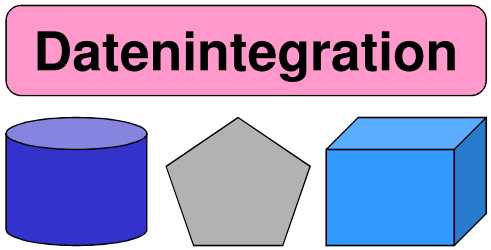
Datenintegration

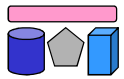


Kapitel 7: Datenfusion

Michael Hartung in Vertretung von **Dr. Andreas Thor**

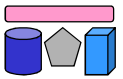
Wintersemester 2010/11

Universität Leipzig
Institut für Informatik
http://dbs.uni-leipzig.de



Inhalt

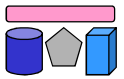
- Datenqualität
  - Beispiel und (typische) Probleme
  - Informationsqualität
- Object Matching (Duplikaterkennung)
  - Definition und Modell
  - Ähnlichkeitsmaße
  - MOMA
- Konfliktbehandlung
  - Vereinigung
  - Gruppierung und Aggregation



Datenqualität in Integrierten Inf. Systemen

- Integrierte Informationssysteme besonders anfällig für Qualitätsprobleme
  - Qualität der Ursprungsdaten (Eingabe, Fremdfirmen,...)
  - Qualität der Quellsysteme (Konsistenz, Constraints, Fehler, ...)
  - Qualität der Integrationsprozesse (Parsen, Transformieren, Mappings, ...)

  $\rightarrow$ Akkumulation!
- Probleme treten erst bei integrierter Sicht zu Tage



Datenqualität am Beispiel

- Google Scholar: Suchmaschine für wissenschaftliche Publikationen (scholar.google.com)
  - Datengenerierung u.a. durch automatisches Crawling und Extraktion aus PDF-Dokumenten
- Datenqualitätsproblem: Duplikate, u.a. durch
  - Tippfehler in den Originaldaten
  - Falsche Reihenfolge der Autoren
  - Heterogene Bezeichnung des Konferenzen/Journals (u.a. Abkürzung vs. Langname)
  - Extraktionsfehler (u.a. falsche Titel, fehlende Autoren)

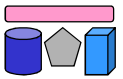
A **survey** of approaches to automatic **schema** matching - all 35 versions »
E **Rahm**, PA Bernstein - The VLDB Journal The International Journal on Very Large ... , 2001 - Springer
... A **survey** of approaches to automatic **schema** matching **...** Page 2. E. Rahm, PA Bernstein:
A **survey** of approaches to automatic **schema** matching 335 2.1. **...**
Cited by 1222 - Related Articles - Web Search

[CITATION] A **survey** of approaches to automatic **schema** matching
PA Bernstein, E **Rahm** - VLDB Journal, 2001
Cited by 17 - Related Articles - Web Search

[CITATION] A **survey** of approaches to automatic **schema** mapping
E **Rahm**, PA Bernstein - The VLDB Journal, 2001
Cited by 7 - Related Articles - Web Search

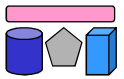
[CITATION] A **survey** of approaches to automatic **schema** mapping" the VLDB Journal
E **Rahm**, PA Bernstein - Vol
Cited by 3 - Related Articles - Web Search

[CITATION] PA Bernstein A **survey** of approaches to automatic **schema** matching The VLDB Journal 10: 334350 (2001)
E **Rahm** - Digital Object Identi. er (DOI)
Cited by 1 - Related Articles - Web Search



Beispiele mangelnder Datenqualität

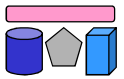
- Analyse wissenschaftlicher Publikationen
  - Unvollständige und nicht duplikatfreie Listen pro Autor, pro Konferenz, ...
  - Unvollständige Zitierungszahlen → Fehler in abgeleiteten Werten
- Customer-Relationship-Management
  - Kunden doppelt geführt → verärgerte Kunden durch unpassende Angebote
  - Kunden falsch bewertet → verpasste Gelegenheiten z.B. für Cross-Selling
  - Falsche Adressen → Portokosten
- Britische Gendatenbank (siehe Heise News vom 28.08.2007)
  - Genprofile von 4 Millionen Menschen: “rund 550.000 Namen in der Datenbank sind falsch, falsch geschrieben oder fehlerhaft”, “Personen[zahl] ... etwa um 13,7 Prozent niedriger sei als die Zahl gesamten Einträge”
  - Folgen?
- Kostenschätzungen durch Analysen
  - 25-40% der operativen Kosten; Milliardenbeträge



Informationsqualität

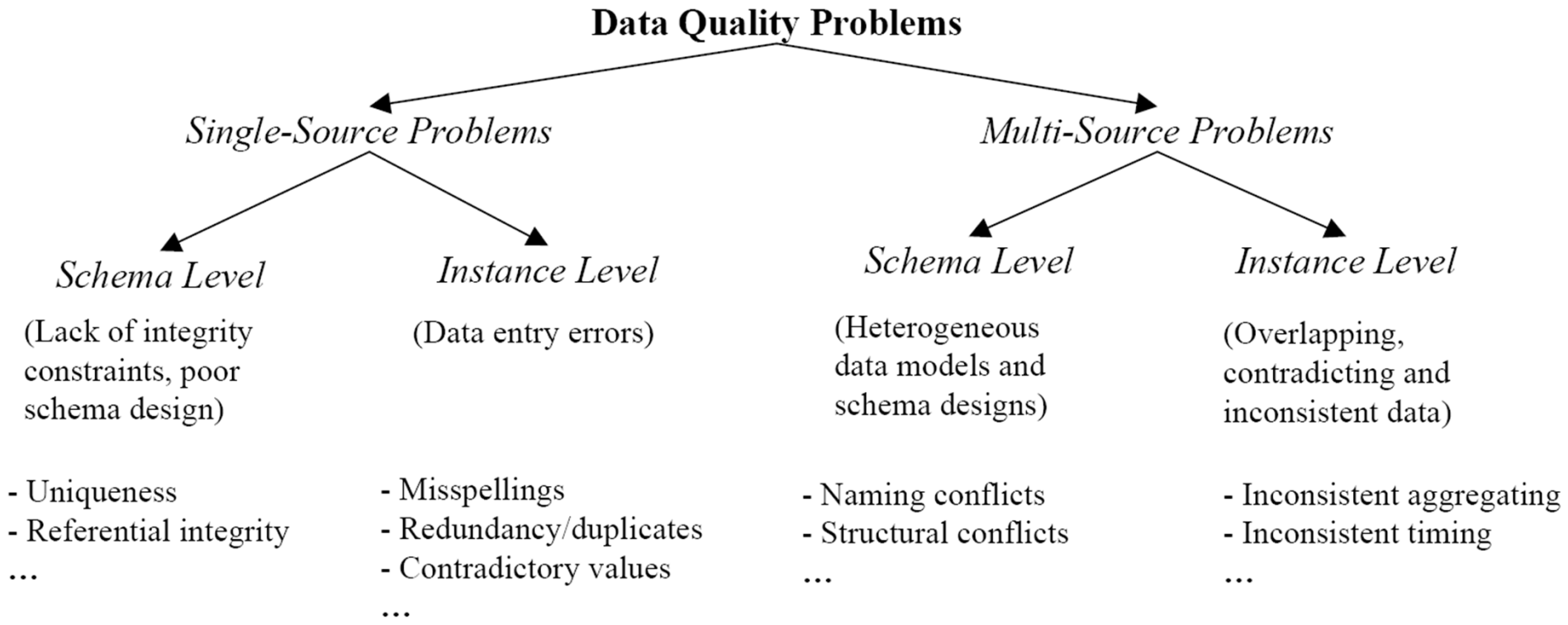
- Datenbasierte Inf.-Qualität (Datenqualität)
  - Korrektheit, Vollständigkeit, Interpretierbarkeit, Relevanz, Zuverlässigkeit, liefert Mehrwert, ...
- Technische Inf.-Qualität
  - Verfügbarkeit, Kosten, Latenz (Zeit bis zum ersten Ergebnis), Antwortzeit (Zeit bis zum kompletten Ergebnis), Sicherheit, Aktualität, ...
- Intellektuelle Inf.-Qualität
  - Glaubhaftgkeit, Objektivität, Reputation, ...
- Inf.-Qualität bzgl. Realisierung der Inf.-Systems (Darstellung)
  - Datenmenge, Kompaktheit, Konsistenz, Verständlichkeit, Verifizierbarkeit, ...

