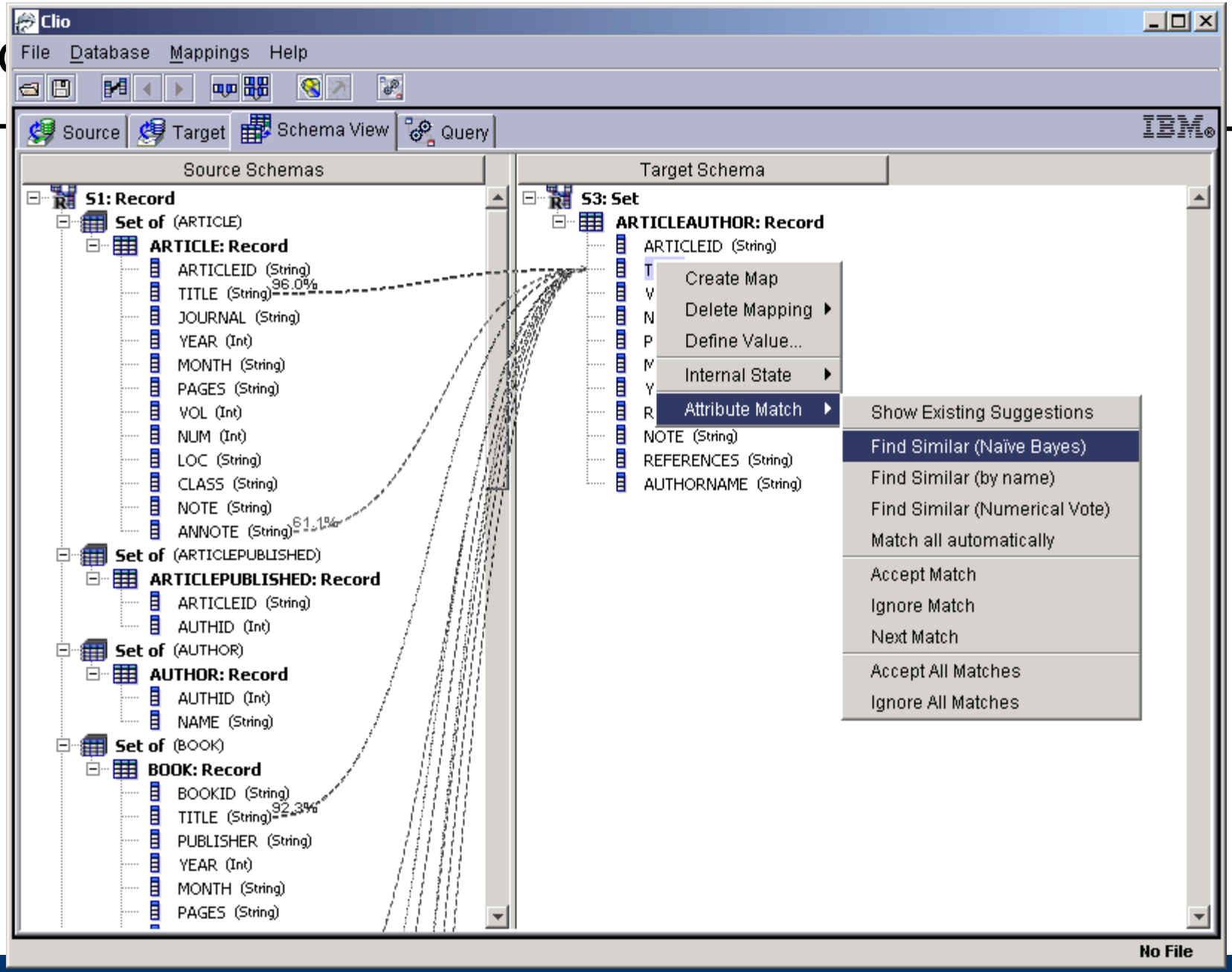
Woher wissen wir das?

- Attribute haben ähnliche Namen
- Ähnlicher Tabellenkontext
- Werte sehen nach Vor- oder Nachnamen aus
- Werte sehen ähnlich aus
- Wo Vornamen sind, sind meist auch Nachnamen
- Personen haben Vor- und Nachnamen
- Mitarbeiter_innen sind Personen
- Vieles davon erschließt sich dem Computer nicht ohne weiteres



Verfahren

- Wir betrachten die folgenden Verfahren näher
  - Labelbasiert: Verwendung der Intension
    - Namen der Schemaelemente
  - Instanzbasiert: Verwendung der Extension
    - Eigentliche Daten
  - Duplikatbasiert: Gleiche Tupel zeigen auf gleiche Stuktur
  - Strukturbasiert: Ausnutzung der Struktur der Schemata
- Aktuelle Schema Matcher verwenden alle Methoden und noch einige mehr
  - Hinzuziehen von Hintergrundwissen (Ontologien)
  - Linguistische Analyse langer Strings (Texte)


