

# Data Warehousing

## Kapitel 4: Schemaintegration und Data Cleaning

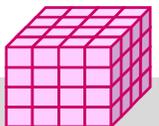
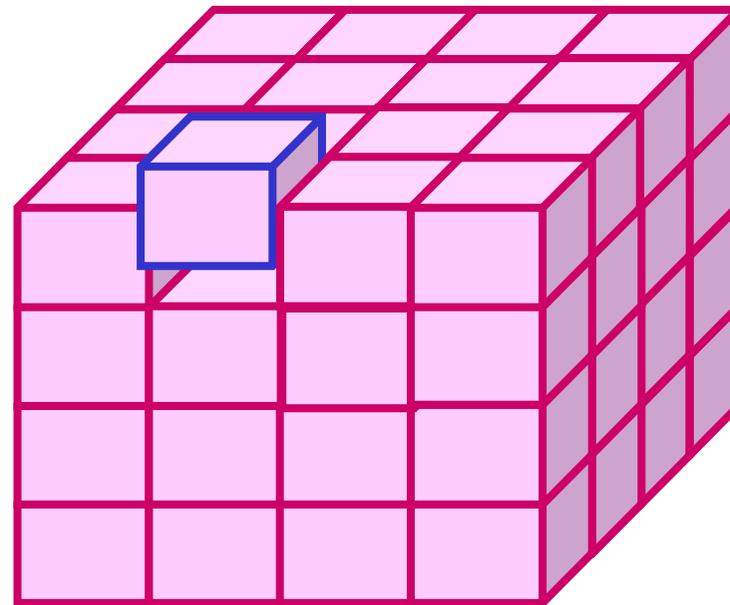
**Dr. Michael Hartung**

Sommersemester 2011

Universität Leipzig

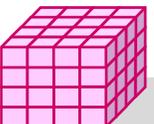
Institut für Informatik

<http://dbs.uni-leipzig.de>



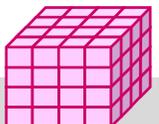
# 4. ETL: Schemaintegration + Data Cleaning

- ETL-Überblick
- Schemaintegration
- (Semi-)automatisches Schema Matching
  - COMA++
- Data Cleaning
  - Probleme
  - Teilaufgaben
  - Objekt-Matching
- Tool-Unterstützung
  - MS SQL-Server 2005
  - MOMA



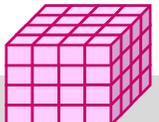
# ETL-Prozess

- Data Warehousing und ETL: materialisierter Ansatz zur Datenintegration
  - Erzeugung einer aggregierten, materialisierten Datenquelle für Online-Analysen
  - komplexer, aufwändiger Integrationsprozess
  - Offline-Durchführung erlaubt höhere Datenqualität gegenüber virtueller Datenintegration (Datentransformation während Query-Verarbeitung)
- *Extraktion*: Selektion eines Ausschnitts der Daten aus Quellen
  - ausgeführt an den entsprechenden Quellen
- *Transformation*: Aufbereitung und Anpassung der Daten an vorgegebene Schema- und Qualitätsanforderungen
  - ausgeführt im temporären Arbeitsbereich (Data Staging Area)
- *Laden*: physisches Einbringen der Daten aus Arbeitsbereich in das Data Warehouse, einschließlich evtl. notwendiger Aggregationen

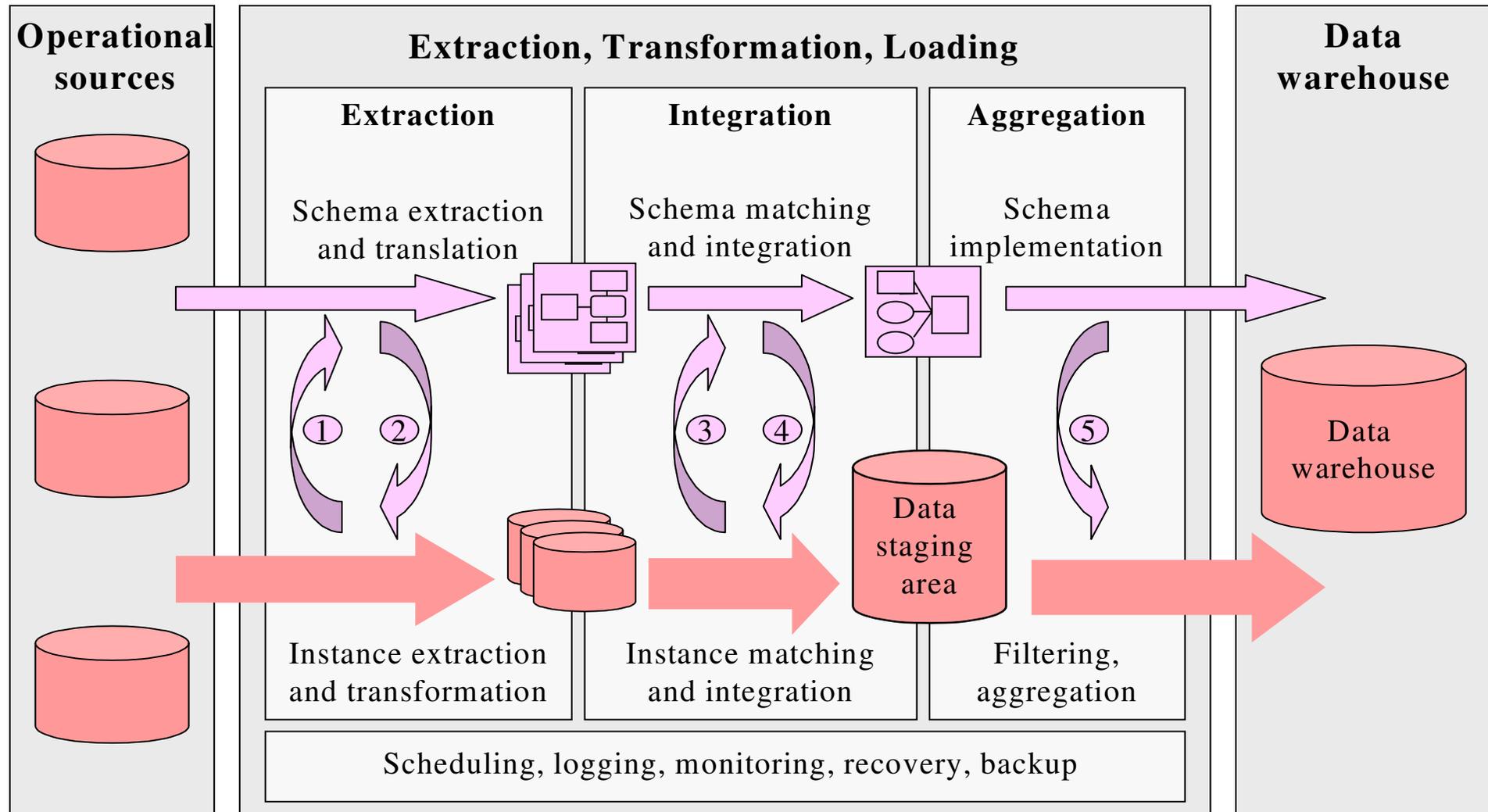


# ETL-Prozess (2)

- Aufwändigster Teil des Data Warehousing
  - Vielzahl von operativen Quellen
  - Heterogenität der Datenquellen (DBMS, Schemata, Daten)
  - Gewährleistung hoher Qualität der Warehouse-Daten
- Entscheidende Rolle im Data Warehousing, da großer Einfluss auf
  - Genauigkeit und Richtigkeit der später durchgeführten Analysen
  - die darauf basierenden Entscheidungen: „Garbage In, Garbage Out“

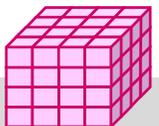


# ETL-Prozess: Ablauf



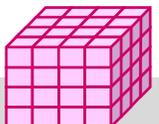
- Legends:**
- Metadata flow
  - Data flow
  - ① Instance characteristics (real metadata)
  - ② Translation rules
  - ③ Instance matching and integration
  - ④ Mappings between source and target schema
  - ⑤ Filtering and aggregation rules

E. Rahm, H. H. Do: *Data Cleaning: Problems and Current Approaches*. IEEE Techn. Bulletin on Data Engineering, 2000



# ETL als Integrationsprozess

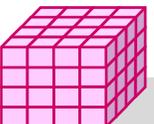
- ETL: Integration auf 2 Ebenen, Schemaintegration und Datenintegration
- Schemaintegration
  - Konstruktion eines Data Warehouse-Schemas aus existierenden Quellschemata
  - Ableitung von Korrespondenzen zwischen dem Data Warehouse-Schema und existierenden Quellschemata: *Schema Matching*
- Datenintegration / Data Cleaning
  - Transformation heterogener Daten in die einheitliche, durch das Data Warehouse-Schema vorgeschriebene Repräsentation
  - Entdeckung und Behebung von Datenqualitätsproblemen
  - Entdeckung äquivalenter Objekte/Sätze (Korrespondenzen auf Instanzenebene): *Objekt-Matching* / Duplikaterkennung



# Schemaintegration - Anforderungen

- Minimalität
  - keine Redundanz im integrierten Schema
  - Abbildung mehrerer gleicher/ähnlicher Konzepte in lokalen Schemata auf ein Konzept im integrierten Schema
- Korrektheit
  - Äquivalenz der im integrierten Schema enthaltenen Informationen mit denen in den lokalen Schemata
  - Konsistenz der während der Integration ergänzten Informationen, z.B. Beziehungen zwischen Konzepten im integrierten Schema
- Verständlichkeit

Vollständigkeit (Beibehaltung aller Informationen aus Quellschemas) ?



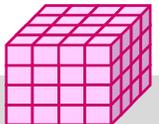
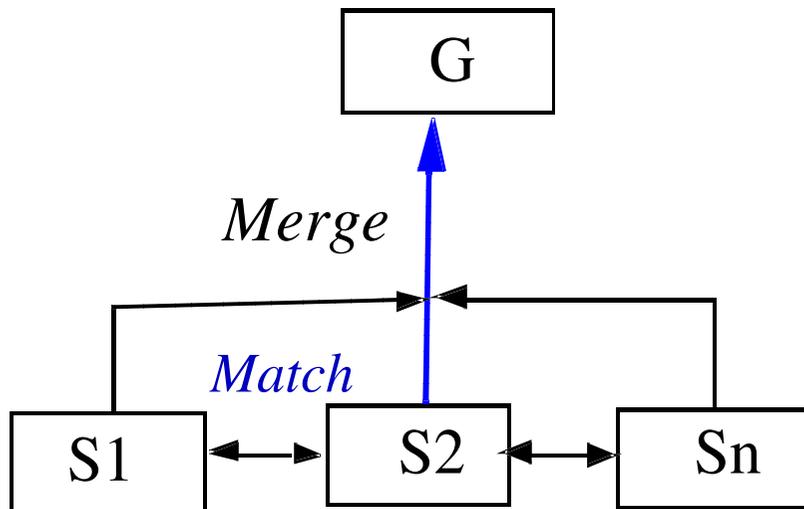
# Schemaintegration (2)

- Probleme der Schemaintegration
  - Heterogenität der Schemarepräsentationen, z.B. relational (SQL), XML, Entity-Relationship (ER), objekt-orientiert (UML), ...
  - Semantische Heterogenität der Schemaelemente (Namenskonflikte, strukturelle Konflikte)
- Alternativen
  - Bottom-Up-Schemaintegration
  - Top-Down-Schemaintegration

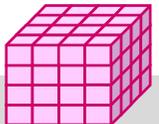
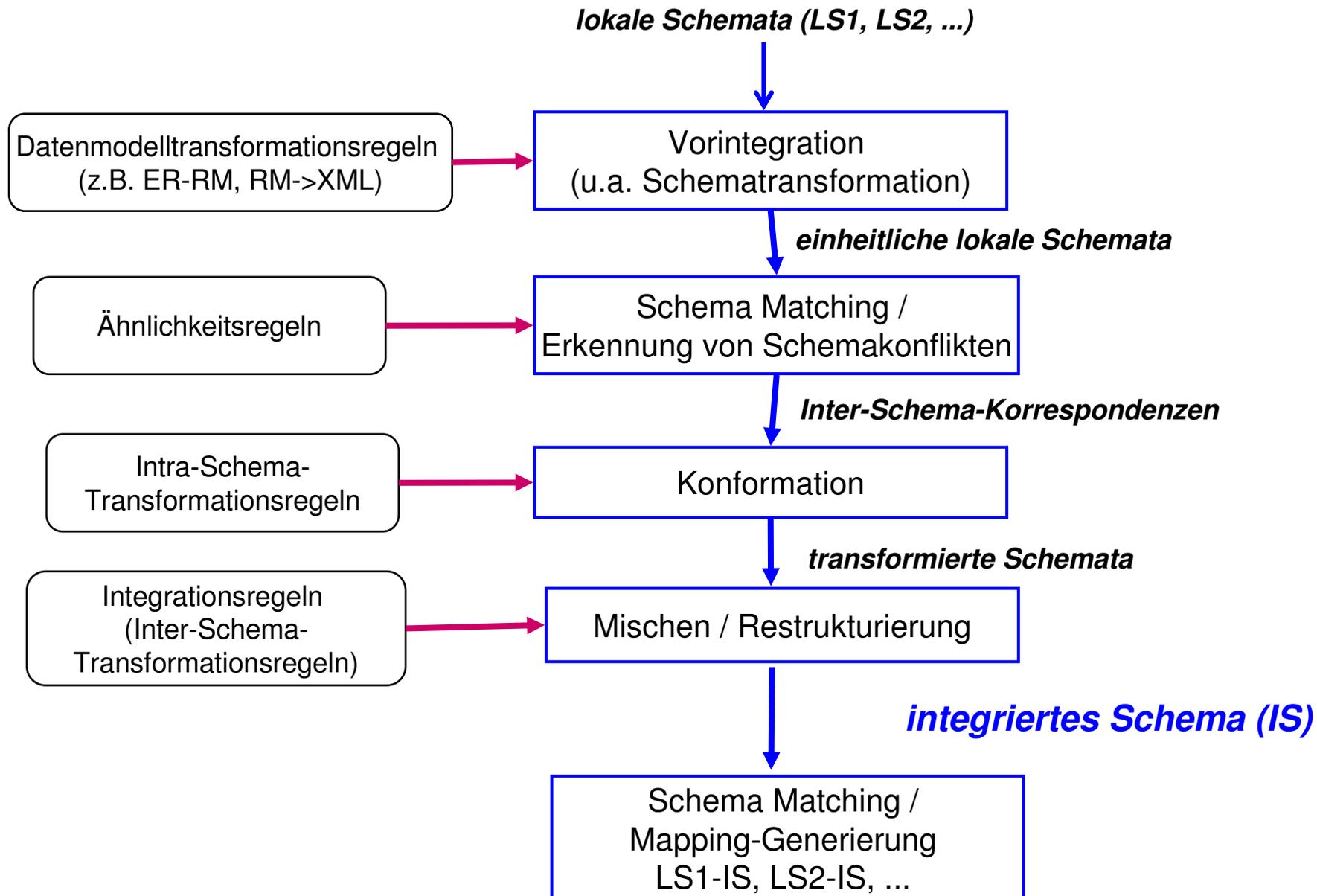