“Even though quality cannot be defined, you know what it is.” (Robert Pirsig)

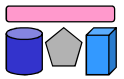


Probleme bzgl. Datenqualität

- Probleme auf Schema- und auf Instanzebene
- Probleme bezüglich einer oder mehrerer Datenquellen (Single-Source vs. Multi-Source)

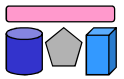


[RD00] Rahm, Do: Data Cleaning: Problems and Current Approaches. IEEE Data Eng. Bull. 23(4): 3-13 (2000)



Probleme bzgl. Instanzdatenqualität

- Intra-Source
  - Fehlende Integritätsbedingungen oder Konsistenz-Checks
  - Nicht korrekte Einträge (Tippfehler, Übertragungsfehler, fehlerhaftes OCR, ...)
  - Obsolete Einträge (falsche Aktualisierungszeitpunkte, vergessene Aktualisierung, ...)
  - Unterschiedliche Kodierung (1,2,...,5 bzw. „sehr gut", „gut", ..., „mangelhaft“)
  - Schreibvarianten
    - Kantstr. / Kantstrasse / Kant Str. / Kant Strasse
    - Kolmogorov / Kolmogoroff / Kolmogorow
- Inter-Source
  - Lokal konsistent aber global inkonsistent
  - Hervorgerufen durch Unterschiede in
    - Abkürzungen
    - Standards
    - Datentypen
    - Konventionen
    - Aktualisierungszeitpunkten
    - ...



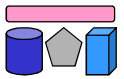
Object Matching: Problem

- Erkennung “gleicher” Objekte in Datenquellen
  - Entity Matching, Deduplication, Duplicate Detection, Record Linkage, Object Consolidation, Merge/Purge Problem, Reference Reconciliaton, ...
- Gegeben: Objektmengen A = $\{a_i\}$ und B = $\{b_j\}$
  - Objekte repräsentiert als Datenbanksätze, XML-Fragmente, RDF-Tripel, ...
  - Spezialfall: A = B (Duplikaterkennung innerhalb einer Datenquelle)
- Gesucht: Partitionierung von A $\times$ B in
  - M = { (a_i, b_j) } mit a_i und b_j referenzieren das gleiche Realweltobjekt
  - U = { (a_i, b_j) } mit a_i und b_j referenzieren *nicht* das gleiche Realweltobjekt
- Beispiel: Personen-/Adressdaten

Id	LastName	FirstName	Street	City	Gender
a_1	Smith	Kristen	2 Hurley Pl	South Fork, MN	female
a_2	Smith	Christian	Hurley St 2	S Fork MN	male
b_1	Smith	Christoph	23 Harley St	Chicago, IL	male
b_2	Smith	Kris L.	2 Hurley Place	South Fork, MN	female

M =

U =

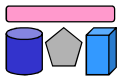


Object Matching: Evaluation

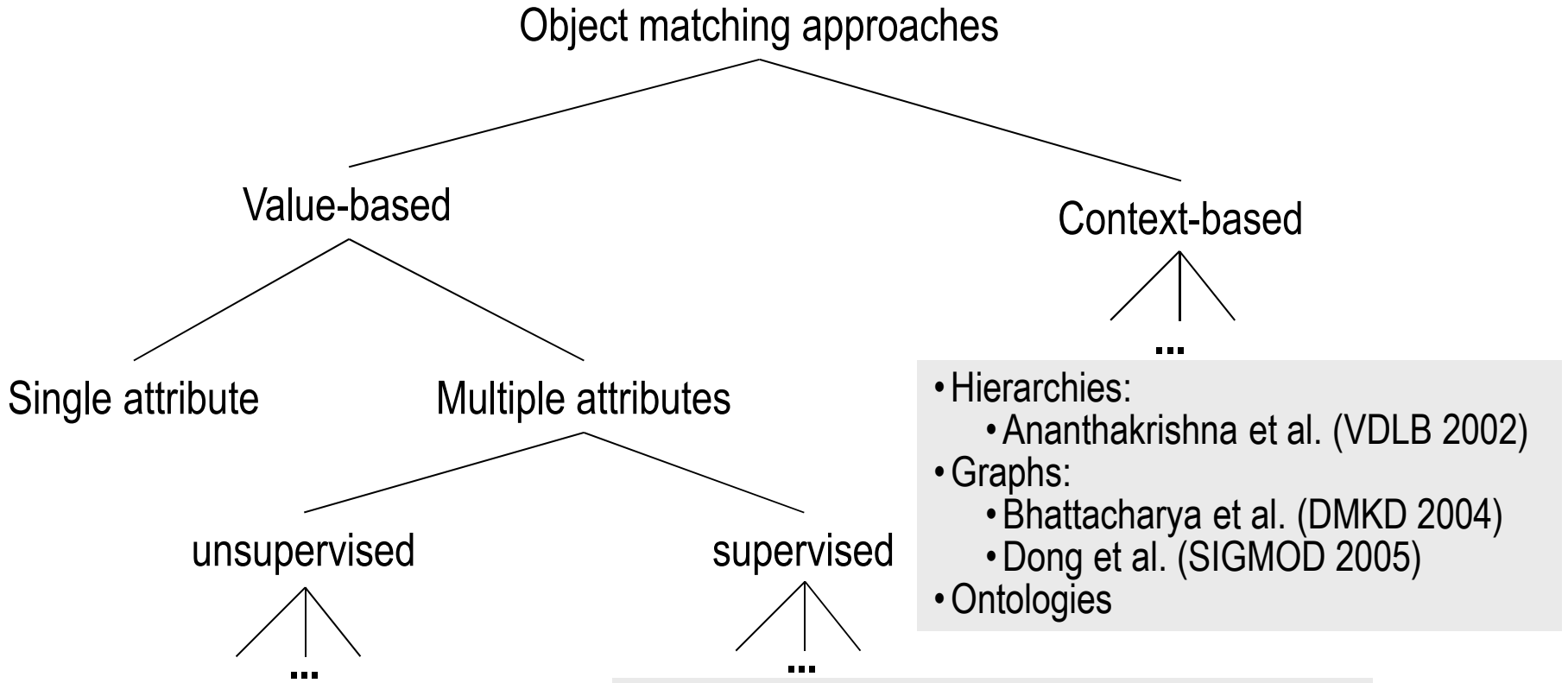
- Bewertung mittels Precision und Recall
  - Qualitätsmaße aus dem Information Retrieval