Clio
File Database Mappings Help
Source Target Schema View Query
IBM
Source Schemas
S1: Record
Set of (ARTICLE)
ARTICLE: Record
ARTICLEID (String)
TITLE (String)
96.0%
JOURNAL (String)
YEAR (Int)
MONTH (String)
PAGES (String)
VOL (Int)
NUM (Int)
LOC (String)
CLASS (String)
NOTE (String)
ANNOTE (String)
61.1%
Set of (ARTICLEPUBLISHED)
ARTICLEPUBLISHED: Record
ARTICLEID (String)
AUTHID (Int)
Set of (AUTHOR)
AUTHOR: Record
AUTHID (Int)
NAME (String)
Set of (BOOK)
BOOK: Record
BOOKID (String)
TITLE (String)
92.3%
PUBLISHER (String)
YEAR (Int)
MONTH (String)
PAGES (String)
Target Schema
S3: Set
ARTICLEAUTHOR: Record
ARTICLEID (String)
NOTE (String)
REFERENCES (String)
AUTHORNAME (String)
Create Map
Delete Mapping
Define Value...
Internal State
Attribute Match
Show Existing Suggestions
Find Similar (Naïve Bayes)
Find Similar (by name)
Find Similar (Numerical Vote)
Match all automatically
Accept Match
Ignore Match
Next Match
Accept All Matches
Ignore All Matches
No File

Inhalt dieser Vorlesung

- Schema Matching
- Matching Attributes
  - Labelbasiert
  - Instanzbasiert
- Matching schemata
- Varianten

Labelbasiertes Schema Matching

- Geg. zwei Schemata mit Attributmengen A und B
  - A={ID, Name, Vorname, Alter}, B={No, Name, First_name, Age}
- Kernidee
  - Bilde Kreuzprodukt aller Attribute aus A und B
  - Für jedes Paar berechne die Ähnlichkeit der Attributnamen
    - Z.B. Editabstand, Jaro-Winkler, …
  - Die ähnlichsten Paare sind die Matches

	ID	Name	Vorname	Alter
No				
Name				
First_name				
Age				

Beispiel 2

- Geg. zwei Schemata mit Attributmengen A und B
  - A={ID, Name, Vorname, Alter}, B={No, Name, First_name, Age}
- Kernidee
  - Bilde Kreuzprodukt aller Attribute aus A und B
  - Für jedes Paar berechne die Ähnlichkeit der Attributnamen
    - Z.B. Editabstand, Jaro-Winkler, …
  - Die ähnlichsten Paare sind die Matches

	ID	Name	Vorname	Alter
No	2	3	6	5
Name	4	0	3	4
First_name	9	6	5	8
Age	3	3	5	3

Beispiel 3

- Geg. zwei Schemata mit Attributmengen A und B
  - A={ID, Name, Vorname, Alter}, B={No, Name, First_name, Age}
- Kernidee
  - Bilde Kreuzprodukt aller Attribute aus A und B
  - Für jedes Paar berechne die Ähnlichkeit der Attributnamen
    - Z.B. Editabstand, Jaro-Winkler, …
  - Die ähnlichsten Paare sind die Matches

	ID	Name	Vorname	Alter
No	2	3	6	5
Name	4	0	3	4
First_name	9	6	5	8
Age	3	3	5	3

Beispiel 4

- Geg. zwei Schemata mit Attributmengen A und B
  - A={ID, Name, Vorname, Alter}, B={No, Name, First_name, Age}
- Kernidee
  - Bilde Kreuzprodukt aller Attribute aus A und B
  - Für jedes Paar berechne die Ähnlichkeit der Attributnamen
    - Z.B. Editabstand, Jaro-Winkler, …
  - Die ähnlichsten Paare sind die Matches

	ID	Name	Vorname	Alter
No	2	3	6	5
Name	4	0	3	4
First_name	9	6	5	8
Age	3	3	5	3

Probleme

- Ambiguität: Mehrere gleich gute Matches
  - Gerade bei kurzen Labels
- Asymmetrie: Keine eineindeutige Zuordnung
- Fazit: Begrenzte Qualität
  - Umso schlechter, je weniger „sprechend" Attributnamen sind
    - Häufig sehr kurz – sparen Schreibarbeit
    - Abkürzungen, Fremdsprachen, zusammengesetzte Namen
    - id_pers_abt
  - Aber liefert Evidenzen – muss man kombinieren

Instanzbasiertes Schema Matching

- Geg. Instanzen zweier Schemata mit Attributen A und B
- Kernidee
  - Für jedes Attribut: Extrahiere Eigenschaften seiner Werte
    - Beispielwerte, maximale Werte, Durchschnitt, Buchstabenhäufigkeiten, Länge, statistische Eigenschaften, etc.
  - Bilde Kreuzprodukt aller Attribute aus A und B
  - Berechne Ähnlichkeit der Attribute als Ähnlichkeit der Eigenschaften
- Warum nicht Jaccard über Wertemengen?
  - Gesucht: Gleiche Intension, nicht gleiche Extension
  - Intension über Eigenschaften annähern
  - Trotzdem sinnvoll, wenn Werte klare Bedeutung haben
    - Integer: Eher nein
    - Produktnamen: Vielleicht ja

Beispiel

ID	Name	Whg
1	Müller	Danziger Str, Berlin
2	Meyer	Boxhagenerstr, Berlin
4	Schmidt	Turmstr, Köln

Nr	Adresse	Telefon
1	Seeweg, Berlin	030-3324566
3	Aalstr, Schwedt	0330-1247765
4	Rosenallee, Kochel	0884-334621

- Eigenschaften: Datentyp, durchschnittliche Länge
  - A = {(NUM, 1), (STRING, 6), (STRING, 18)}
  - B = {(NUM, 1), (STRING, 16), (STRING, 11)}
- Ähnlichkeit: Euklidischer Abstand d (NUM=0, STR=1)