# Bottom-Up-Integration (*Global as View*)

- vollständiges Mischen aller Source-Schemata in globales Schema
- setzt Abgleich zwischen Source-Schemas voraus, insbesondere Bestimmung von Korrespondenzen / Konflikten
- globales Schema entspricht gemeinsamer Sicht (View) auf die zugrundeliegenden Quellen
- neue Quelle ändert globales Schema

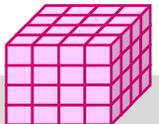
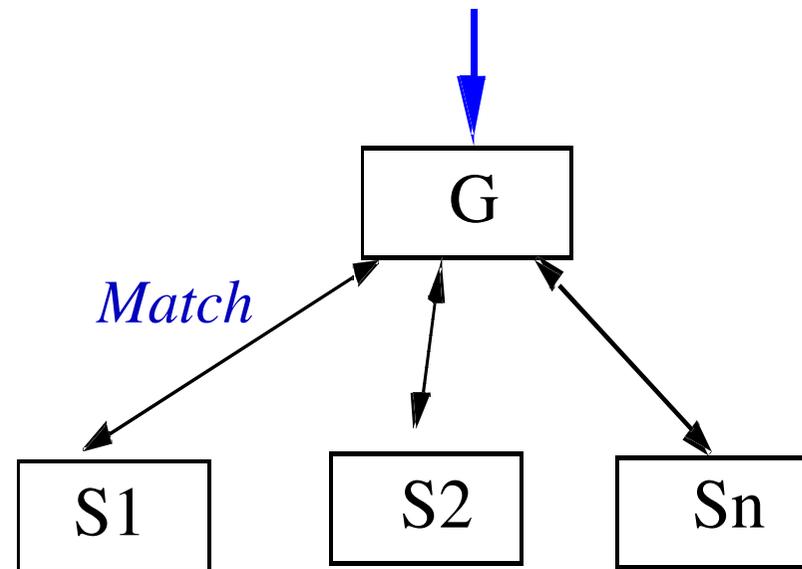


# Bottom-Up-Schemaintegration (2)

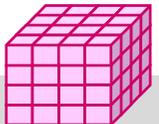
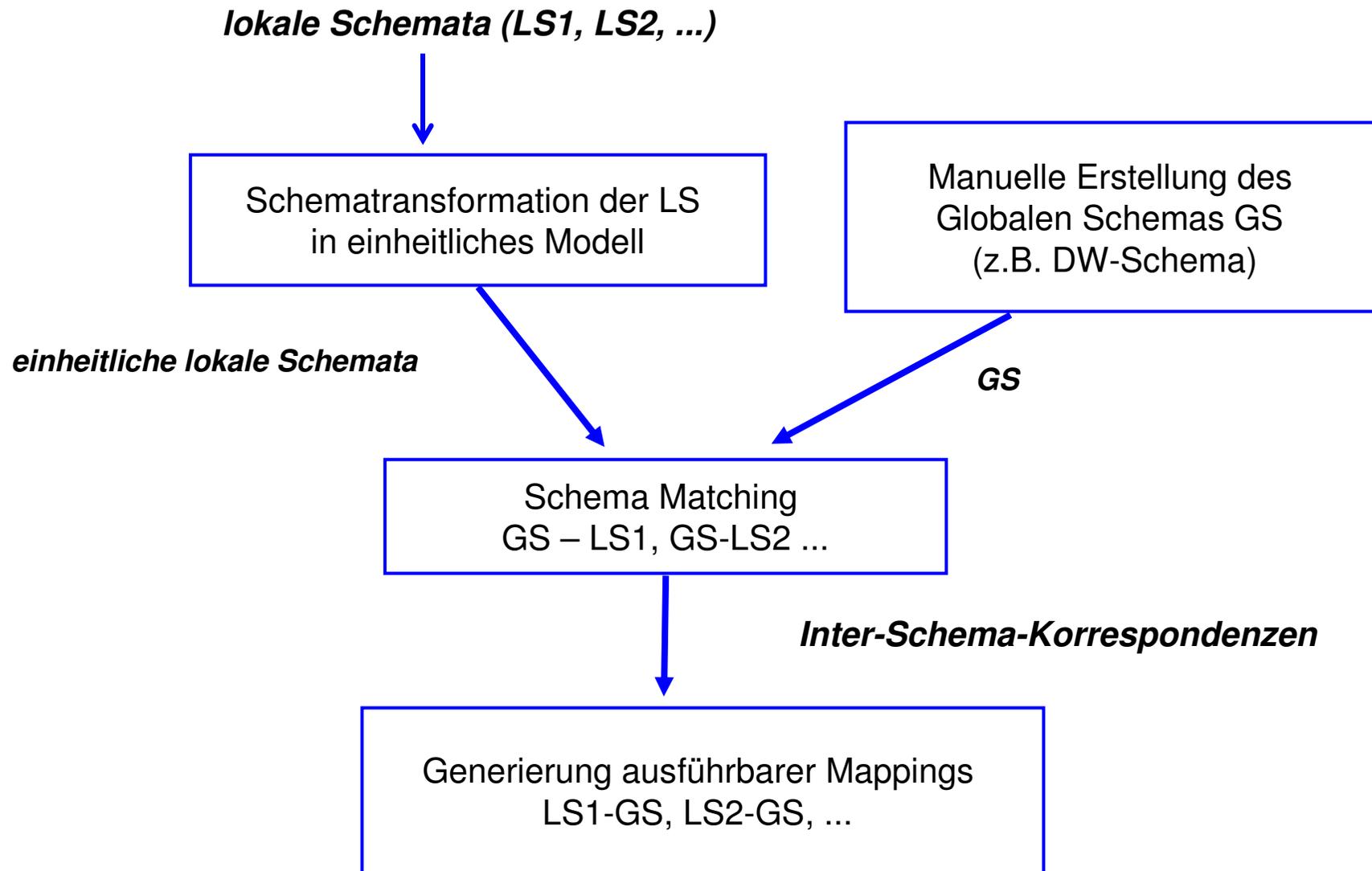


# Top-Down-Integration (*Local as View*)

- globales Schema  $G$  ist vorgegeben
- jede Source  $S$  wird unabhängig von anderen Sources mit globalem Schema abgeglichen, d.h. ein Mapping  $G - S$  erstellt (Mapping beschreibt Inhalt der Quelle)
- aufwändige Query-Verarbeitung bei virtueller Integration
- $G$  berücksichtigt i.a. nur Teile der lokalen Schemata



# Top-Down-Schemaintegration (2)



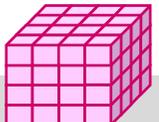
# Namenskonflikte

- **Synonyme:** Repräsentation ein und desselben Konzepts durch unterschiedliche Namen:
- **Homonyme:** Nutzung gleicher Namen für verschiedene Konzepte
- **Hyponyme/Hyperonyme:** Unter-/Oberbegriffe

Mitarbeiter
Name
Adresse

Firma
Name
Adresse

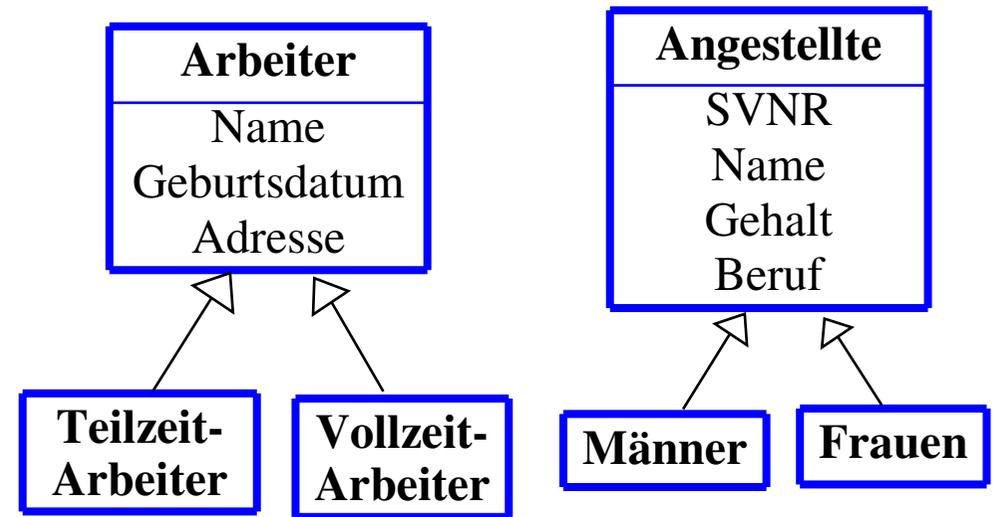
Angestellte
Name
Anschrift



# Strukturelle Konflikte

## ■ Entity vs. Entity

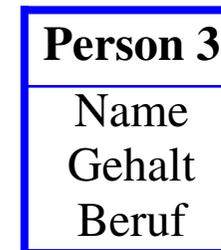
- unterschiedliche Schlüssel
- unterschiedliche Attributmengen, fehlende Attribute
- unterschiedliche Abstraktionsebenen (Generalisierung, Aggregation)
- **unterschiedliche Realitätsauschnitte** (RWS, real world states)
  - disjunkt (disjoint):
  - überlappend (overlaps):
  - enthalten (contains):



Männer



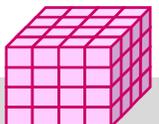
Frauen



> 18 Jahre



Männer > 18



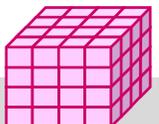
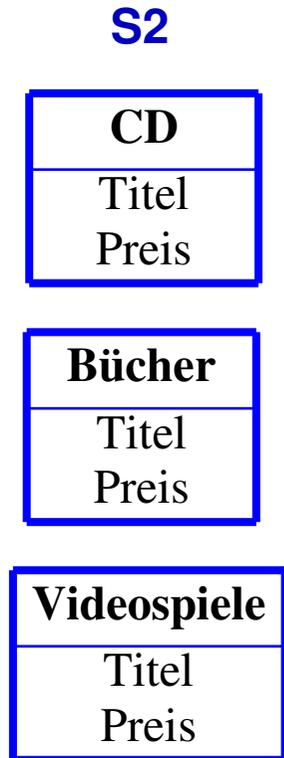
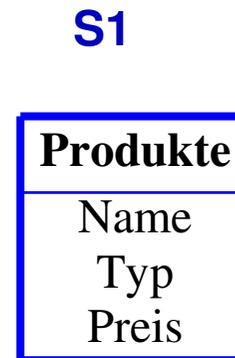
# Strukturelle Konflikte (2)

## ■ Attribut vs. Entity-Konflikte

- Repräsentation von Attributen als eigenständige Entities/Relationen

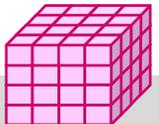
## ■ Attribut vs. Attribut-Konflikte

- unterschiedliche Datentypen  
*Preis (Float) vs. Preis (String)*
- unterschiedliche Detailgrade  
*Name vs. Vorname und Nachname*
- unterschiedliche Einheiten: *\$ vs. Euro*
- unterschiedliche Genauigkeiten: *Tausend Euro vs. Euro*
- unterschiedliche Integritätsbedingungen, Wertebereiche, Default-Werte ...  
*Alter > 18 vs. Alter > 21*
- unterschiedliche Behandlung von Nullwerten
- unterschiedliche Verwaltung der referentieller Integrität