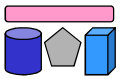
		Realität	
		Duplikat	Kein Duplikat
Methode	Duplikat	true-positive	false-positive
	Kein Duplikat	false-negative	true-negative

- Precision = TP/(TP+FP)
  - Wie viele der vorhergesagten Duplikate sind wirklich welche?
  - Auch: Spezifität
- Recall = TP/(TP+FN)
  - Wie viele der echten Duplikate haben wir gefunden?
  - Auch: Sensitivität
- Beispiel:
  - Verfahren: M = { (a_1, b_2) } und U = { (a_1,b_1), (a_2,b_1), (a_2,b_2) }
  - Realität: M = { (a_1, b_2) , (a_2,b_1) } und U = { (a_1,b_1), (a_2,b_2) }
  - Precision:
  - Recall:

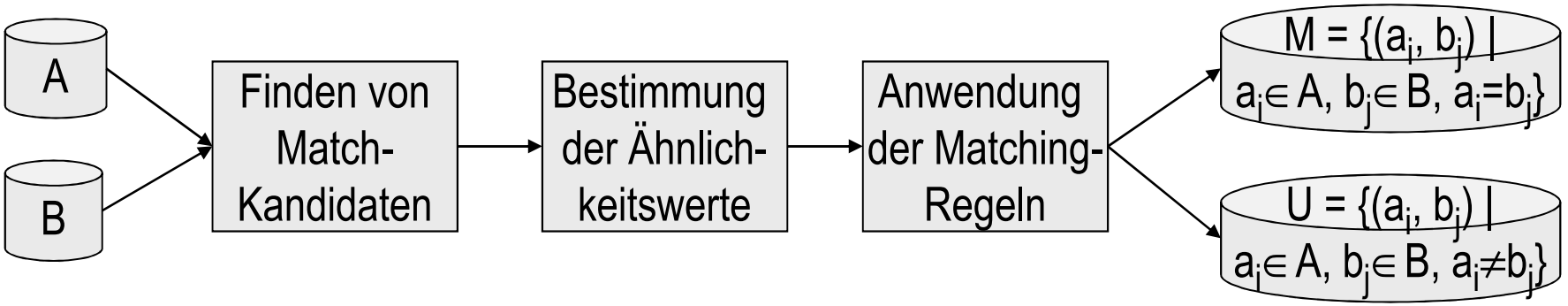


Übersicht über Object-Matching-Ansätze

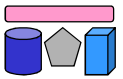




Object Matching: Modell



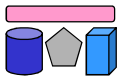
- Finden von Match-Kandidaten
  - "Welche Objekte sollen (intensiv) miteinander verglichen werden?"
  - Ziel: effiziente Verarbeitung durch frühzeitigen Ausschluss ungleicher Objekte
- Bestimmung der Ähnlichkeitswerte
  - "Wie ähnlich sind sich die Objekte a und b bzgl. eines Ähnlichkeitsmaßes?"
  - Ziel: mathematische Quantifizierung der Ähnlichkeit zwischen Objekten
- Anwendung der Matching-Regeln
  - "Wann werden die Objekte a und b als gleich angesehen (basierend auf Ä-Werten)?"
  - Ziel: korrekte Klassifikation in "match" und "non-match"



Finden von Match-Kandidaten

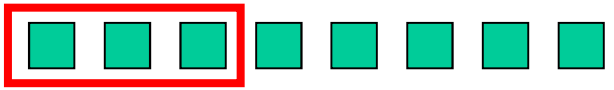
- Ähnlichkeitsbestimmung des karthesischen Produkts $A \times B$ sehr aufwändig $\rightarrow$ Reduktion auf "relevante Paare (a_i, b_j)"
  - #Match-Paare << #Nicht-Match-Paare
- Ziel: Viele Nicht-Matches entfernen (hohe Reduction Rate), aber keine Matches (fälschlicherweise) entfernen (Pairs Completeness = 1)

- Blocking
  - Bestimme Blockschlüssel aus Objektattributen
    - Beispie für Personenl: Erster Buchstabe des Nachnamens + Geschlecht
  - Nur Objekte mit gleichem Schlüssel werden weiter betrachtet
- Sorted-Neighborhood-Methode
  - nächste Folie
- Clustering
  - Verwende billige Ähnlichkeitsfunktion und bilde Cluster



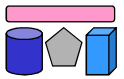
Sorted-Neighborhood-Methode

- Bilde Schlüssel aus Objektattributen
  - Beispiel: Die ersten drei Buchstaben des Nachnamens + die ersten vier Konsonanten des Vornamens + der erste Buchstabe des Geschlechts
- Sortiere Objekte bzgl. Schlüssel
- Vergleiche nur Objekte die maximal *w*-1 Positionen entfernt liegen
  - Sliding Window der Größe w



- Aufwand für n Objekte
  - SNM: Schlüsselbildung + Sortieren + Vergleich = O(n) + O(n*log(n)) + O(n*w)
  - naiver Ansatz: $O(n^2)$
- Problem: Was ist ein guter Schlüssel? (Attribute, Reihenfolge, ...)
  - Multi-Pass-Ansatz, d.h. mehrere Durchläufe mit unterschiedlichen Schlüsseln

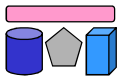
Id	LastName	FirstName	Street	City	Gender	Schlüssel
a_1	Smith	Kristen	2 Hurley Pl	South Fork, MN	female	
a_2	Smith	Christian	Hurley St 2	S Fork MN	male	
b_1	Smith	Christoph	23 Harley St	Chicago, IL	male	
b_2	Smith	Kris L.	2 Hurley Place	South Fork, MN	female	



Ähnlichkeitsbestimmung, Ähnlichkeitsmaße

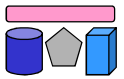
- Attributbasierte Ähnlichkeitsbestimmung
  - Gleiche/ähnliche Objekte haben gleiche/ähnliche Attributwerte
  - Bestimmung der Ähnlichkeit von Attributwerten → Field Matching
- Kontextbasierte Ähnlichkeitsbestimmung
  - Verwendung von Struktur, assoziierten Informationen, ...
  - Beispiel: Neighborhood-Matcher in MOMA

- Generische Ähnlichkeitsmaße
  - Weit verbreitet, da in vielen Bereichen einsetzbar
  - Vorwiegend für Zeichenketten (Strings):
    - Soundex, Edit-Distance, q-gram, Jaccard, Jaro-Winkler, ...
    - kein bestes Ähnlichkeitsmaß; Qualität abhängig von Match-Problem (z.B. String-Länge)
- Domänenspezifische Ähnlichkeitsmaße
  - Spezielle Funktionen, z.B. Abstand geografischer Orte aus Längen- und Breitengrad
  - Verwendung von Hintergrundwissen (z.B. Synonyme)



Ähnlichkeitsmaße für Strings: Soundex

- Entwickelt 1918 zur Volkszählung in den USA
- Idee: Gleichklingende Wörter werden in identische Zeichenfolge übersetzt
- Soundex-Code:
  - Erster Buchstabe gefolgt von Codes für die nächsten drei Konsonanten
  - Ähnliche Konsonanten besitzen den gleichen Code (B und P erhalten „1“, V und B erhalten „0“)
  - Beispiel: Soundex („Naumann“) = Soundex („Neuman“) = N550
- Gut: Beachtet Lautähnlichkeiten, oft implementiert
- Schlecht: Sehr heuristisch (fehlende Konsonanten? Warum nur 3? Warum keine Vokale?)
- Nur für Namen geeignet, Fehlermodell ist „verhören“ (und nicht vertippen)
- Abhängigkeit von Sprache (Soundex für das Neugriechische?)