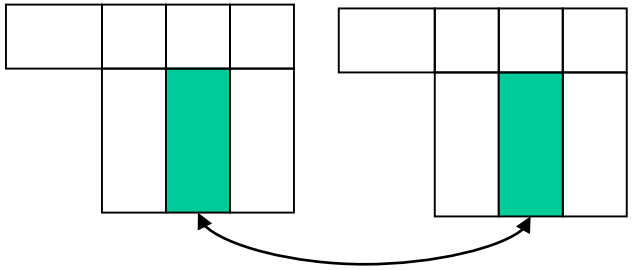
	ID	Name	Whg
Nr	d(<0,1>,<0,1>)	d(<1,6>,<0,1>)	d(<1,18>,<0,1>)
Adresse	d(<0,1>,<1,16>)	d(<1,6>,<1,16>)	d(<1,18>,<1,16>)
Telefon	d(<0,1>,<1,11>)	d(<1,6>,<1,11>)	d(<1,18>,<1,11>)

Probleme

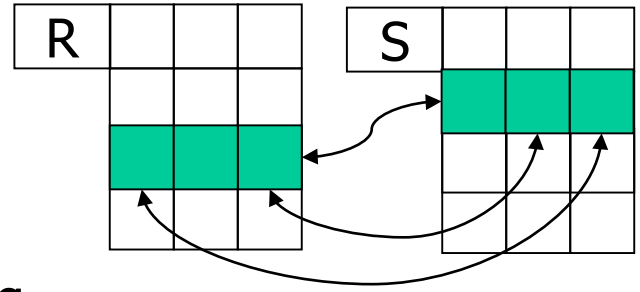
- Welche Eigenschaften wählen?
- Wie sollen Eigenschaften verglichen werden?
  - Typisch: Darstellen der Eigenschaften als Vektor
  - Ähnlichkeitsmaße für Vektoren: Winkel, Euklidischer Abstand, …
- Gleiche Eigenschaften ≠ gleiche Intension
  - Telefonnummern / Faxnummer
  - Namen von Mitarbeitern, Namen von Kunden
  - Surrogate-Keys <integer>
- Fazit: Funktioniert manchmal, manchmal nicht

Variante: Duplikatbasiertes Matching [BN05]

- Bisheriges instanzbasiertes Matching ist vertikal
  - Vergleich kompletter Spalten
  - Ignoriert die Zusammengehörigkeit von Werten zu Tupeln



- Horizontales instanzbasiertes Matching
  - Vergleicht paarweise alle Tupel
  - Finde (einige) potentielle Duplikate
  - In den Duplikaten müssen die einzelnen Attributwerte nahezu identisch sein
  - Ergibt ein Attributmatching pro Duplikat
  - Finales Matching z.B. durch Majority Voting über matchende Tupel-Paare



Beispiel

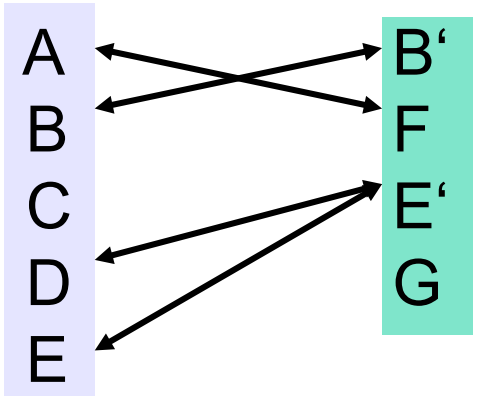
A	B	C	D	E
Max	Michel	m	601- 4839204	601- 4839204
...	...	..	...	...

?

?

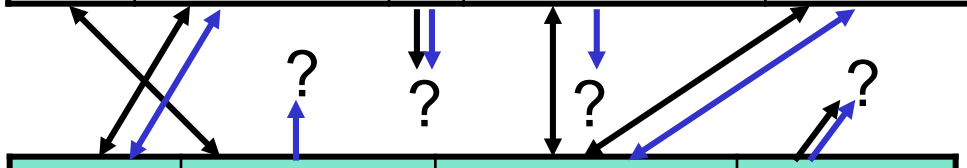
B‘	F	E‘	G
Michel	maxm	601- 4839204	UNIX
...	...	...	...

- Matching nach einem Duplikat



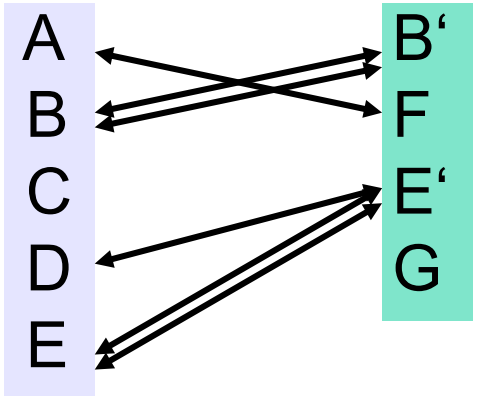
Beispiel 2

A	B	C	D	E
Max	Michel	m	601- 4839204	601- 4839204
Sam	Adams	m	541- 8127100	541- 8121164