# Behandlung von Konflikten

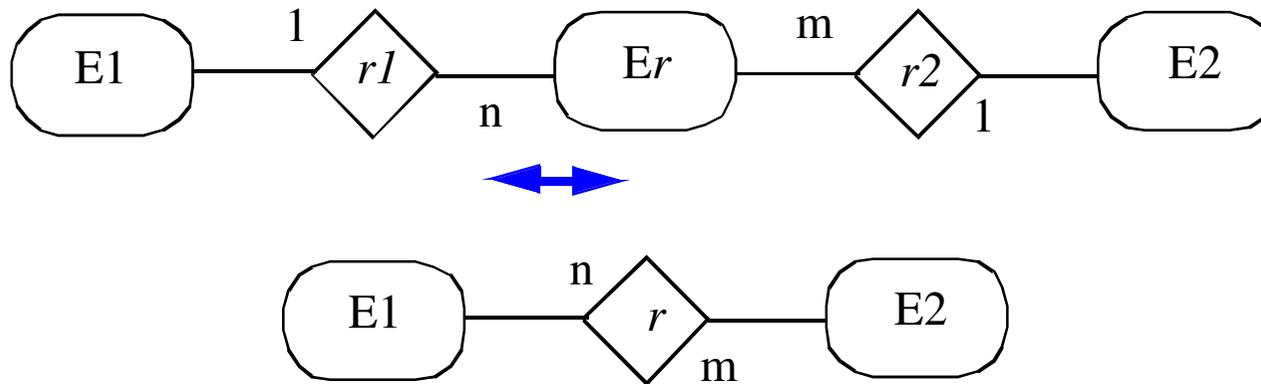
- **Konflikterkennung: Vergleich der Schemata**
  - Identifikation der ähnlichen/gleichen in den Schemata enthaltenen Information
  - Identifikation verschiedener Strukturen, die ähnliche Informationen repräsentieren
- **Repräsentation der Inter-Schema-Korrespondenzen**
  - Synonyme, Matches: Kunde = Klient
  - Is-A-Korrespondenzen: Angestellte is-a Person
  - RWS-Korrespondenzen: RWS(Produkte) contains RWS(Bücher)
- **Behebung von Schemakonflikten v.a. bei Bottom-Up-Integration (Merge) erforderlich**
  - Umbenennungen zur Behebung von Namenskonflikten
  - Schemakonformation und –restrukturierungen
  - Top-Down-Integration: Spezifikation notwendiger Transformationen bei Mapping-Erzeugung



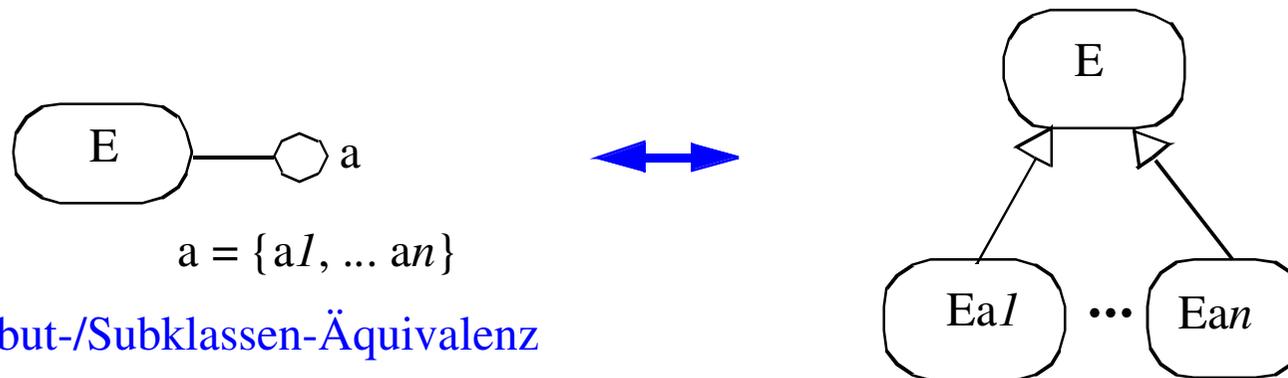
# Schema-Konformationstransformationen



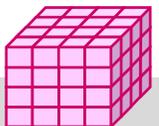
Entity-/Attribut-Äquivalenz



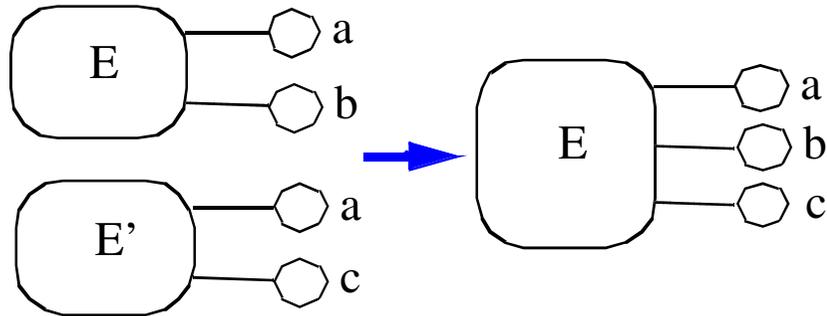
Entity-/Relationship-Äquivalenz



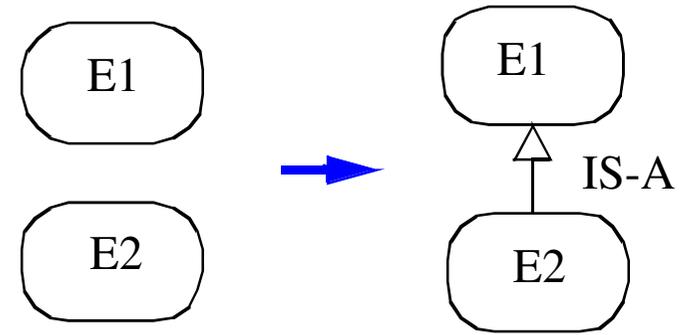
Attribut-/Subklassen-Äquivalenz



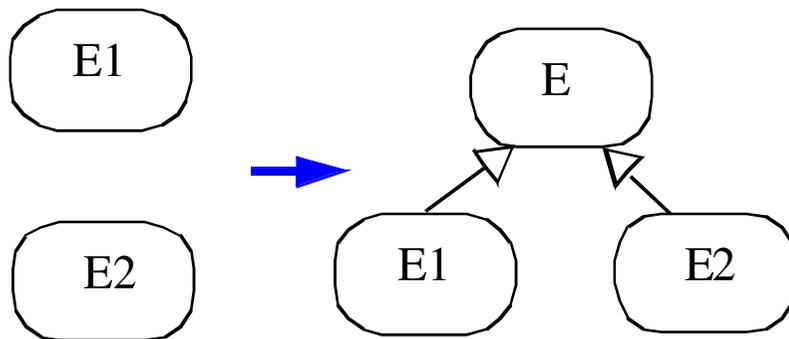
# Schema-Merging-Transformationen



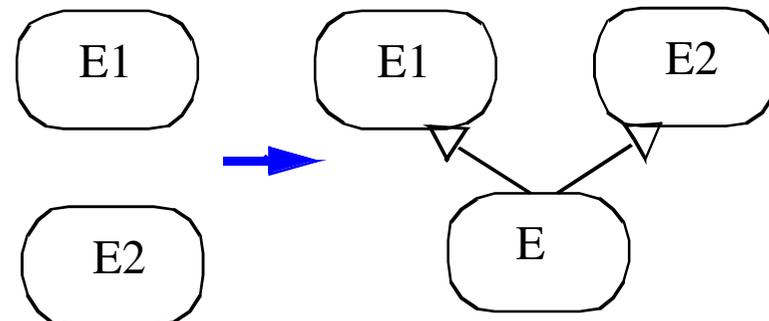
Vereinigung der Attribute



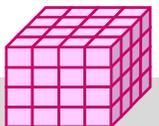
Einführung einer IS-A-Beziehung  
*RWS(E1) contains RWS(E2)*



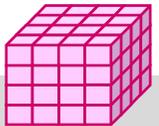
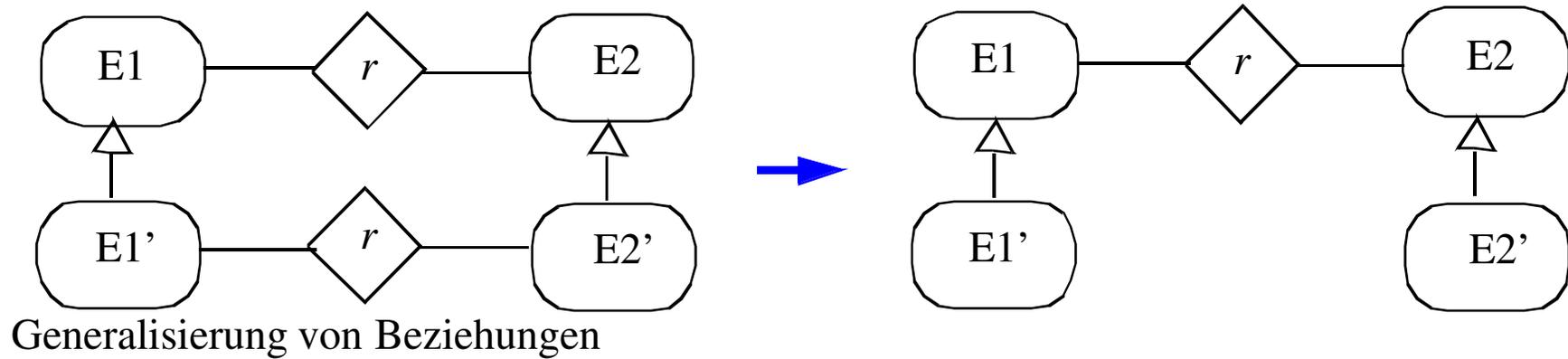
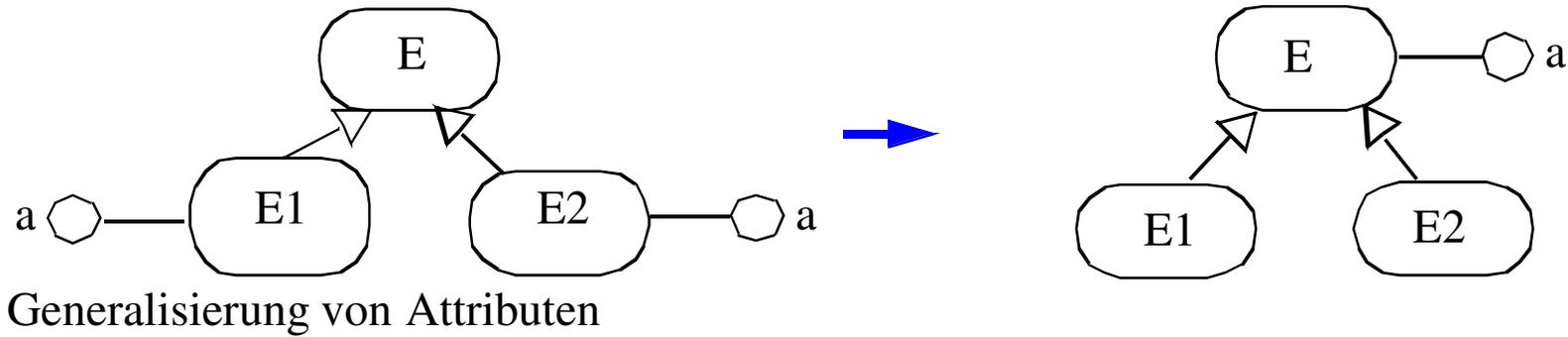
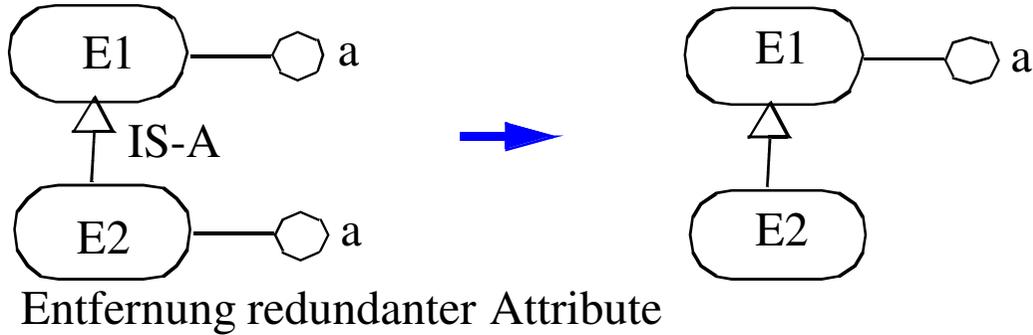
Einführung einer Superklasse  
*RWS(E1) disjoint RWS(E2)*



Einführung einer Subklasse  
*RWS(E1) overlaps RWS(E2)*

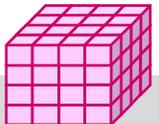


# Schema-Restrukturierungstransformationen



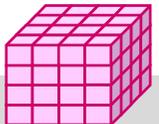
# Automatisierungsbedarf

- bisherige Schemaintegrationsansätze weitgehend manuell
  - nutzerdefinierte Korrespondenzen und Konfliktbehandlung
  - Nutzung spezifischen Schema-/Domain-Wissens
  - aufwändig / fehleranfällig vor allem für größere Schemata
  - nicht skalierbar auf viele Schemata
  - Hoher Anpassungsaufwand bei Schemaänderungen
- Skalierbarkeit erfordert semi-automatische Lösungen / Tools!
  - Vollautomatische Lösungen aufgrund semantischer Heterogenität nicht möglich
  - Namensproblematik (Synonyme, Homonyme)
  - begrenzte Mächtigkeit von Metadaten / Schemasprachen
- (Teil-)Automatisches Schema-Matching
  - v.a. für große Schemata wichtig
  - Nutzer-Feedback notwendig, jedoch im begrenzten Umfang