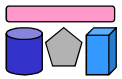
Ähnlichkeitsmaße für Strings: Edit-Distance

- Gegeben seien zwei Strings a und b (|a|=n, |b|=m)
- Editabstand = Anzahl der Edit-Operationen, um a in b zu konvertieren
  - Edit-Operationen: Einfügen, Löschen oder Ersetzen eines Zeichens
- Ähnlichkeit ist normierter Editabstand
  - auch: Levenshtein-Abstand

$$sim_{edit}(a,b) = 1 - \frac{dist(a,b)}{\max(|a|,|b|)}$$

- Editabstandsfunktion d(i,j) mit 0≤i≤n und 0≤j≤m
  - berechne den Editabstand zwischen a[1..i] und b[1..j]
  - d(0,j) = j (=j Einfügeoperationen) und d(i, 0) = i (i Löschoperationen)
  - d(n,m) = dist(a,b) = Editabstand zwischen a und b

$$d(i,j) = \min\begin{Bmatrix} d(i,j-1)+1 \\ d(i-1,j)+1 \\ d(i-1,j-1)+t(i,j) \end{Bmatrix} \qquad t(i,j) = \begin{cases} 1 : wenn\ a[i] \neq b[j] \\ 0 : sonst \end{cases}$$

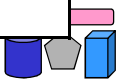


Ähnlichkeitsmaße für Strings: Edit-Distance (2)

- Beispiel: RASEN und HASE
  - RASEN $\rightarrow$ RASE $\rightarrow$ HASE
  - RASEN $\rightarrow$ HASEN $\rightarrow$ HASE
  - Edit-Abstand = 2 $\rightarrow$ String-Ähnlichkeit = 1 – 2 / max(5;4) = 0,6
- Berechnung des kürzesten Editabstandes mittels Abstandsmatrix
  - Speichern der Teillösungen (für rekursive Berechnung)
  - Initialisierung mit festen Werten d(i,0) und d(0,j)
  - Sukzessive Berechnung von d(i,j) mit steigenden i, j

$$d(i,j) = \min \begin{Bmatrix} d(i,j-1)+1 \\ d(i-1,j)+1 \\ d(i-1,j-1)+t(i,j) \end{Bmatrix}$$

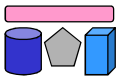
		H	A	S	E
	0	1	2	3	4
R	1				
A	2				
S	3				
E	4				
N	5				



Ähnlichkeitsmaße für Strings: q-gram

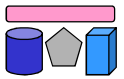
- Gegeben seien zwei Strings a und b (|a|=n, |b|=m)
- Idee: Wieviele gleiche Substrings der Länge q enthalten a und b?
  - Häufig q = 3 (Trigram)
- Ähnlichkeit mittels Dice-Koeffizient $sim_{qgram}(a,b) = \frac{2 \cdot |Q(a) \cap Q(b)|}{|Q(a)| + |Q(b)|}$
  - Q(a) = Menge der Trigramme von a
  - zu vergleichende Strings erhalten Präfix und Suffix mit je q-1 Füllzeichen

- Beispiel: RASEN und HASE mit q=3
  - RASEN $\rightarrow$ ##R , #RA , RAS , ASE , SEN , EN% , N%%
  - HASE $\rightarrow$
  - Ähnlichkeit =

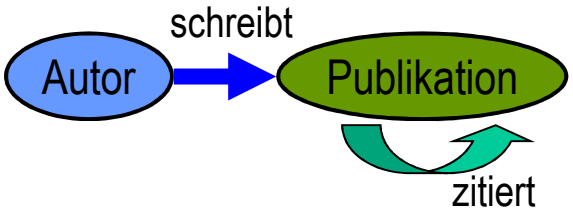


Ähnlichkeiten für Bäume

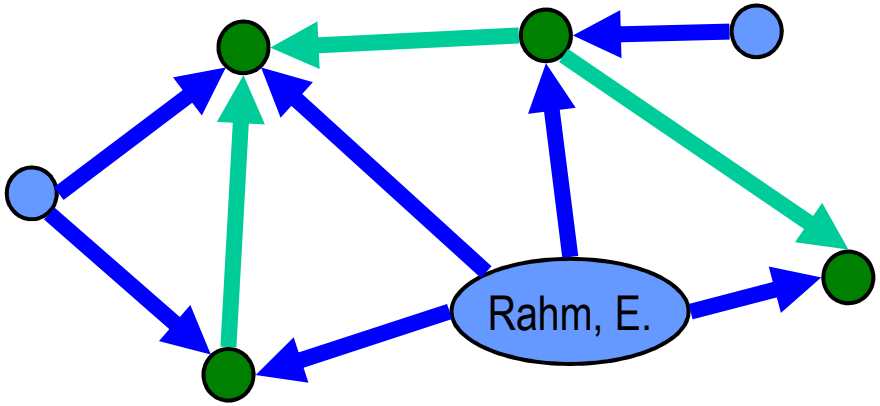
- Relevanz u.a. für XML-Dokumente (Baumstruktur)
- Ausnutzung der Hierarchie in der Ähnlichkeitsbeachtung
  - Bottom-Up
    - Sukzessive Aggregation von Ähnlichkeiten für Unterbäume
    - Eltern aggregieren Ähnlichkeiten ihrer Kinder
    - Teuer – immer den gesamten Baum betrachten
  - Top-Down
    - Kinder nur vergleichen, wenn Eltern hinreichend ähnlich
    - Billiger – viele Kinder werden nie verglichen
    - Aber: Geringerer Recall
- Spezielle Baumprobleme
  - Fehlende / neue Kinder
    - XML ist semistrukturiert
    - Ähnliches Problem: NULL-Werte in RDBMS
  - Reihenfolgevertauschungen
  - Möglichkeit: Tree-Edit-Distance

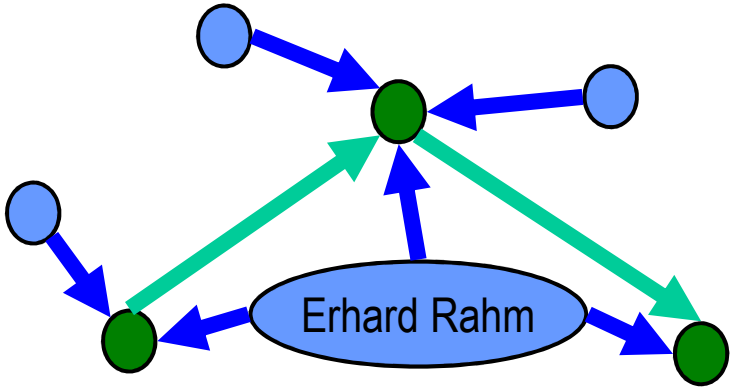


Ähnlichkeit für Graphen

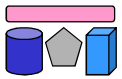


- Darstellung von Objektbeziehungen als Graph





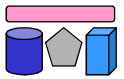
- Anwendungen
  - Erkennung gleicher Autoren aufgrund Ko-Autorenschaft und zitierter Dokumente
    - Autoren neigen dazu, oft mit denselben Koautoren zu publizieren
    - Autoren neigen dazu, gleiche Papiere (vor allem eigene) zu zitieren
  - Erkennung von gleichen Webseiten aufgrund der Struktur der Links
    - Mirror-sites, Google-Bombs
  - Erkennen gleicher Personen in sozialen Netzwerken
  - ...
- Beispiel-Algorithmus: MOMA's Neighborhood-Matcher