B‘	F	E‘	G
Michel	maxm	601- 4839204	UNIX
Adams	samuel	541- 8127164	WinXP

Matching nach zwei Duplikaten



Probleme

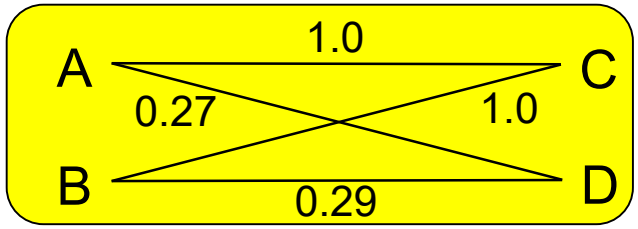
- Funktioniert nur, wenn Duplikate vorhanden sind und gefunden werden
  - Das ist starke Einschränkung
- Schwierigkeiten bei struktureller Heterogenität
  - Gleiche Attribute unterschiedlich auf Relationen verteilt
- Duplikaterkennung setzt idR Korrespondenzen voraus
  - Henne – Ei Problem (siehe Ähnlichkeitsfunktionen für Tupel)
- Duplikaterkennung ist teuer (später)
  - Und wir müssen Tupel ansehen, nicht Schemaelemente
- Aber: Kann sehr ähnliche Attribute korrekt trennen
  - Telefonnummern<>Faxnummern , Nachname<>Geburtsname

Inhalt dieser Vorlesung

- Schema Matching
- Matching Attributes
- Matching schemata
  - Global matching: Gale-Shapely
  - From attributed to relations
  - Matching schemata
- Varianten

Globales Matching

- Label- und instanzbasiert: Jedes Attributmatching individuell entscheiden
- Tatsächlich will man aber Schemata aufeinander abbilden
- Schemamatching ergibt sich nicht einfach aus besten Attributmatchings
  - Muss kein Fehler sein:
    C könnte PK und D ein FK dazu sein



- Erster Schritt: Globales Matching
  - Gesucht: Menge von disjunkten Attributmatchings, die global optimal ist
  - Klassische Formulierung: Finde maximales bipartites Matching
  - Alternative: Stable Marriage

Stable Marriage

- Gegeben: n Frauen und m Männer
  - (Wir sind hier konservativ)
  - Abstrakter: Zwei Mengen von Attributen
- Monogamie: Je eine Frau kann nur mit einem Mann verheiratet sein und umgekehrt
  - Nur 1:1 Attribut-Matchings
- Präferenzen: Jede Frau hat eine Rangliste ihrer bevorzugten Männer und umgekehrt
  - Attribut-Ähnlichkeit nach beliebigem Verfahren
- Gesucht: Stabile Ehen
  - Es darf keine Paare geben, für die gilt: f1 heiratet m1, f2 heiratet m2, aber f1 bevorzugt m2 und m2 bevorzugt f1
  - Diese Ehe wäre instabil

Beispiel 1

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

Quelle: David Toth, "The Stable Marriage Problem: More Marital Happiness than Reality TV" , April 25, 2003, Connecticut College, New London, CT, USA,

Beispiel 2

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)

Beispiel 3

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)

Beispiel 4

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 & verlässt 1: (2, C) (3, B)

Beispiel 5

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

Beispiel 6

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 & 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)

Beispiel 7

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 & 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)

Beispiel Ende

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

- Damit haben wir vier stabile Paare

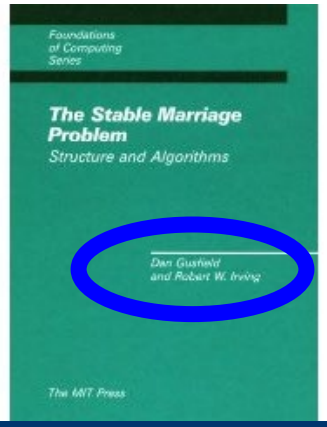
Algorithmus [GS62]

- Vorgehen war Greedy: Gale-Shapely Algorithmus
  - Findet garantiert eine stabile Lösung (aber nicht die optimale)
  - Bei gleicher Zahl Männer/Frauen gibt es immer eine stabile Lösung

```
function stableMatching {
  Initialize all m ∈ M and w ∈ W to free;
  while ∃ free man m who still has a woman w to propose to {
    w = m's highest ranked such woman to whom he has not yet proposed;
    if w is free
      (m, w) become engaged
    else some pair (m', w) already exists
      if w prefers m to m'
        (m, w) become engaged
        m' becomes free
    else
      (m', w) remain engaged
  }
}
```

Bipartites Matching – Stable Marriage

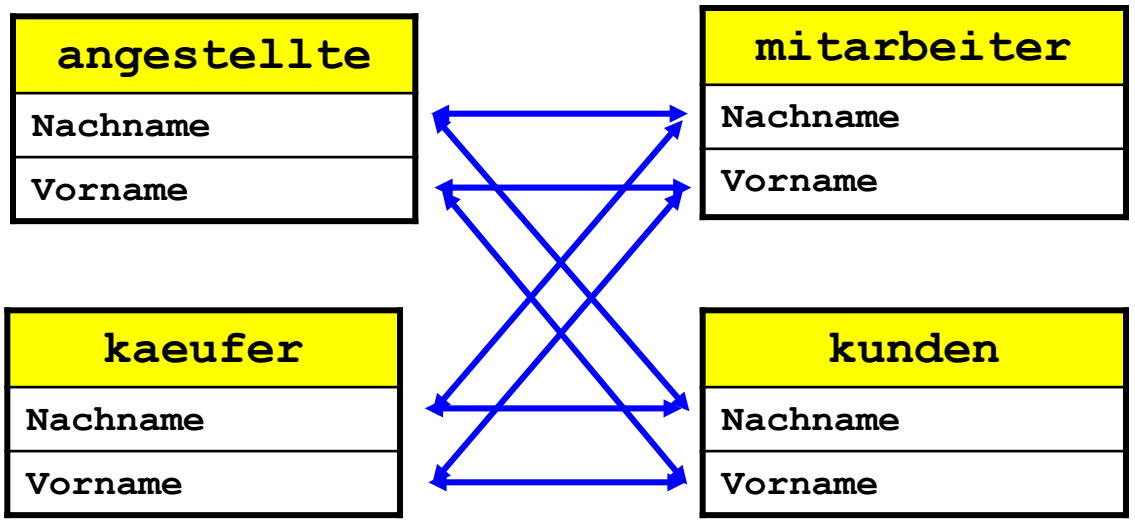
- Stable Marriage benötigt Rangordnung und keine numerischen Gewichte
  - Konkrete Werte sind oft ziemlich arbiträr
  - Ordnungen sind stabiler als konkrete Werte
- Stable Marriage liefert i.d.R. nicht die optimale Lösung im Sinne eines bipartiten Matchings
- Bipartites Matching führt i.d.R. nicht zu einer im Sinne des Stable Marriage Problems stabilen Lösung
- Komplexität
  - Bipartites Matching: $O(n^{2*}\log(n))$, z.B. ungarischer Algorithmus
  - Stable Marriage: $O(n^2)$, z.B. Gale-Shapley