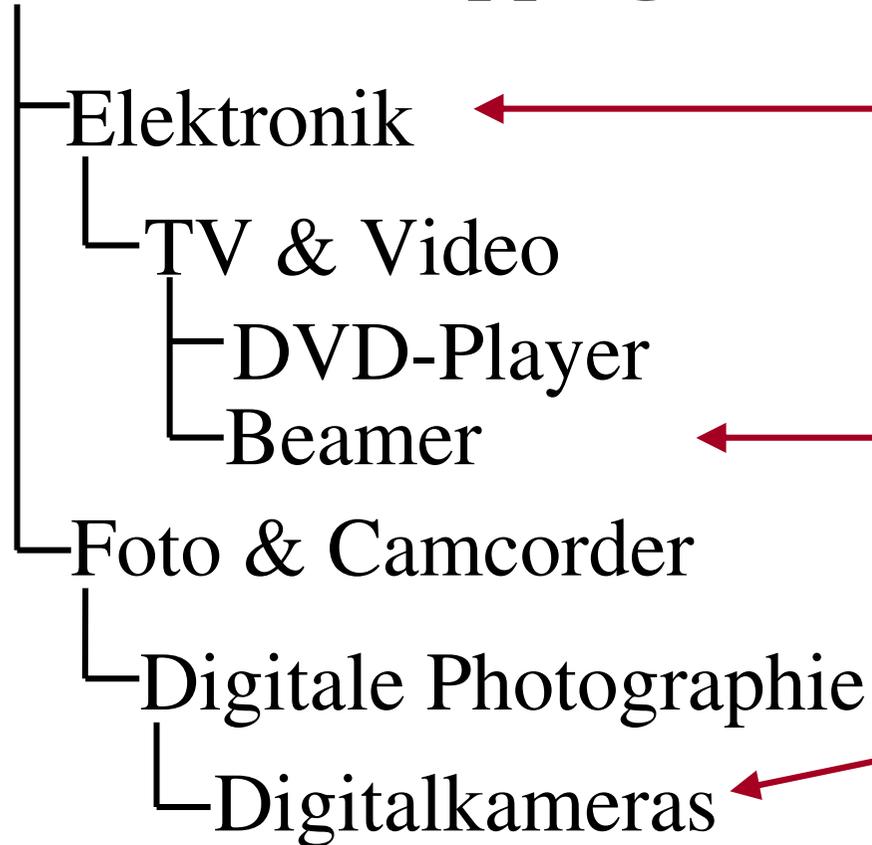
# Schema Matching

- Finden semantischer Korrespondenzen zwischen 2 Schemas
  - DB-Schemas, XML-Schemas, Ontologien, ...
- Kritischer Schritt in zahlreichen Applikationen
  - Datenintegration: Data Warehouses, Mediatoren, P2P
  - E-Business: XML Message Mapping; Katalogintegration
  - Semantic Web: Ontology Matching
- Input:
  - 2 Schemas  $S_1$  and  $S_2$
  - Evtl. Dateninstanzen zu  $S_1$  und  $S_2$
  - *Hintergrundwissen*
- Output: Mapping zwischen  $S_1$  und  $S_2$

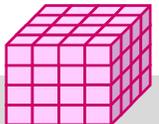


# Match-Beispiel: Produktkataloge

## Yahoo.de Shopping



## Amazon.de



# Manuelle Mapping-Definitionen

## ■ Beispiel: PowerMart

The screenshot displays the Informatica PowerMart Designer interface for defining a manual mapping. The main workspace shows the following components:

- Source Definition (movie (SYBASE)):**

K.	Name	Datatype
	mid	int
	title	varchar
	year	int
	mv_type	varchar
	original	int
- SQL JoinSQ (Source Qualifier):**

Name	Datatype
mid	integer
bemtype	string
text	string
bem	small inte
mid1	integer
strinn	string
- Target Definition (movie\_orig\_rem (INFORMIX)):**

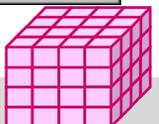
K.	Name	Datatype
	mid	Integer
	title	Varchar
	year	Integer
	mv_type	Varchar
	remark	Varchar
- Filter (isAKA):**

Name	Datatype
mid	integer
title	string
year	integer
mv_type	string
original	integer
- Target Definition (movie\_aka (INFORMIX)):**

K.	Name	Datatype
	mid	Integer
	title	Varchar
	year	Integer
	mv_type	Varchar
	original	Integer

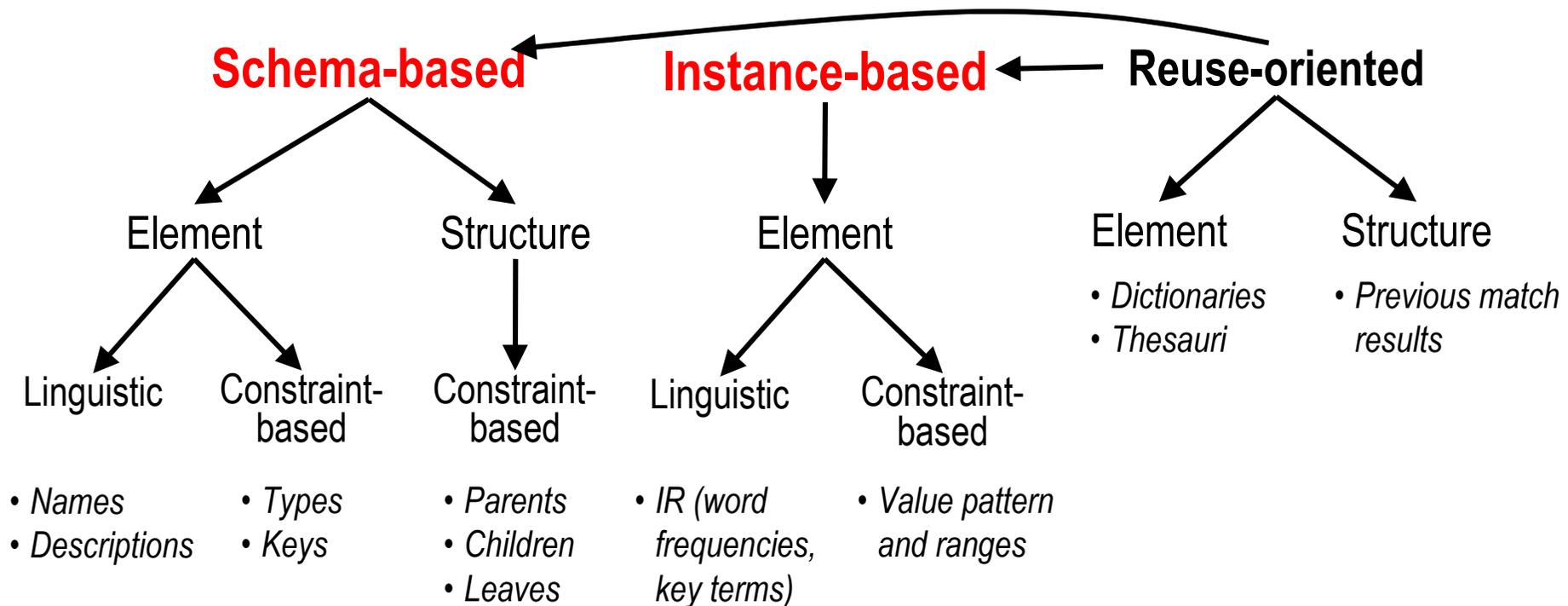
The log window at the bottom shows the following messages:

```
11/02/98 15:23:00 *** Saving... Repository repositorydb, Version 1.0.0, Folder SHARED
Source EINGABE_DATE11 inserted.
Source EINGABE2_DATE11 inserted.
Source EINGABE3_DATE11 inserted.
Target q68t035 inserted.
Validating transformations of mapping MappingCocolMini...
...transformation validation completed with no errors.
```



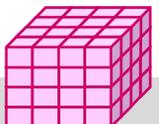
# Automatische Match-Ansätze\*

## ■ Einzelne Ansätze



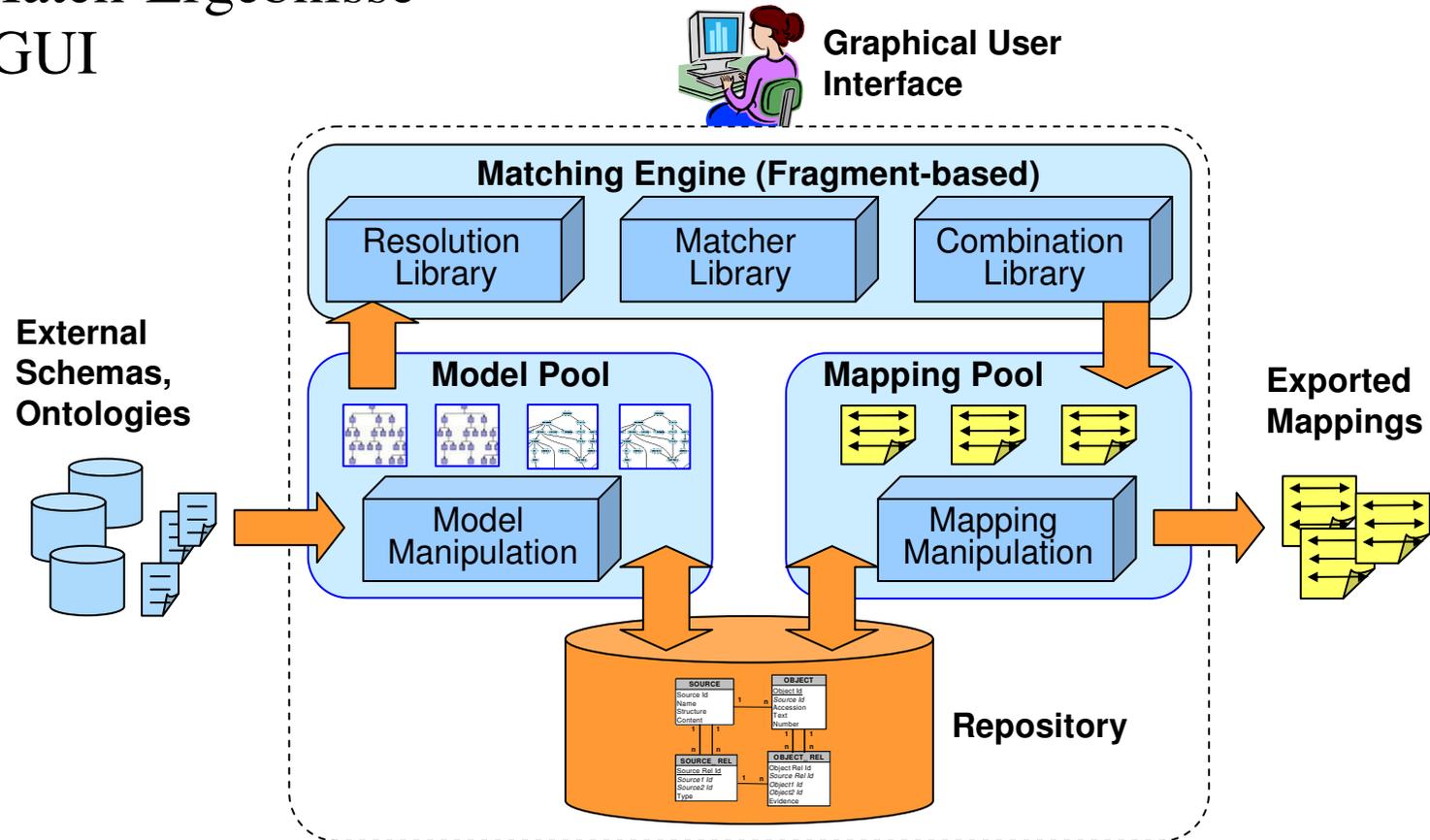
## ■ Kombinerende Ansätze: Hybrid vs. Composite

\* Rahm, E., P.A. Bernstein: *A Survey of Approaches to Automatic Schema Matching*. VLDB Journal 10 (4), 2001



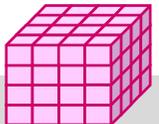
# COMA++: Generische Match-Plattform\*

- Unterstützt XML, relationale Schemas und OWL-Ontologien
- Composite-Ansatz mit flexibler Konfiguration von Matchern
- Match-Strategien für große Schemas: Fragment-basiert / Wiederverwendung vorhandener Match-Ergebnisse
- Umfassendes GUI



\*Do, H.H., E. Rahm: *COMA - A System for Flexible Combination of Schema Matching Approaches*. VLDB 2002