Matching-Regeln

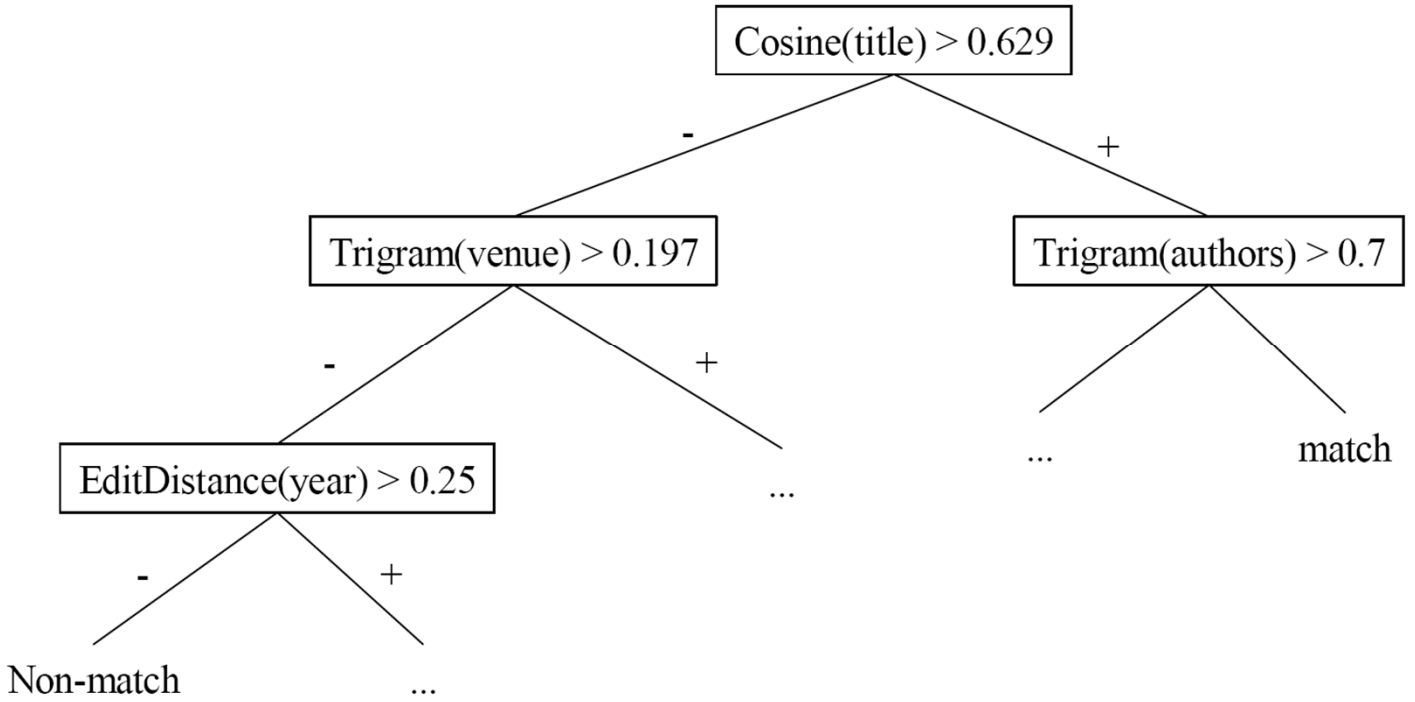
- Verfahren (auf Basis ermittelter Ähnlichkeitswerte) zur Entscheidung “match” oder “non-match”

- Schwellwertbasiert: alle Paare mit Ähnlickkeit $\geq t$
- Top-N: für jedes Objekt x diejenigen N Objekte y mit größten sim(x,y)
- Kombination mehrerer Ähnlichkeitswerte mittels Regeln
  - Gewichtung der Attribute und/oder Ähnlichkeitsmaße
  - Beispiel: $sim_1(\text{FirstName}) \geq 0.5 \wedge sim_2(\text{LastName}) \geq 0.8 \rightarrow$ “match” else “non-match”
- Constraints
  - Einbringen von Domänenwissen zur Überprüfung des Match-Ergebnisses
  - Beispiel:
    - Matching von Personen, Quelle A liefert Alter, Quelle B liefert Einkommen
    - $\text{Alter}(p_A) < 10 \wedge \text{Einkommen}(p_B) > 50.000€ \rightarrow (p_A, p_B)$ ist non-match



Lernverfahren für Object-Matching

- Finden effektiver Match-Einstellungen ist schwierig
  - Auswahl der Attribute, Matcher (Ähnlichkeitsmaße), Einstellungen (u.a. Schwellwerte und Gewichtungen bei Matching-Regeln)
- Machine Learning verspricht Verbesserung
  - manuell spezifizierte Trainingsdatenmenge
  - Lernen von Match-Kriterien (z.B. mit Entscheidungsbaum)
  - Problem: gute Trainingsdaten mit vertretbarem manuellen Aufwand



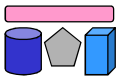
Object-Matching-System MOMA

- MOMA = **M**apping based **O**bject **Ma**tching
- Framework für Objekt-Matching (Fuzzy Match)
  - Unterstützung komplexer Match-Workflows
  - Kombination mehrerer Matcher / Ergebnisse
  - Wiederverwendung bereits berechneter Mappings
  - Unterstützuung heterogener Datenquellen, v.a. von Web-Daten
- Mapping-basierter Ansatz
  - Match-Ergebnis ist Mapping bestehend aus Instanz-Korrespondenzen („Same-Mapping“)
  - bereits existierende Mappings (z.B. Web-Links) werden ausgenutzt
  - semantische Beziehungen zwischen Objekten („Assoziations-Mappings“) werden durch spezielle Matcher bzw. Workflows ausgenutzt