Inhalt dieser Vorlesung

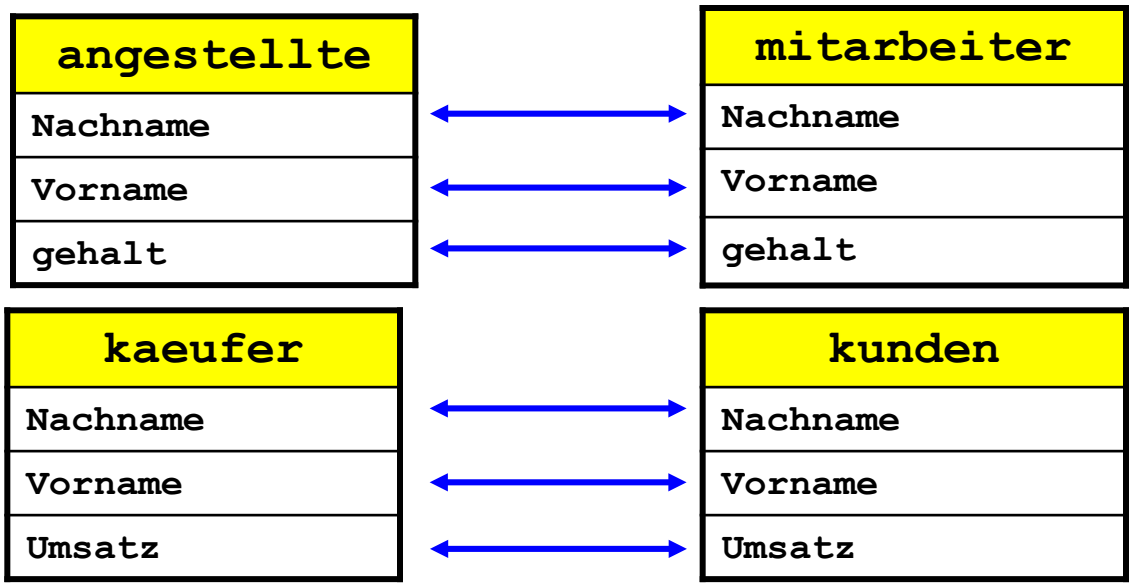
- Schema Matching
- Matching Attributes
- Matching schemata
  - Global matching: Gale-Shapely
  - From attributed to relations
  - Matching schemata
- Varianten

Probleme



- Attribut-Attribut-Matching
  - Instanzbasiert: Wertecharakteristika sehr ähnlich
  - Labelbasiert: Attributnamen sehr ähnlich
  - Alle Matchings etwa gleich gut

Besser



- Kontext der Attribute ausnutzen
- Attribute einer Relation sind auch im anderen Schema gerne (nicht immer) in einer Relation

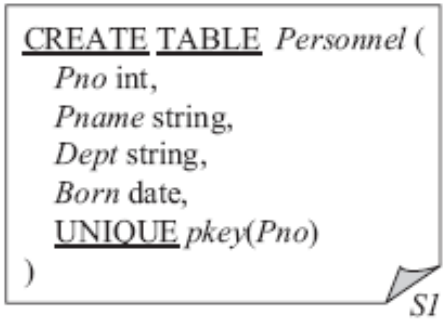
Von Attributen zu Relationen

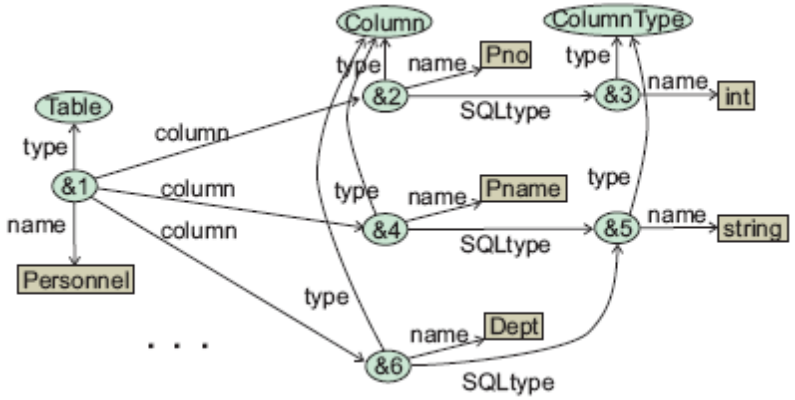
- Zwischenschritt: Matchen von Relationen und Attributen
- Einfache Methode
  - Eingabe
    - Berechne paarweise Ähnlichkeiten für alle Attribute
      - Über alle Relationen hinweg
    - Berechne paarweise Ähnlichkeit der Relationennamen
  - Dann zweistufiges Verfahren
    - Berechne Matching der Relationen – aggregierter Score aus Ähnlichkeiten der Relationennamen und der Attributmengen
    - Berechne Matching der Attribute nur innerhalb der gematchten Relationenpaare
- Ignoriert aber Beziehungen zwischen Relationen