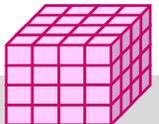
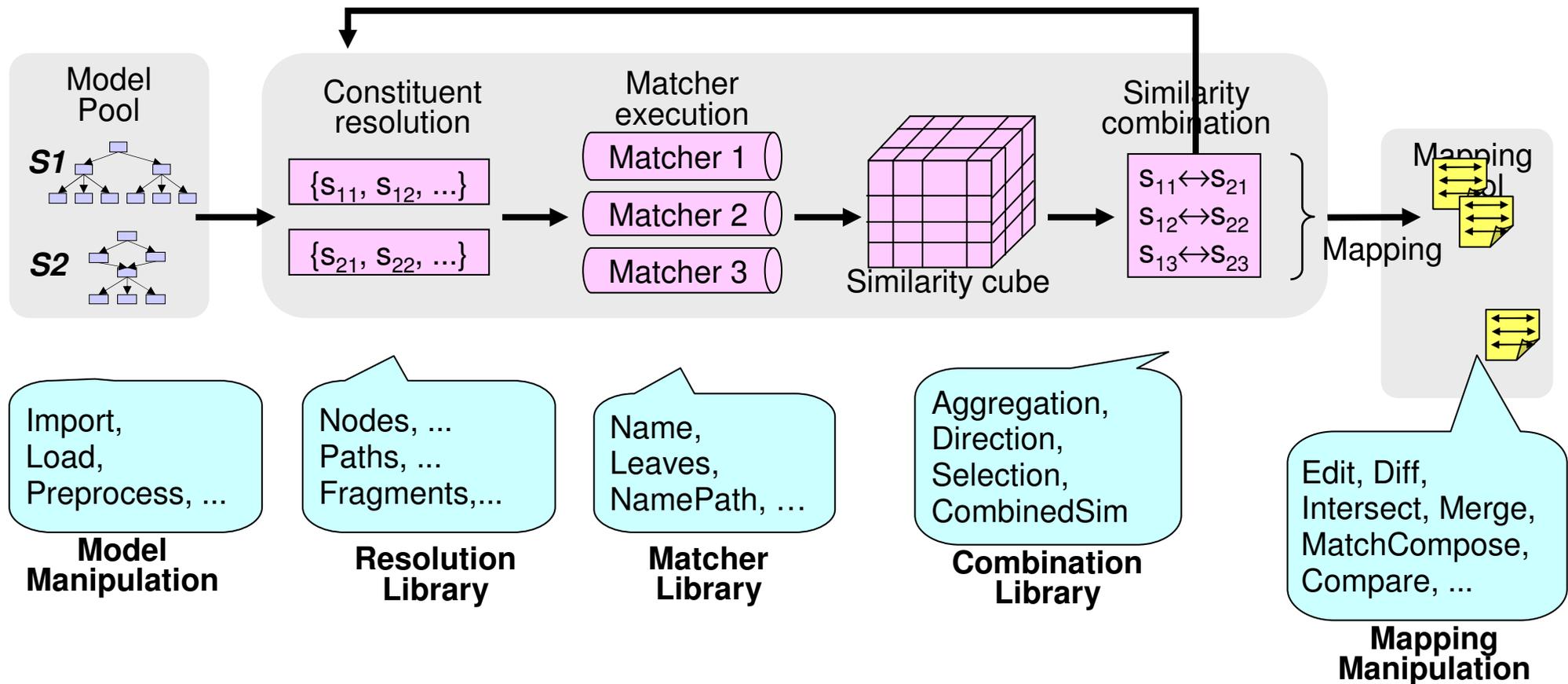
Aumüller D., H.-H. Do, S. Massmann, E. Rahm: *Schema and Ontology Matching with COMA++*. Sigmod 2005



# Match-Verarbeitung

- Ausführung verschiedener Match-Algorithmen
- Manipulation/Kombination der erzeugten Mappings

Match Iteration



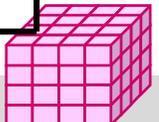
# Matcher-Bibliothek

## ■ Basis-Matcher:

- String-Matcher: Synonym, Trigram, Affix, EditDistance
- Type-Matcher
- Taxonomie-Matcher
- Reuse-Matcher: Wiederverwendung von Mappings

## ■ Hybrid-Matcher: feste Kombination anderer Matcher

<i>Name</i>	<i>Constituents</i>	<i>Matchers/Sim measures</i>	<i>Combination</i>
Name	Name tokens	Synonym/Taxonomy, Trigram	Avg, Both, Max1, Avg
NameType	Node	Name, Type	Wgt(0.7,03), Both, Max1, Avg
NameStat	Node	Name, Statistics	
Children	Children	NameType	Avg, Both, Max1, Avg
Leaves	Leaves	NameType	
Parents	Parents	Leaves	
Siblings	Siblings	Leaves	
NamePath	Ascendants	Name	



# Schema-Beispiel

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:complexType name="PO1">
  <xsd:sequence>
    <xsd:element name="DeliverTo" type="Address"/>
    <xsd:element name="BillTo" type="Address"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Address">
  <xsd:sequence>
    <xsd:element name="Street" type="xsd:string"/>
    <xsd:element name="City" type="xsd:string"/>
    <xsd:element name="Zip" type="xsd:decimal"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

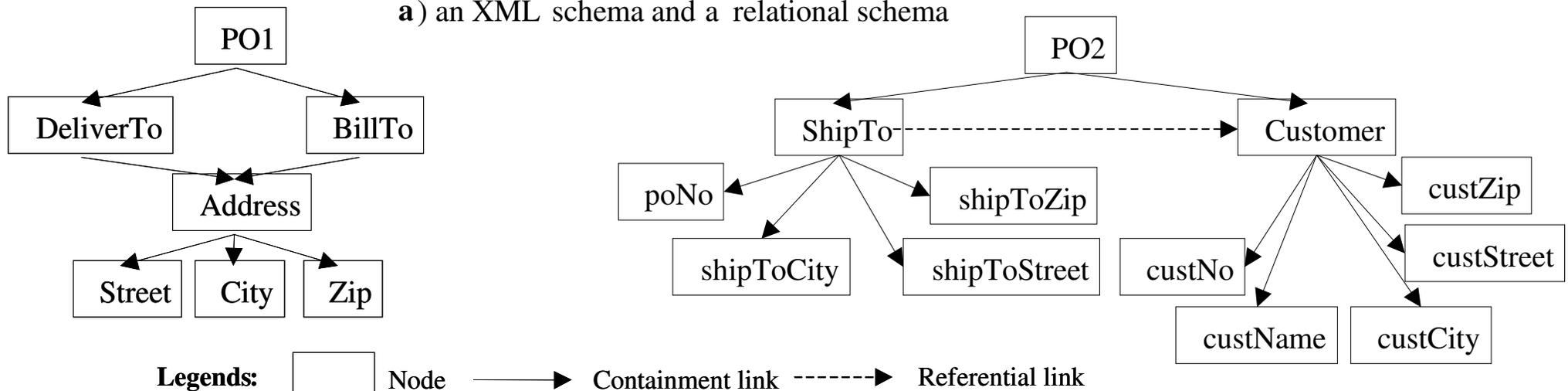
```

CREATE TABLE PO2.ShipTo (
  poNo      INT,
  custNo    INT REFERENCES PO2.Customer,
  shipToStreet VARCHAR(200),
  shipToCity  VARCHAR(200),
  shipToZip   VARCHAR(20),
  PRIMARY KEY (poNo)
);

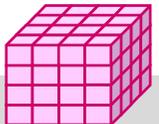
CREATE TABLE PO2.Customer (
  custNo    INT,
  custName  VARCHAR(200),
  custStreet VARCHAR(200),
  custCity  VARCHAR(200),
  custZip   VARCHAR(20),
  PRIMARY KEY (custNo)
);

```

a) an XML schema and a relational schema

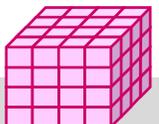


b) Their corresponding graph representation



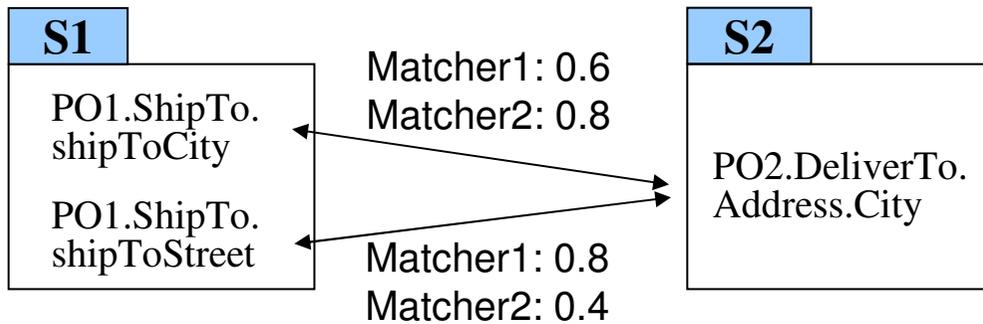
# String-Matching: Beispiel

- „Street“ vs. „ShipToStreet“
- Edit Distance:
  - Distanz entspricht Anzahl notwendiger Einfüge-, Lösch- bzw. Änderungsschritte auf Zeichen bezogen auf maximale Stringlänge
  - Ähnlichkeit:  $1 - \text{Distanz}$
  
- n-Gram (z.B.  $n=3$  / Trigram):
  - Bestimme alle Zeichensequenzen der Länge  $n$  (n-grams)
  - Ähnlichkeit entspricht dem Anteil übereinstimmender n-grams

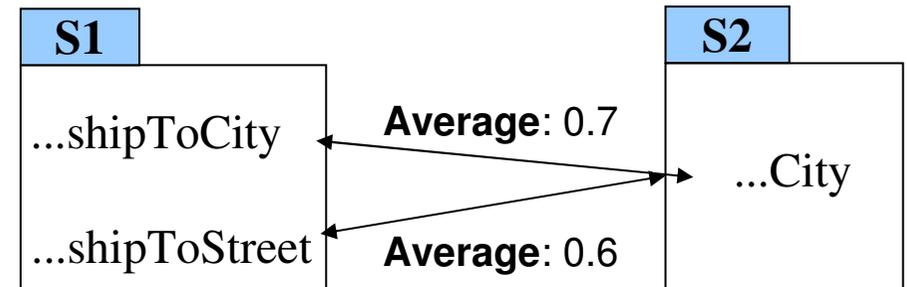


# Kombination von Match-Ergebnissen: Beispiel

## 1. Matcher execution



## 2. Aggregation



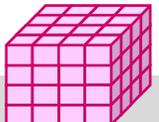
## 3. Selection

**Max1**

<i>S2 elements</i>	<i>S1 elements</i>	<i>Sim</i>
...City	...shipToCity	0.7

**Threshold(0.5)**

<i>S2 elements</i>	<i>S1 elements</i>	<i>Sim</i>
...City	...shipToCity	0.7
...City	...shipToStreet	0.6

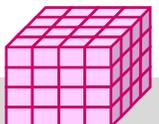
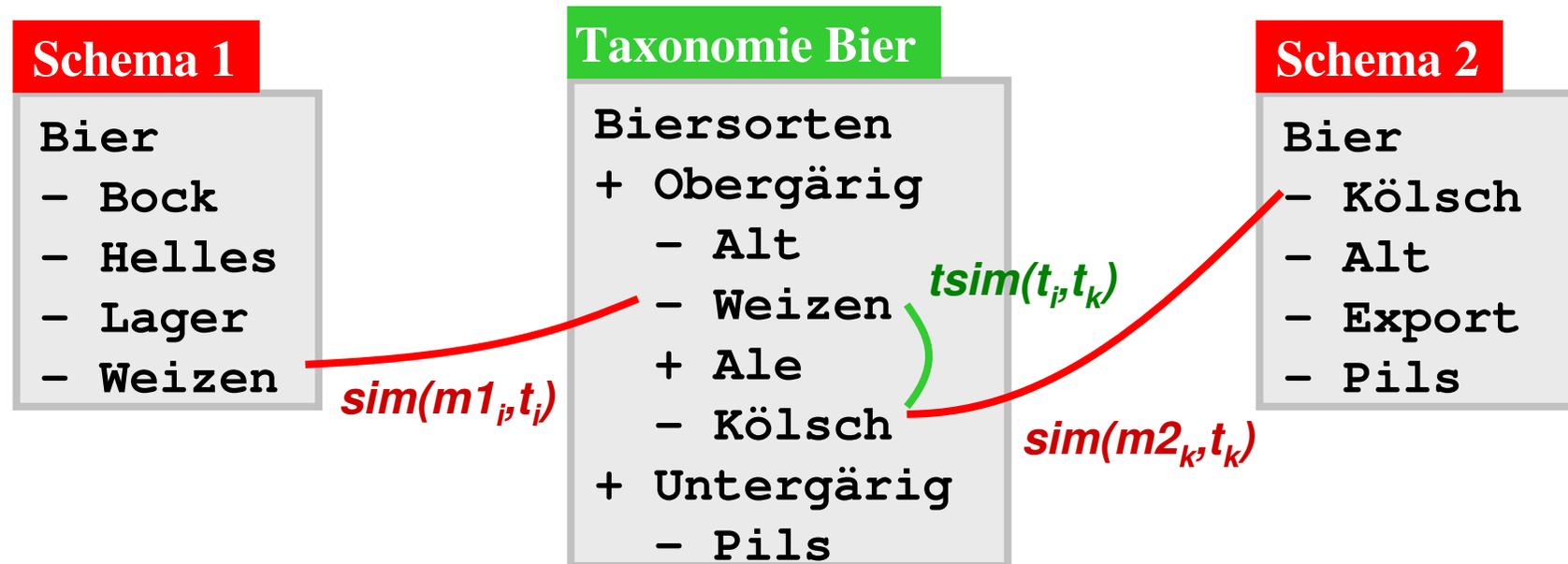


# Taxonomie-Matcher

- Taxonomie als Referenz zwischen Schemas/Modellen

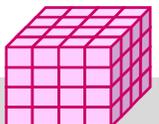
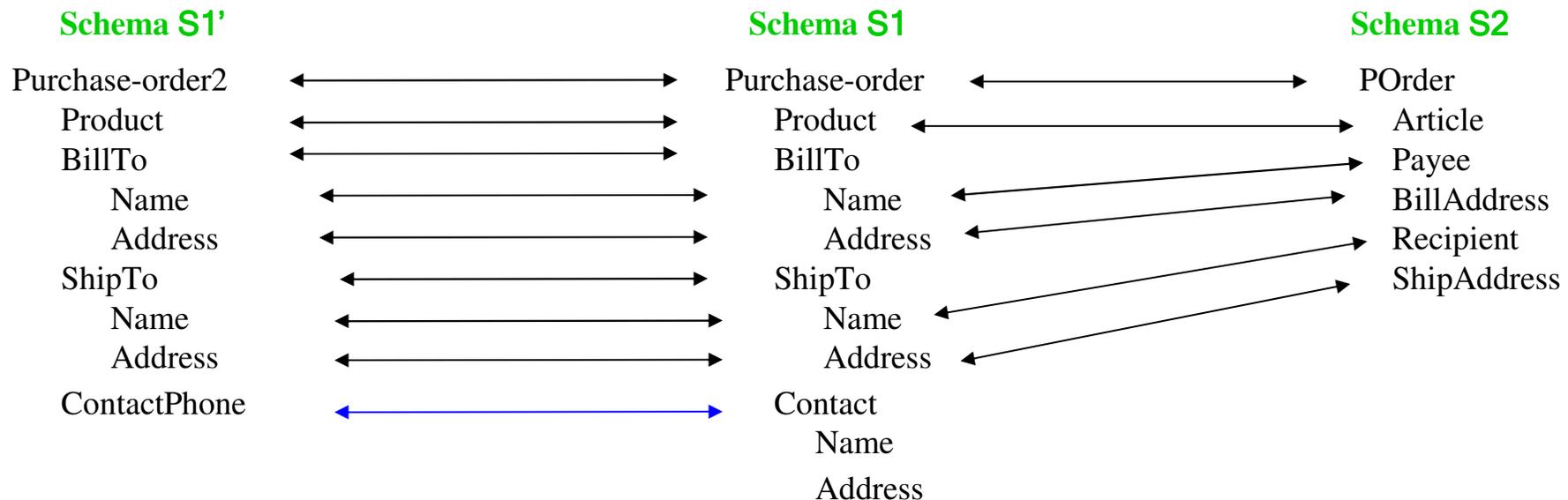
$$\text{sim}(\text{Weizen}, \text{Kölsch}) = 0.8$$