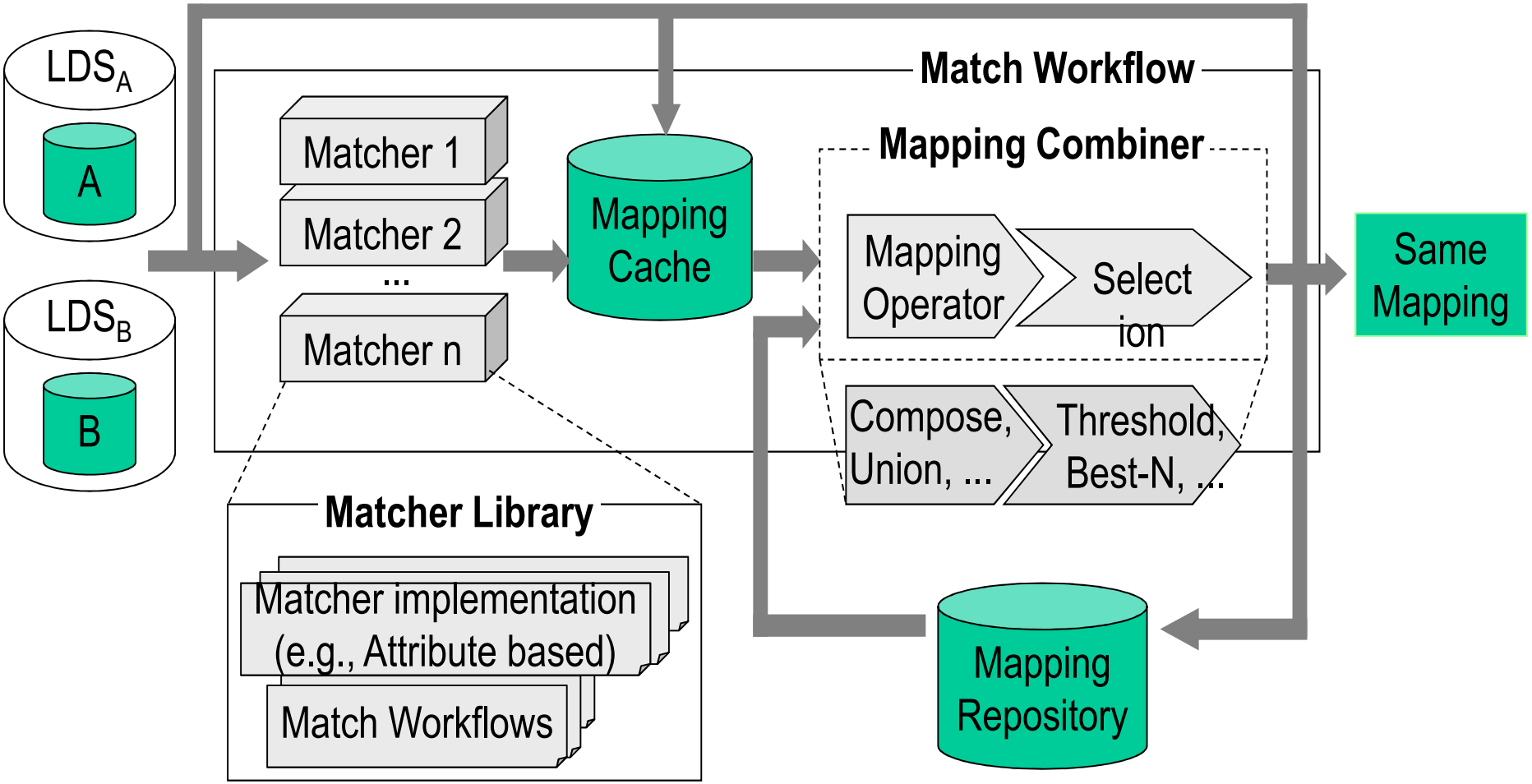
$Quelle_A$	$Quelle_B$	Sim
a_1	b_1	0.9
a_2	b_2	0.7
a_3	b_3	1

- Realisierung im Rahmen der iFuice-P2P-Architektur zur Datenintegration

Thor, A., Rahm, E.: MOMA - A Mapping-based Object Matching System. CIDR, 2007

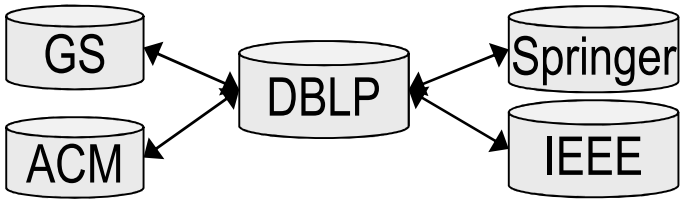


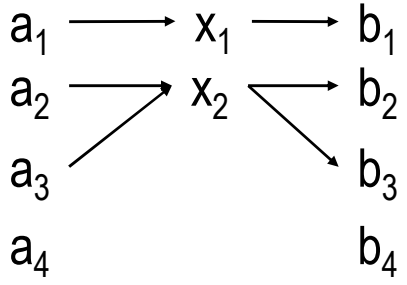
MOMA-Architektur



Wiederverwendung von Match-Ergebnissen

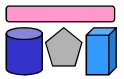
- Vereinigung / Durchschnitt von Mappings zur Absicherung der Match-Qualität
  - Vereinigung erhöht i.Allg. Recall auf Kosten der Precision
  - Durchschnitt erhöht i.Allg. Precision auf Kosten des Recalls
- Komposition von Mappings ermöglicht effiziente Berechnung neuer Mappings
  - compose $(M_{AX}, M_{XB}) = \{ (a_i, b_k) \mid (a_i, x_j) \in M_{AX}, (x_j, b_k) \in M_{XB} \}$
  - Besonders gut geeignet, falls Hub-Datenquelle vorhanden ist (Sternstruktur)
  - Fehlende Objekte in „mittlerem“ Datenquelle führen zu fehlenden Korrespondenzen (Bsp: a_4-b_4); Komposition kann zu falschen Korrespondenzen führen (Bsp: a_2-b_3)
  - Grundannahme: Transitivität der Ähnlichkeit (“$a \approx x \wedge x \approx b \rightarrow a \approx b$”) ... gültig?





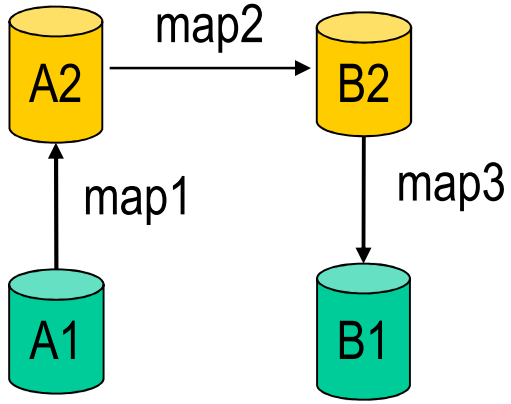
Neighborhood-Matcher: Idee

- Motivation: Wertevergleich für heterogene Objekte schwierig
- Beispiel für gleiche Konferenzen
  - Proceedings of the 27th International Conference on Very Large Databases
  - Proc. of VLDB 2001, Italy
- Lösung 1: Match-Verfahren mittels Domänenwissen
  - Abkürzungen, z.B. VLDB = Very Large Databases
  - Zuordnungen, z.B. „VLDB 2001“ = „27. VLDB“
  - ...
  - Problem: Woher kommt Domänenwissen? Bei jeder Domäne anders!
- Lösung 2: Verwendung assoziierter Informationen
  - Beispiel: „Zwei Konferenzen sind gleich, wenn die Menge der zugehörigen Publikationen gleich sind.“
  - Mögliche Abschwächungen
    - alle → viele
    - gleich → ähnlich



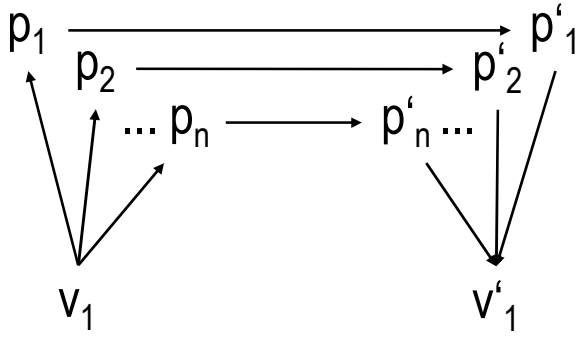
Neighborhood-Matcher: Match-Workflow

- Verwendung von Assoziations-Mappings
  - Syntax: Gleicher Struktur wie Same-Mappings; fester „Ähnlichkeitswert“ = 1
  - Semantik: Korrespondenzen zwischen assoziierten Objekten, z.B. Publikationen - Venue
- Match-Workflow als Kompositon von drei Mappings
  - map1 und map3 sind Assoziations-Mappings
  - map2 ist ein Same-Mapping