Similarity Flooding [MGMR02]

- Geg. zwei Schemata R, S mit Relationen, Attributen, FK-PK Beziehungen und Datentypen
- Kernidee
  - Vergleiche alle Attributpaare und finde Matching-Kandidaten
  - Nutze Beziehungen zwischen Schemaelementen zur Auswahl der finalen Matchings
    - Attribute gehören zur selben Relation?
    - Attribute haben denselben Datentyp?
    - Attribute sind PK-FK-Paar?
    - Relationen sind durch PK-FK verbunden?
- Formuliert und gelöst als Graphproblem
  - R und S in einen Graphen umwandeln
  - Ähnlichkeiten von Paaren über Kanten abfärben lassen

Schema -> Graph

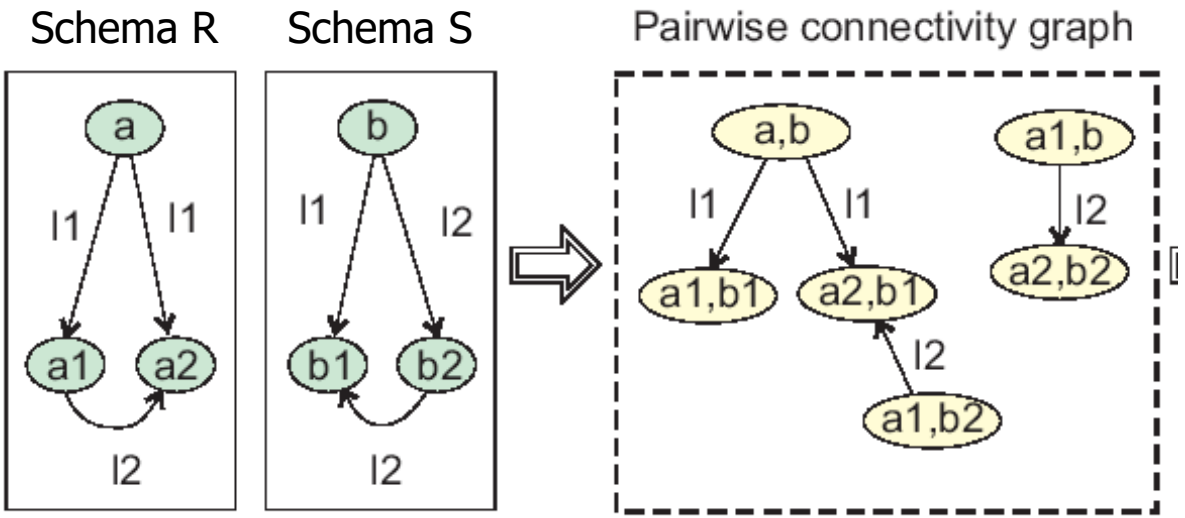




- &-Knoten repräsentieren Elemente des Datenmodells, insb. Relationen oder Attribute
- Eigenschaften sind als verbundene Knoten repräsentiert
  - Namen, Datentypen, …
- Art der Beziehung definiert durch Kantenlabel
  - Column_of, Type (column, table), SQLType, Name, FK-PK

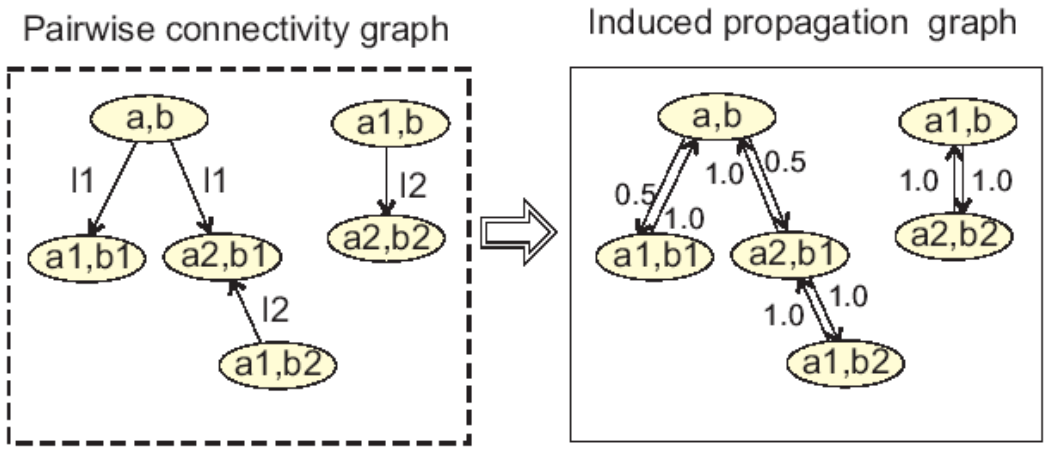
Zusammenführen: Kandidaten für Matchings

- Konstruktion des Pairwise Connectivity Graph (PCG)
  - Bedeutung (x,p,y): Kante mit Label p von Knoten x zu Knoten y
  - Sei (x, p, x‘)∈R und (y, p, y‘)∈S → Füge ((x,y), p, (x‘,y‘)) zu PCG
    - Knoten im PCG sind Paare von Elementen aus beiden Schemata
    - Es werden nur solche Paare betrachtet, die in den beiden Schemata mindestens durch eine Kante mit gleichem Label verbunden sind
    - Das sind uU sehr viele – welches sind die besten?