- Ähnlichkeit zweier Schema-Elemente  $\rightarrow$  Kombination aus  $\text{sim}(m,t)$  und  $\text{tsim}(t,t)$ :
  - **lexikalischen Ähnlichkeiten** der Schema-Elemente *mit* der Taxonomie
  - **semantische Ähnlichkeit** durch Distanz der Begriffe *innerhalb* der Taxonomie (verschiedene Heuristiken denkbar)



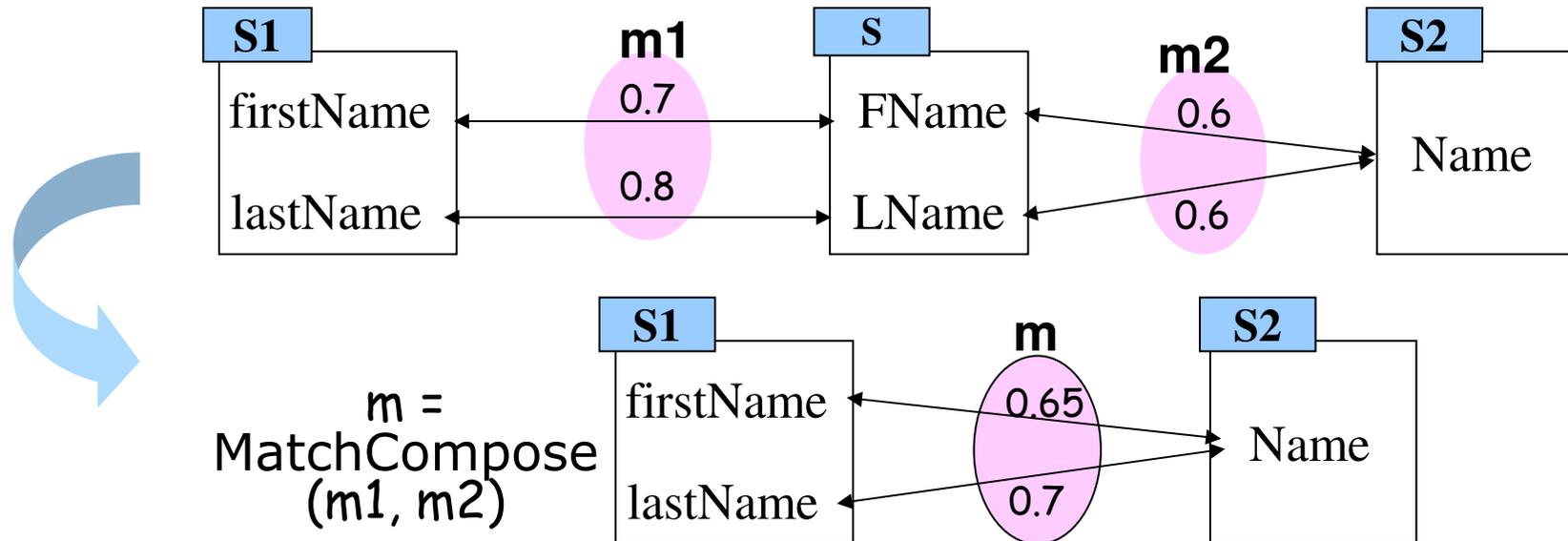
# Wiederverwendung (Reuse)

- Nutzung von Hilfsquellen
  - Nutzerspezifizierte Synonymtabellen
  - Allgemeine/Domänenspezifische Vokabulare, Wörterbücher
  - Gemeinsame Ontologien
- Nutzung bereits bestätigter Match-Ergebnisse für ähnliche Match-Probleme
  - Speichern von Schemas und Mappings in Repository
  - Besonders vorteilhaft für Abgleich neuer Schema-Versionen (Schema-Evolution)
- Beispiel: Wiederverwendung des vorhandenen (bestätigten) Mappings S1—S2 zur Lösung des neuen Match-Problems S1'—S2

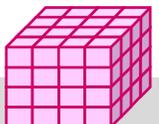


# Wiederverwendung von Mappings

- **MatchCompose**-Operation: Ähnlichkeit/Match als transitive Beziehung



- **Wiederverwendungsmöglichkeiten für neues Match-Problem S1-S2**
  - Direkte Mappings S1-S2
  - Mapping-Pfade (S1-S3-S2, S2-S4-S5-S1, ...)
  - Nutzung ähnlicher Mappings, z.B. mit unterschiedlichen Schemaversionen



# Coma++ Nutzerschnittstelle

The screenshot shows the Coma++ application window with the following components:

- Menu Bar:** Repository, Source, Target, Matcher, Mapping, Help
- Toolbar:** Includes icons for file operations, execution, and editing.
- Left Sidebar:**
  - Project Tree: Paragon\_Apertum > Mapping1
  - Mapping Info: Mapping: only Contact, Mapping3\_TEST
  - Summary Table:

Name	Mapping1
Total	48
Info	COMA
Operation	SCHEMA
Config	124 COMA 101 DOWNPATHS 114,115...
- Main Panels:**
  - Paragon (XDR) Schema:**
    - Supplier: SupID : String, SupName : String, City : String, Address : String, PostalCode : String, Country : String, County : String
    - BillTo: City : String, Address : String, PostalCode : String, Country : String, County : String
    - ShipTo: City : String, Address : String, PostalCode : String, Country : String, County : String, BranchID : String, CompanyName : String
    - Contact: Department : String, Name : String, Phone : String, Fax : String, EMail : String
    - CurrencyInfo: CurrencyID : String, ExchangeRate : String
  - Apertum (XDR) Schema:**
    - DeliverTo: SupplierReferenceNo : string, BuyerReferenceNo : string, VAT\_RegistrationNo : string
    - Address: PostCode : string, City : string, State : string, CountryCode : string
    - Contact: Name1 : string, Street : string
    - Contact: Title : string, FirstName : string, LastName : string, JobTitle : string, Phone : string, Fax : string, EMail : string
    - InvoiceTo: SupplierReferenceNo : string, BuyerReferenceNo : string, VAT\_RegistrationNo : string
    - Address: PostCode : string

Red lines connect corresponding fields between the two schemas, such as Supplier.SupID to DeliverTo.SupplierReferenceNo and Supplier.Address to DeliverTo.Address.PostCode.

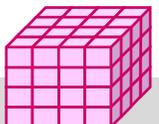
Matching is done.



# Data Cleaning\*

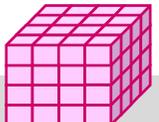
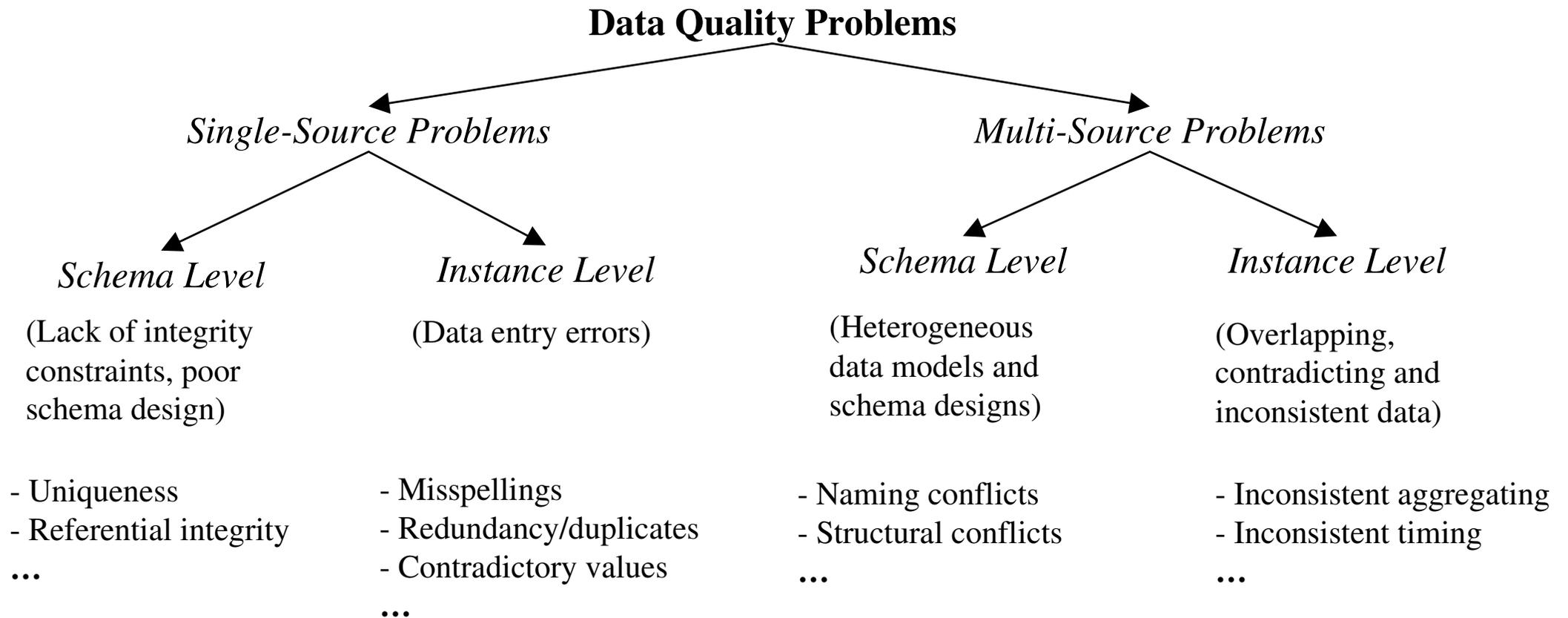
- Datenanalyse
  - Entdeckung von Datenfehlern und -inkonsistenzen
  - manuell bzw. Einsatz von Analyse-Tools
- Definition von Mapping-Regeln und Transformations-Workflows
  - Datentransformationen auf Schemaebene
  - Cleaning-Schritte zur Behandlung von Instanzdaten
  - deklarative Spezifikation erlaubt automatische Generierung von ausführbaren Skripts
- Test / Verifizierung der Transformations-Workflows
  - Korrektheit und Effektivität auf Kopien/Ausschnitt der Daten
- Transformation
  - regelmäßige Ausführung der geprüften Transformationsschritte
- ggf. Rückfluss korrigierter Daten in operative Quellsysteme

\* E. Rahm, H. H. Do: *Data Cleaning: Problems and Current Approaches*.  
IEEE Techn. Bulletin on Data Engineering, Dec. 2000



# Probleme bezüglich Datenqualität

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



# Single-Source Probleme

## ■ Ursachen:

- Fehlen von Schemata (z.B. bei Dateien) und von Integritäts-Constraints
- Eingabefehler
- unterschiedliche Änderungsstände

Name	Adresse	Phone	Erfahrung	Beruf
Peter Meier	Humboldtstr. 12, 04173 Liepzig	9999- 999999	A	Dipl- Informatiker
Schmitt, Ingo	Lessingplatz 1, 98321 Berlin	030- 9583014	M	Dipl.-Inf.
..	...	...	..	..