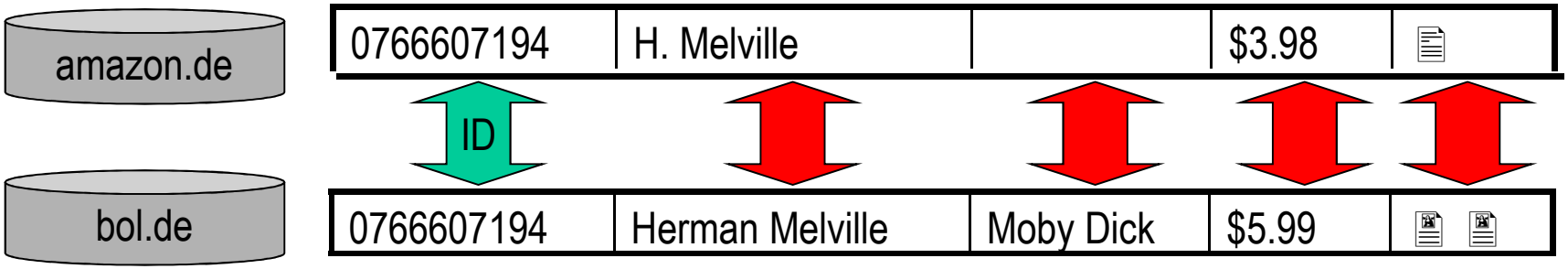
- Idealfall (rechts) nicht immer erreicht, da
  - Assoziations-Mappings unvollständig, z.B. nicht alle Publikationen in Datenquelle zu jedem Venue verfügbar
  - Same-Mapping fehlerhaft, z.B. als Ergebnis eines automatischen Match-Verfahrens
- Ähnlichkeitswert für $(v_i\text{-}v'_j)$
  - Dice-Koeffizient (analog q-gram)

$$\frac{2 \cdot \#\text{Pfade von } v_i \text{ nach } v'_j}{\#\text{Korresp. } (v_i,p_r) + \#\text{ Korresp. } (p'_s, v'_j)}$$

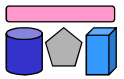


Datenkonflikte und Konfliktlösung

- Datenkonflikt: Duplikate haben unterschiedliche Werte für ein semantisch gleiches Attribut
- Datenkonflikte sind
  - Intra-Source: Innerhalb eines Informationssystems
  - Inter-Source: Bei der Integration mehrerer Informationssysteme



- Datenauswahl: Präferenzordnung
  - Aktualität der Quelle
  - Vertrauen in Datenquelle
  - Quelle mit mehr Informationen
  - Mehr als 2 Quellen: Majority Voting
- Datenfusion
  - Verwendung von Konfliktlösungsfunktionen
  - „Aggregation“ der Konfliktwerte in einen
  - Hochgradig domänen-/attributabhängig



SQL: Standard Union

- Unbefriedigend
  - Keine Duplikaterkennung, nur Eliminierung identischer Tupel
  - Schemata müssen identisch sein, keine strukturelle Heterogenität
  - Kein Füllen von Lücken: NULL ≠ Wert
  - Keine Datenfusion (Probleme mit PK‘s)

Mitarbeiter m

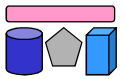
p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Stefanie	Meier	34

Employees e

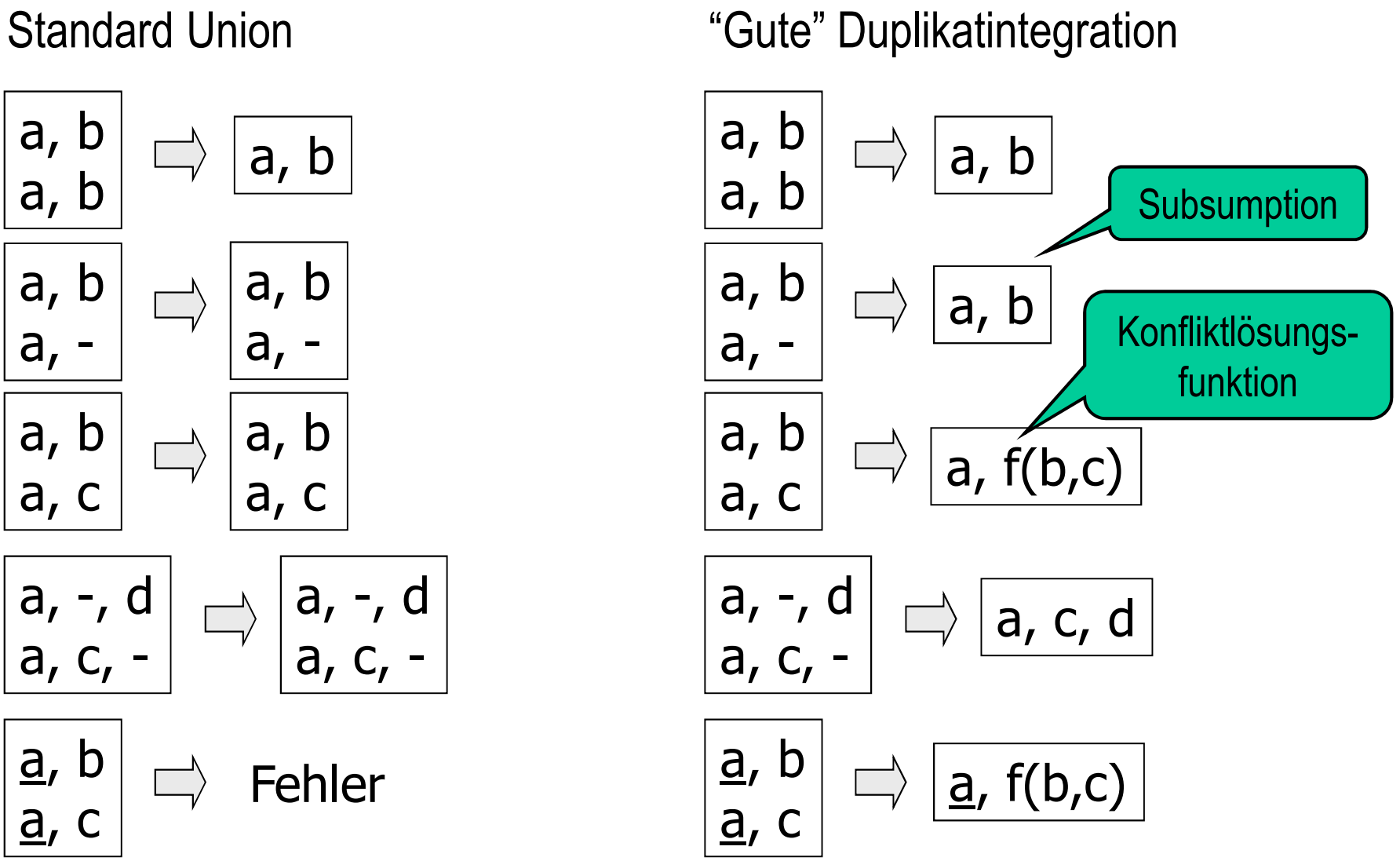
p_id	vorname	nachname	alter
5	Petra	Weger	28
2	Stefanie	Meier	Null

m ∪ e

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Stefanie	Meier	34
5	Petra	Weger	28
2	Stefanie	Meier	Null



Duplikatintegration



Annahme: a identifiziert Objekte (und damit Duplikate)

Outer Union

- R1 und R2 Relationen mit Schemata S1 bzw. S2
- Outer Union füllt R1 und R2 mit Attributen und NULL-Werten auf, so dass beide dem Schema S1 ∪ S2 entsprechen
- Dann berechne die normale Union
- Kann also mit strukturell heterogenen Schemata umgehen
- Attribute in S1 und S2 mit identischem Namen sind nur einmal im Resultat
- Aber keine Datenfusion