Kantengewichte

- Kanten werden nach Art der Kante gewichtet
  - Parameter: Je nach Kantentyp
  - Muss nicht symmetrisch sein (gerichtete Kanten)



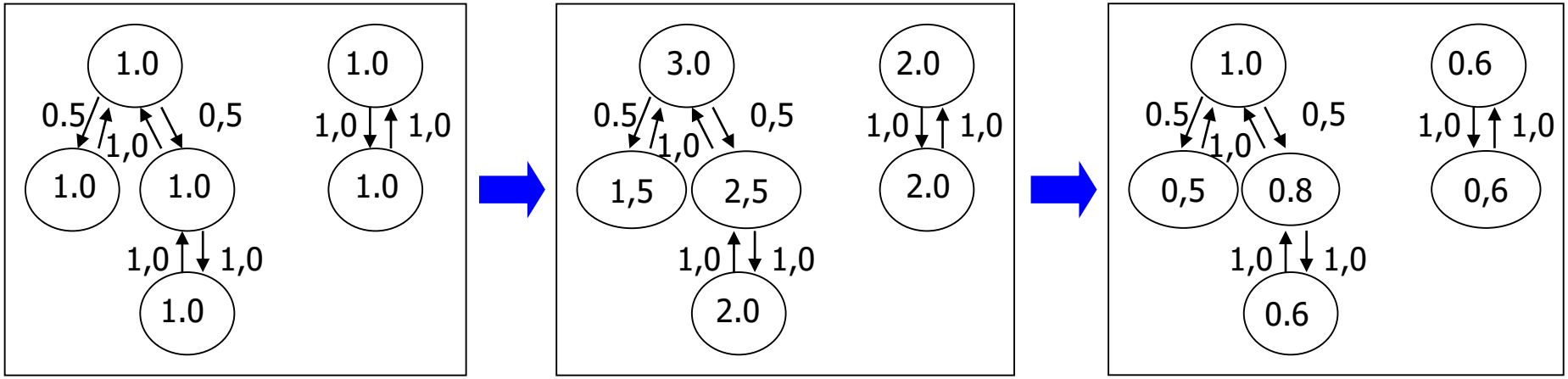
Idee des „Flooding“

- Knoten im PCG (also Paare) sind sich einzeln ähnlich
  - Ähnlichkeit der Elemente (Label, Instanzen, Datentyp …)
- Knoten erben die Ähnlichkeit ihrer „incoming“ Nachbarn
  - „Incoming“: Kante vom Nachbar zum Knoten
- Suche ein Gleichgewicht
  - Finde für jeden Knoten e ein Knotengewicht w‘(e) so, dass es der Summe der (kanten-gewichteten) Gewichte w‘ der Nachbarn von e entspricht

$$\forall e \in PCG: w'(e) = \sum_{n:\exists n \to e} w'(n) * w(n,e)$$

Lösungsverfahren

- Iteratives Lösungsverfahren (sketch)
  - Initialisiere w‘ von Knoten (a,b) mit sim(a,b)
  - Addiere zu w‘ die (kanten-gewichteten) w‘ seiner Nachbarn
  - Normiere alle w‘ Graphen mit dem maximalen Gewicht im PCG
  - Iteriere, bis Summe aller Änderungen kleiner als Schwellwert
- Knoten mit höchsten Scores sind die matchenden Paare
  - Wie viele?



Probleme

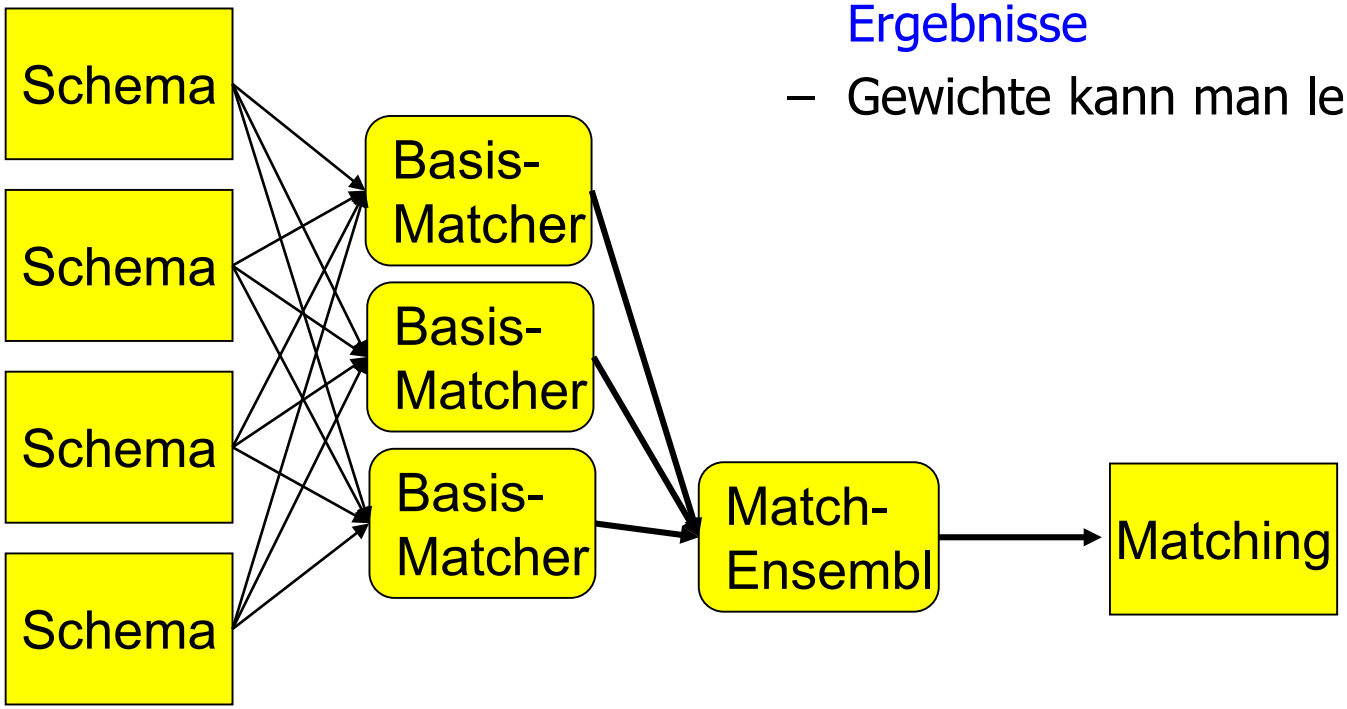
- Benötigt Verfahren für initiale Ähnlichkeiten sim(a,b)
- Diverse Parameter
  - Welche Kantengewichte? Wie lange iterieren? Schwellwert für Match-Paare?
- Kann u.U. langsam konvergieren
- Fazit: Wichtige Ergänzung zu den bisherigen Verfahren