Multivalue-Feld

Misspelling

Fehlender Wert

Transposition

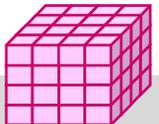
Attributwert-abhängigkeit

Kryptische Werte

Uneinheitliche Bezeichnungen

# Multi-Source-Probleme

- überlappende, widersprüchliche bzw. inkonsistente Daten
  - aufgrund unabhängiger Erzeugung / Speicherung in verschiedenen Quellen
- Hauptproblem: Behandlung überlappender Daten
  - Gängige Bezeichnungen: *Duplikate*, *Merge/Purge-Problem*, *Object Identity Problem*, *Record Linkage* ,
  - Beschreibung einer Instanz der realen Welt durch mehrere Datensätze unterschiedlicher Quellen
  - Oft nur teilweise Redundanz (einzelne Attribute, nur in Teilmenge der Datenquellen) -> Fusion der Instanzen notwendig
- Unterschiedliche Repräsentationen der Instanzdaten
  - versch. Wertebereiche (z.B. *Geschlecht = {1,2}* vs. *Gender = {m,w}*)
  - verschiedene Einheiten (z.B. *Verkauf in EUR* vs. *Verkauf in Tsd.EUR*)
  - verschiedene Genauigkeiten
- unterschiedliche Änderungsstände und Aggregationsstufen der Quelldaten



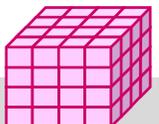
# Multi-Source-Dateninkonsistenzen: Beispiel

*Source1: Customer*

<i>CID</i>	<i>Name</i>	<i>Street</i>	<i>City</i>	<i>Sex</i>
11	Kristen Smith	2 Hurley Pl	South Fork, MN 48503	0
24	Christian Smith	Hurley St 2	S Fork MN	1

*Source2: Client*

<i>Cno</i>	<i>LastName</i>	<i>FirstName</i>	<i>Gender</i>	<i>Address</i>	<i>Phone/Fax</i>
24	Smith	Christoph	M	23 Harley St, Chicago IL, 60633-2394	333-222-6542 / 333-222-6599
493	Smith	Kris L.	F	2 Hurley Place, South Fork MN, 48503-5998	444-555-6666



# Beispiel (2)

<i>CID</i>	<i>Name</i>	<i>Street</i>	<i>City</i>	<i>Sex</i>
11	Kristen Smith	2 Hurley Pl	South Fork, MN 48503	0
24	Christian Smith	Hurley St 2	S Fork MN	1

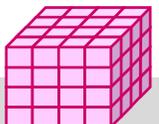
**Source1:  
Customer**

<i>No</i>	<i>LName</i>	<i>FName</i>	<i>Gender</i>	<i>Street</i>	<i>City</i>	<i>State</i>	<i>ZIP</i>	<i>Phone</i>	<i>Fax</i>	<i>CID</i>	<i>Cno</i>
1	Smith	Kristen L.	F	2 Hurley Place	South Fork	MN	48503-5998	444-555-6666		11	493
2	Smith	Christian	M	2 Hurley Place	South Fork	MN	48503-5998			24	
3	Smith	Christoph	M	23 Harley Street	Chicago	IL	60633-2394	333-222-6542	333-222-6599		24

**Customers** (Integrierte und bereinigte Daten)

<i>Cno</i>	<i>LastName</i>	<i>FirstName</i>	<i>Gender</i>	<i>Address</i>	<i>Phone/Fax</i>
24	Smith	Christoph	M	23 Harley St, Chicago IL, 60633-2394	333-222-6542 / 333-222-6599
493	Smith	Kris L.	F	2 Hurley Place, South Fork MN, 48503-5998	444-555-6666

**Source2:  
Client**



# Datenanalyse

- Entdeckung von Fehlern / Verifikation korrekter Werte
- Ableitung von (wirklichen) Metadaten
- Berechnung der Statistiken zu Attributen auf Basis ihrer Instanzen
  - Datentyp, Länge, Maximum und Minimum, Null-, Default-Werte, Kardinalität, ...
  - Ermitteln von Wertebereichen, Häufigkeiten und Mustern von Attributwerte
- Erkennung von Ausreißern, funktionalen Abhängigkeiten

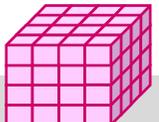
Attribute Values	#occurences
IBM	3000
I.B.M.	360
Intel Bus Mach	213
International Business Machine	36

Instanzwerte	Pattern	Identifizierte Datenkategorie
(978) 555-1212	(nnn) nnn-nnnn	Telefonnummer
036-55-1234	nnn-nn-nnnn	Social Security Number
abc@web.de	aaa@aaa.aa	Email-Adresse
12.03.2008	nn.nn.nnnn	Datum



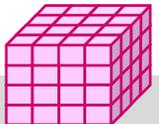
# Behandlung von Single-Source-Problemen

- Definition und Einführung von Standardrepräsentationen
    - einheitliches Format für Datums-/Zeit-Angaben
    - einheitliche Groß/Kleinschreibungsform für Namen / String-Attribute
    - einheitliche Abkürzungen, Kodierungsschemas
  - Bereitstellung von (Konversions-)Tabellen zur expliziten Werteabbildung
- | Legacy Value   | New Value |
|----------------|-----------|
| IBM            | IBM       |
| I.B.M          | IBM       |
| Intel Bus Mach | IBM       |
| ...            | ...       |
- Extraktion von individuellen Werten aus Freiform-Attributen
    - Parsing und Attribut-Splitting, *z.B. Name -> Vorname / Nachname*
    - Reorganisierung der Wortreihenfolge
  - Validierung / Korrektur mit Hintergrundwissen
    - Überprüfung/Spell checking mit Wörterbüchern Datenbanken mit Adressen, Produktbezeichnungen, Akronymen/Abkürzungen, etc.
    - Nutzung bekannter Attributabhängigkeiten zur Korrektur von fehlenden / falschen Attributwerten



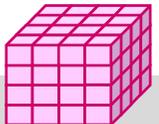
# Behandlung von Multi-Source-Problemen

- Hauptproblem: Entdecken von Duplikaten bzw. korrespondierender Objekte (Objekt Matching)
- Durchführung auf aufbereiteten und gesäuberten Quellen
- Hauptschritte: Identifikation von “ähnlichen” Records (Matching) und Mischen (Merge) zu einem Record mit allen relevanten Attribute ohne Redundanz
- *Exact Matching*: Existenz eines Attributs oder eine Attributkombination zur eindeutigen Identifikation der einzelnen Records
  - Nutzung der Standard-Equijoin-Operationen zur Zusammenführung der zugehörigen Records
  - Sortierung der Records über die Schlüsselattribute und Vergleich der benachbarten Records zur Duplikatidentifikation
- *Fuzzy Object Matching*: keine gemeinsamen Schlüsselattribute (Normalfall)



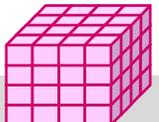
# Fuzzy Object Matching

- Suche und Ranking ähnlicher Records auf Basis von Match-Verfahren bzw. (nutzerdefinierten) Matching-Regeln
  - Berechnung von Ähnlichkeitsmaßen zwischen Objektinstanzen / Sätzen zwischen 0 und 1
  - Robustheit gegenüber leicht variierenden Schreibweisen / Tippfehlern
  - Verwendung von Distanzfunktionen für String-Vergleiche: Edit-, Keyboard-Distanz, n-gram, TF/IDF, ...
  - Kombination der Attribut-Ähnlichkeiten zur Abschätzung der Satz-Ähnlichkeit
  - ggf. Gewichtung der Felder (z.B. hohe Gewichte für Namens-, Adressenfelder beim Matching von Personen-Records)
  - Berücksichtigung von Kontextinformationen (z.B. Gatte bei Personen, Koautoren bei Autoren, etc.)
- Performance-Probleme für große Datenmengen
  - vorhergehendes Ausfiltern sehr unähnlicher Objektpaare zur Einschränkung zu prüfender Match-Kandidaten („Blocking“)



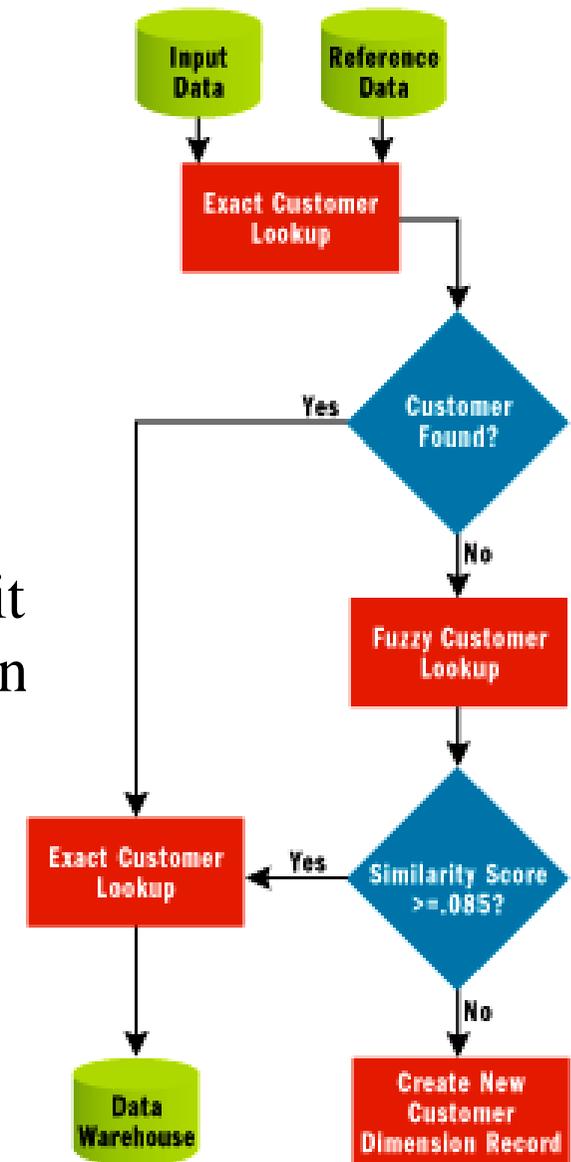
# Tool-Unterstützung

- Datenanalyse-Tools, z.B. Migration Architect, Information Discovery
- ETL-Tools
  - z.B. CopyManager (Information Builders), Extract(ETI), PowerMart (Informatica), DecisionBase (CA/Platinum), DBS-spezifische Tools (Microsoft, IBM, Oracle) etc.
  - Spezialisierung auf Datentransformationen zur Population von Data Warehouses, Data Cleaning oft nur eingeschränkt
  - proprietäre Transformationssprachen mit vordefinierten Funktionen für häufig benötigte Konvertierungen
  - proprietäre APIs zum Aufruf eigener Routinen
- Cleaning-Tools
  - Data Reengineering-Tools (z.B. Vality Integrity): Datenstandardisierung mit Transformationen für Attribute/Sätze, z.B. Split, Delete, Merge
  - spezielle Tools für Namen- und Adressen-Cleaning, z.B. Trillium mit 200.000+ Regeln
  - Duplikat-Eliminierung (auf bereits gesäuberten Datenquellen): DataCleanser (EDD), Merge/Purge Library (Sagent/QM Software), MasterMerge (Pitnew Bowes)
    - Bereitstellung mehrerer Ansätze zum Attributvergleich und zur Kombination der Ähnlichkeitsmaße

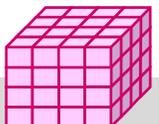


# MS SQL-Server 2005: Data Cleaning Operatoren

- Bestandteil von SQL-Server Integration Services (SSIS; vormals DTS)\*
  - Definition komplexer ETL-Workflows
  - zahlreiche Operatoren
- Fuzzy Lookup
  - “Fuzzy Join” zwischen Eingaberelation und sauberen Sätzen einer Referenztabelle
  - Parameter: Schwellwerte bzgl. String-Ähnlichkeit (Edit Distance) sowie Gewichte zur Kombination von Ähnlichkeiten
- Fuzzy Grouping
  - Gruppierung ähnlicher Sätze (potentielle Duplikate) innerhalb einer Tabelle über String-Matching (Edit Distance)



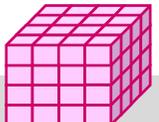
\* <http://msdn.microsoft.com/en-us/library/ms345128.aspx>



# MOMA-Prototyp (U Leipzig)

- MOMA = Mapping based Object Matching
- Framework für Objekt-Matching (Fuzzy Match)
  - Unterstützung komplexer Match-Workflows
  - Kombination mehrerer Matcher / Ergebnisse
  - Wiederverwendung bereits berechneter Mappings
  - Unterstützung 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 („association mappings“) werden durch spezielle Matcher bzw. Workflows ausgenutzt
- Implementierung im Rahmen der iFuice-P2P-Architektur zur Datenintegration

Quelle <sub>A</sub>	Quelle <sub>B</sub>	Sim
a <sub>1</sub>	b <sub>1</sub>	0.9
a <sub>2</sub>	b <sub>2</sub>	0.7
a <sub>3</sub>	b <sub>3</sub>	1



# Objekt-Matching für Webdaten

```
@article{DBLP:journals/vldb/RahmB01,  
  author = {Erhard Rahm and Philip A. Bernstein},  
  title  = {A survey of approaches to automatic schema matching.}  
  journal= {VLDB J.}, year    = {2001}, ...
```

DBLP

[A survey of approaches to automatic schema matching - group of 25 »](#)  
EJ Rahm, PAJ Bernstein - The VLDB Journal The International Journal on Very Large  
... In the next section, we summarize some example applica- tions of **schema match**  
E provides a classification of different ways to perform Match **automatically**. ...  
[Cited by 585](#) [Web Search](#)

Google Scholar

## A survey of approaches to automatic schema matching

Information Fusion

Full text  [Pdf](#) (196 KB)

Source **The VLDB Journal — The International Journal on Very Large Data Bases** [archive](#)

Volume 10 , Issue 4 (December 2001) [table of contents](#)

Pages: 334 - 350

Year of Publication: 2001

ISSN:1066-8888

Authors [Erhard Rahm](#)

Universität Leipzig, Institut für Informatik, 04109 Leipzig, Germany; (e-mail: rahm@informatik.uni-leipzig.de)

[Philip A. Bernstein](#)

Microsoft Research, Redmond, WA 98052-6299, USA; (e-mail: philbe@microsoft.com)

Publisher Springer-Verlag New York, Inc. Secaucus, NJ, USA

Additional Information:

[abstract](#)

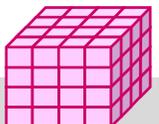
[citing](#)

[index terms](#)

[collaborative colleagues](#)

[peer to peer](#)

ACM



# Duplikate in Webdaten: Beispiel

[A survey of approaches to automatic schema \*\*matching\*\* - group of 25 »](#)

EJ Rahm, PAJ Bernstein - The VLDB Journal The International Journal on Very Large ... , 2001 - Springer

... In the next section, we summarize some example applica- tions of **schema matching**. ...

5 provides a classification of different ways to perform Match **automatically**. ...