E. F. Codd: Extending the Database Relational Model to Capture More Meaning. TODS 4(4): 397-434 (1979)

R

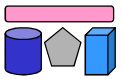
A	B	C
P	1	2
P	2	1
Q	1	2

S

B	D
2	U
3	V

$R \uplus S$

A	B	C	D
P	1	2	⊥
P	2	1	⊥
Q	1	2	⊥
⊥	2	⊥	U
⊥	3	⊥	V



Subsumption

- Subsumption: Tupel t_1 subsumiert t_2, wenn
  - Es die gleichen Attribute hat und
  - t_2 mehr NULL-Werte hat als t_1 und
  - $t_1.A = t_2.A$ für alle Attribute A mit t2.A ≠ NULL
- R↓ ergibt die Tupel aus R, die nicht durch andere Tupel in R subsumiert werden
- Semantik der NULL?
  - Wert unbekannt ("*unknown*")
    - Bsp.: Unbekannter Geburtstag
  - Wert nicht anwendbar ("*inapplicable*")
    - Bsp.: Ehemann/-frau für Ledige
  - Wert zurückbehalten ("*withheld*")
    - Bsp.: Geheime Telefonnummer

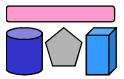
R

p_id	vorname	nachname	alter
1	Peter	Müller	32
1	Peter	Müller	⊥
1	Peter	⊥	⊥
1	Peter	⊥	32
1	Peter	⊥	42
2	Wiebke	⊥	2
2	⊥	Meyer	2

R↓

p_id	vorname	nachname	alter
1	Peter	Müller	32
1	Peter	⊥	42
2	Wiebke	⊥	2
2	⊥	Meyer	2

Ullman: Principles of Database and Knowledge-Base Systems, Volume II., 1989
Garcia-Molina, Ullman, Widom: Database Systems, Prentice Hall, 2002



Minimum Union

$$K \oplus C = (K \uplus C) \downarrow$$

Kunde K

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meyer	⊥

Customer C

p_id	nachname	alter
1	Müller	32
2	Schmidt	⊥
3	⊥	56

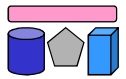
$K \uplus C$

p_id	vorname	nachname	alter

$K \oplus C$

p_id	vorname	nachname	alter

Cesar A. Galindo-Legaria: Outerjoins as Disjunctions,, SIGMOD 1994 Conference



Join

- Ist Join geeignet zur Integration?

Kunde K

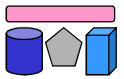
p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meyer	⊥
4	Klaus	Lehmann	28

Customer C

p_id	nachname	alter
1	⊥	32
2	Schmidt	⊥
3	Meier	56
5	Weger	47

$K \bowtie_{p\_id} C$

p_id	K.vorname	K.nachname	C.nachname	K.alter	C.alter
1	Peter	Müller	⊥	32	32
2	Franz	Schmidt	Schmidt	55	⊥
3	Wiebke	Meyer	Meier	⊥	56



Merge

- Vermischt Join und Union zu einem Operator
- COALESCE beseitigt NULLs
- Priorisierung möglich
- Lässt sich mit Hilfe von SQL ausdrücken

```
( SELECT K.p_id, K.vorname,
   Coalesce(K.nachname, C.nachname),
   Coalesce(K.alter, C.alter)
FROM K LEFT OUTER JOIN C ON K.p_id = C.p_id )
UNION
( SELECT C.p_id, K.vorname,
    Coalesce(C.nachname, K.nachname),
    Coalesce(C.alter, K.alter)
FROM K RIGHT OUTER JOIN C ON K.p_id = C.p_id )
```

Customer C

p_id	nachname	alter
1	⊥	32
2	Schmidt	⊥
3	Meier	56
5	Weger	47

Kunde K

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meyer	⊥
4	Klaus	Lehmann	28

C ⊠ K

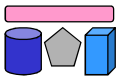
p_id	vorname	nachname	alter

Konfliktlösung

- NULL-Werte nur ein (einfacher) Teil der Konfliktlösung
  - Allgemein: Konfliktlösungsfunktion
- Idee zur Implementierung in SQL
  - Duplikatfindung als Gruppierung
  - Konfliktlösung mit Aggregatfunktionen

$$f(x,y) := \begin{cases} \perp & if\ x = \perp\ and\ y = \perp \\ x & if\ x \neq \perp\ and\ y = \perp \\ y & if\ x = \perp\ and\ y \neq \perp \\ g(x,y) & else \end{cases}$$

Min, Max, Sum, Count, Avg, StdDev	Standard Aggregationsfunktionen
Random	Zufallswahl
First, Last	Nimmt ersten/letzten Wert, reihenfolgeabhängig
Longest, Shortest	Nimmt längsten/kürzesten Wert
Choose(source)	Quellenauswahl
ChooseDepending(col, val)	Wahl abhängig von val in col
Vote	Mehrheitsentscheid
Coalesce	Nimmt ersten nicht-null Wert
Group, Concat	Gruppiert, fügt zusammen
MostRecent	Nimmt aktuellsten Wert
MostAbstract, MostSpecific	Benutzt eine Taxonomie
….	….



Gruppierung/Aggregation zur Integration (1)

- Allgemeines Vorgehen
  - Outer-Union auf alle Quellen
  - Mit einer SQL GROUP BY Anfrage umschließen
    - Gruppierung nach “Duplikat-Id”
  - Aggregat-Funktionen als Konfliktlösungsfunktion für alle anderen Attribute
  - Konfliktlösungsfunktion manuell bestimmen

$K \uplus C$

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meyer	55
1	Peter A.	Müller	32
2	⊥	Schmidt	⊥
3	⊥	Meier	56

p_id	vorname	nachname	alter
1			
2			
3			

Längster String

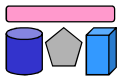
größter Wert

C ist bevorzugte Quelle

```
SELECT p_id, MAXLEN(vorname), CHOOSE(nachname,C), MAX(alter)
FROM   K ⊎ C
GROUP BY  p_id
```

Gruppierung/Aggregation zur Integration (2)

- Vorteile
  - Eleganter Rahmen
  - Effizient (durch Sortierung)
  - Simpel, kurz, deklarativ
- Nachteile
  - Outer Union meistens nicht implementiert
    - Kann durch Sichten simuliert werden
  - Konfliktresolution beschränkt auf eingebaute Aggregatfunktionen
    - MAX, MIN, AVG, VAR, STDDEV, SUM, COUNT
    - User-defined aggregate functions?
  - Duplikaterkennung nur nach Gleichheit
    - Besser: GROUP BY similar(id,name,birthdate)
    - User-defined grouping functions?



Zusammenfassung

- Datenqualität entscheidend für Informationssysteme
  - schwieriges Problem für integrierte Informationssysteme
- Object Matching entscheidender Aspekt
  - Erkennung von Instanzen des gleichen Realweltobjekts
  - Viele Algorithmen (u.a. Ähnlichkeitsmaße) und Frameworks
  - Qualität abhängig von Domäne, Parametern, Hintergrundwissen, ...
- Instanzintegration
  - Konfliktlösung für widersprüchliche / fehlende Werte
  - Konfliktlösungsfunktionen zur Definition, welche Attributwerte im fusionierten/aggregierten Ergebnisobjekt auftreten