Inhalt dieser Vorlesung

- Schema Matching
- Matching Attributes
- Matching schemata
  - Global matching: Gale-Shapely
  - From attributed to relations
  - Matching schemata
- Varianten

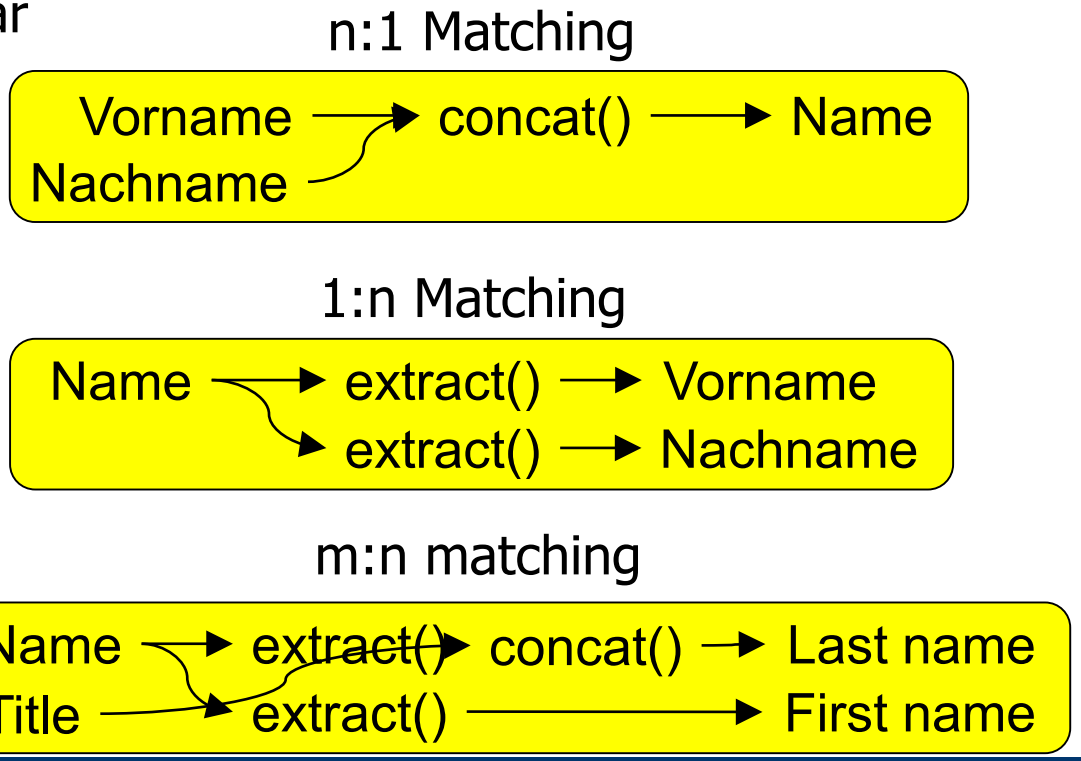
Mischformen

- Ensembles
  - Repertoire bekannter Verfahren
  - Zunächst unabhängige Anwendung
  - Dann (gewichtete) Kombination der Ergebnisse
  - Gewichte kann man lernen



Match-Granularität

- Bisher sind die Matchings immer 1:1
- Es gibt aber oft m:n, n:1 und 1:n Matches
- Wie soll man Werte aufbrechen / reorganisieren?
  - Viele Funktionen denkbar



Literatur

- [RB01] Rahm, Bernstein. A survey of approaches to automatic schema matching. VLDB Journal 10(4), 2001
- [BN05] Bilke, Naumann. Schema Matching using Duplicates. ICDE 2005
- [MGMR02] Melnik, Garcia-Molina, Rahm: Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. ICDE 2002
- [MH03] Madhavan, Halevy: Composing Mappings Among Data Sources. VLDB 2003
- [GS62] Gale, D. and Shapley, L. S. (1962). "College Admissions and the Stability of Marriage." American Mathematical Monthly 69: 9-14.