[Cited by 585 - Web Search](#)

[CITATION] A survey of approaches to **automatic schema matching**

PA Bernstein, E Rahm - VLDB Journal, 2001

[Cited by 14 - Web Search](#)

[CITATION] A survey of approaches to automatic schema **matching**

E Rahm, PA Bernstein - VLDB Journal, 2001

[Cited by 2 - Web Search](#)

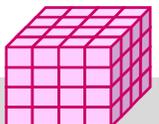
[CITATION] On **matching schemas automatically**

E Rahm, PA Bernstein - VLDB Journal, 2001

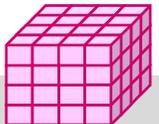
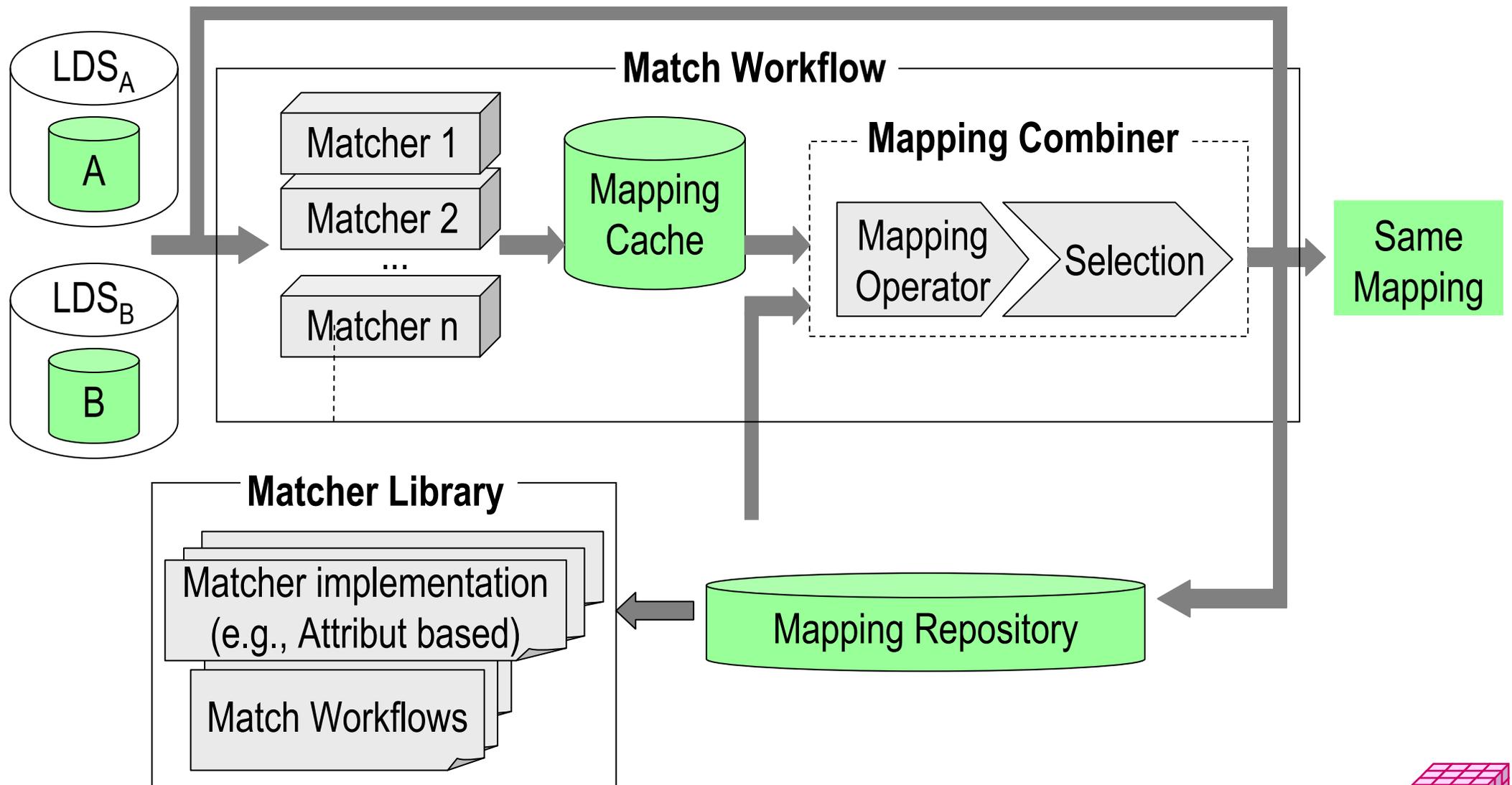
[Cited by 100 - Web Search](#)

Duplikate wegen

- Extraktionsfehlern (Titel)
- Fehlern in Lit.verzeichnissen (Titel, Autorenname und -reihenfolge etc.)

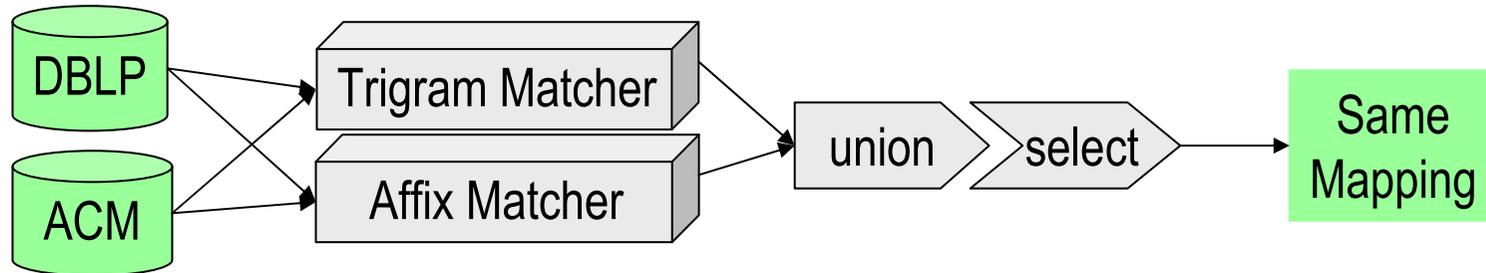


# MOMA-Architektur



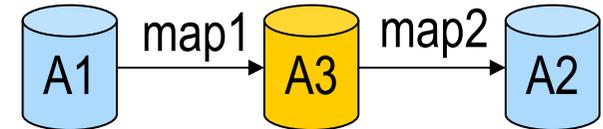
# Beispiel-Workflows

- Kombination unabhängiger Attribut-Matcher-Ausführungen



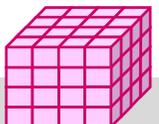
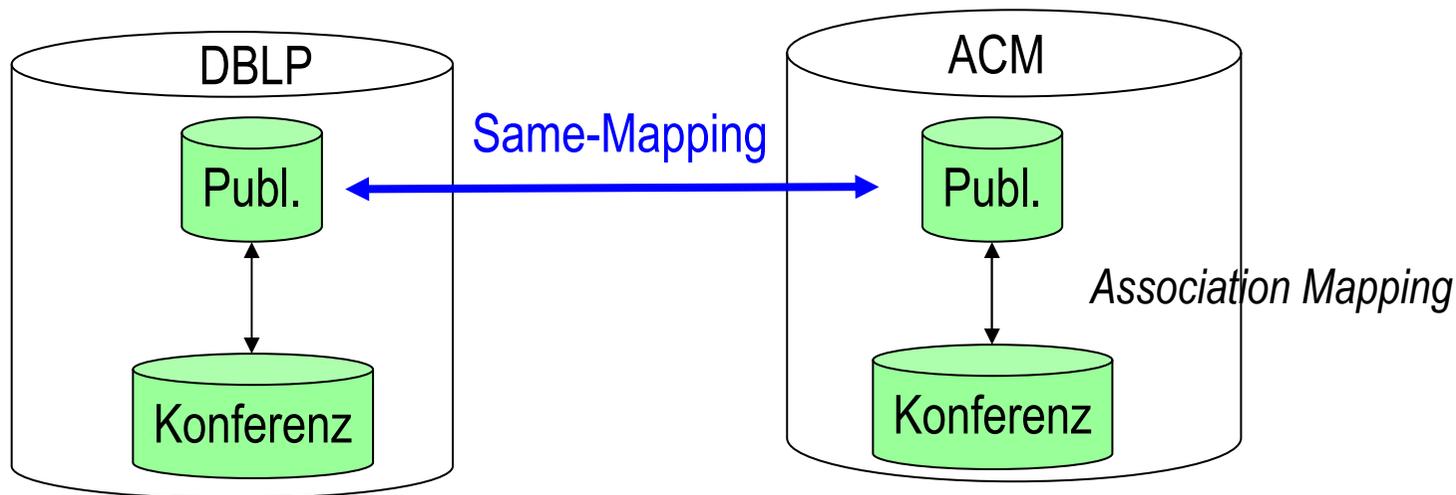
- Komposition von Mappings

- Wiederverwendung exist. Match-Ergebnisse



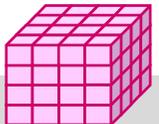
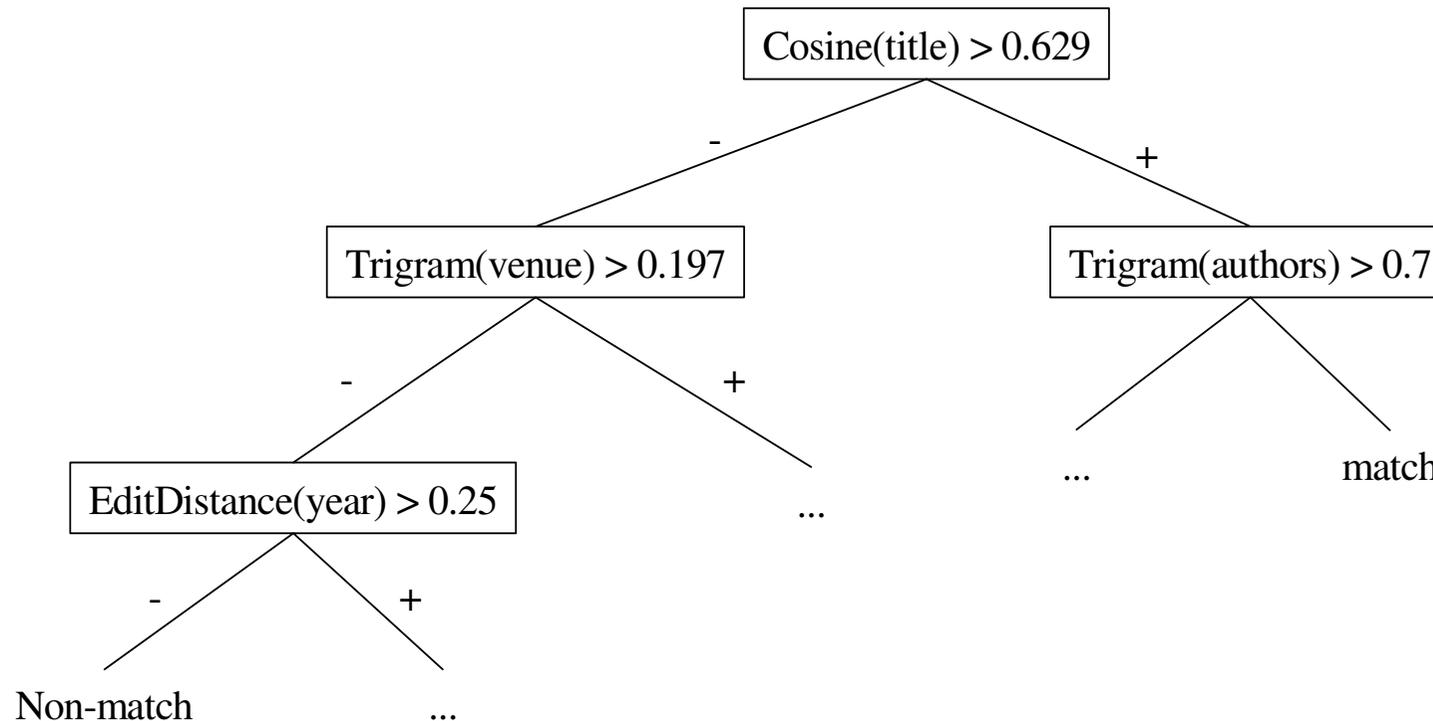
- Nutzung existierender Mappings sowie semantischer Mapping-Typen

- Beispielaufgabe: Bestimme Objekt-Matching für Konferenzen



# Trainingsbasiertes Objekt-Matching

- Finden effektiver Match-Einstellungen ist schwierig
  - Auswahl der Attribute, Matcher, Einstellungen
- Machine Learning verspricht Verbesserung
  - manuell spezifizierte Trainingsdatenmenge
  - Lernen von Match-Kriterien (z.B. mit Entscheidungsbaum)
  - Problem: gute Trainingsdaten mit vertretbarem manuellem Aufwand



# Zusammenfassung

- ETL als komplexer, aufwendiger Integrationsprozess
- Schema- und Datenintegration / Data Cleaning
  - zahlreiche Schema- und Datenkonflikte
  - begrenzte Automatisierbarkeit bezüglich Konflikterkennung und -behandlung
  - möglichst deskriptive Spezifikation aller Datentransformationen zur Behandlung von Schema- und Datenkonflikten
- Fokussierung auf Data Warehouse-spezifisches Zielschema erleichtert Schemaintegration
  - Top-Down-Schemaintegration
  - keine vollständige Integration aller Quell-Schemata erforderlich
- wichtiges Teilproblem: Schema-Matching
  - Nutzung und Kombination mehrerer Lösungsalgorithmen (Matcher)
  - Reuse früherer Match-Ergebnisse
- Unterscheidung quell-lokaler und -übergreifender Datenkonflikte
  - Data Cleaning zunächst auf einzelnen Quellen
  - Duplikat-Identifikation und –Behandlung (Objekt Matching)

