



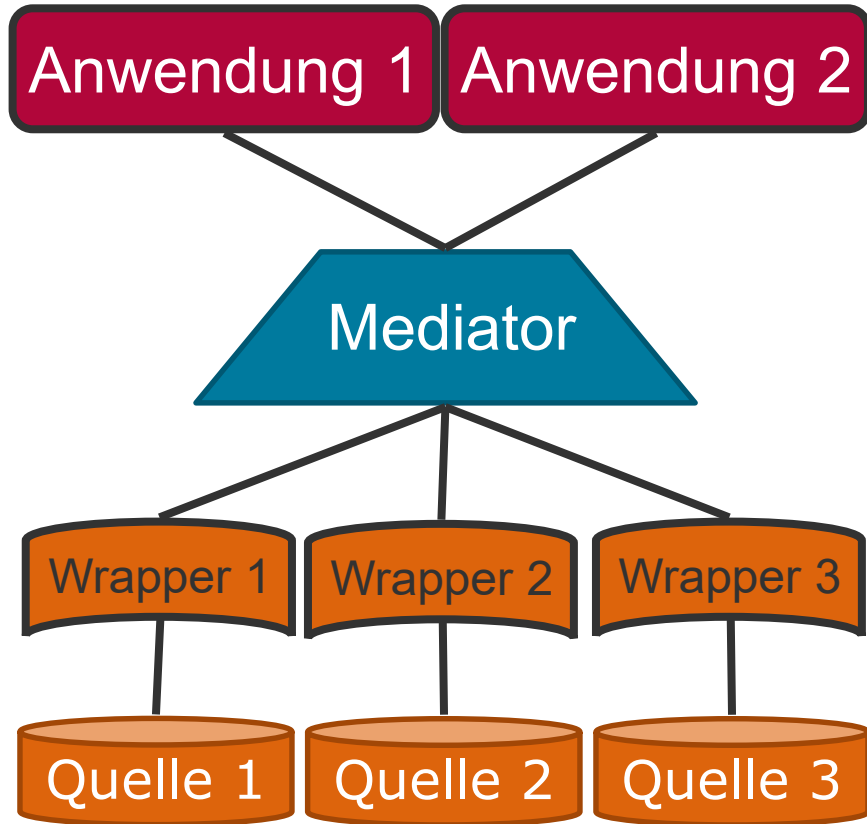
# Überblick

1. Motivation
2. Schema Mapping
3. Schema Matching
4. Mapping Interpretation
5. Mapping Werkzeuge



Felix Naumann  
Information Integration  
Winter 2019/20

## Wdh: Virtuelle Integration



- Datenfluss
- Anfragebearbeitung
- Entwicklung
  - Top-down
- Schema

## Wdh: Bottom-up oder Top-down Entwurf

---

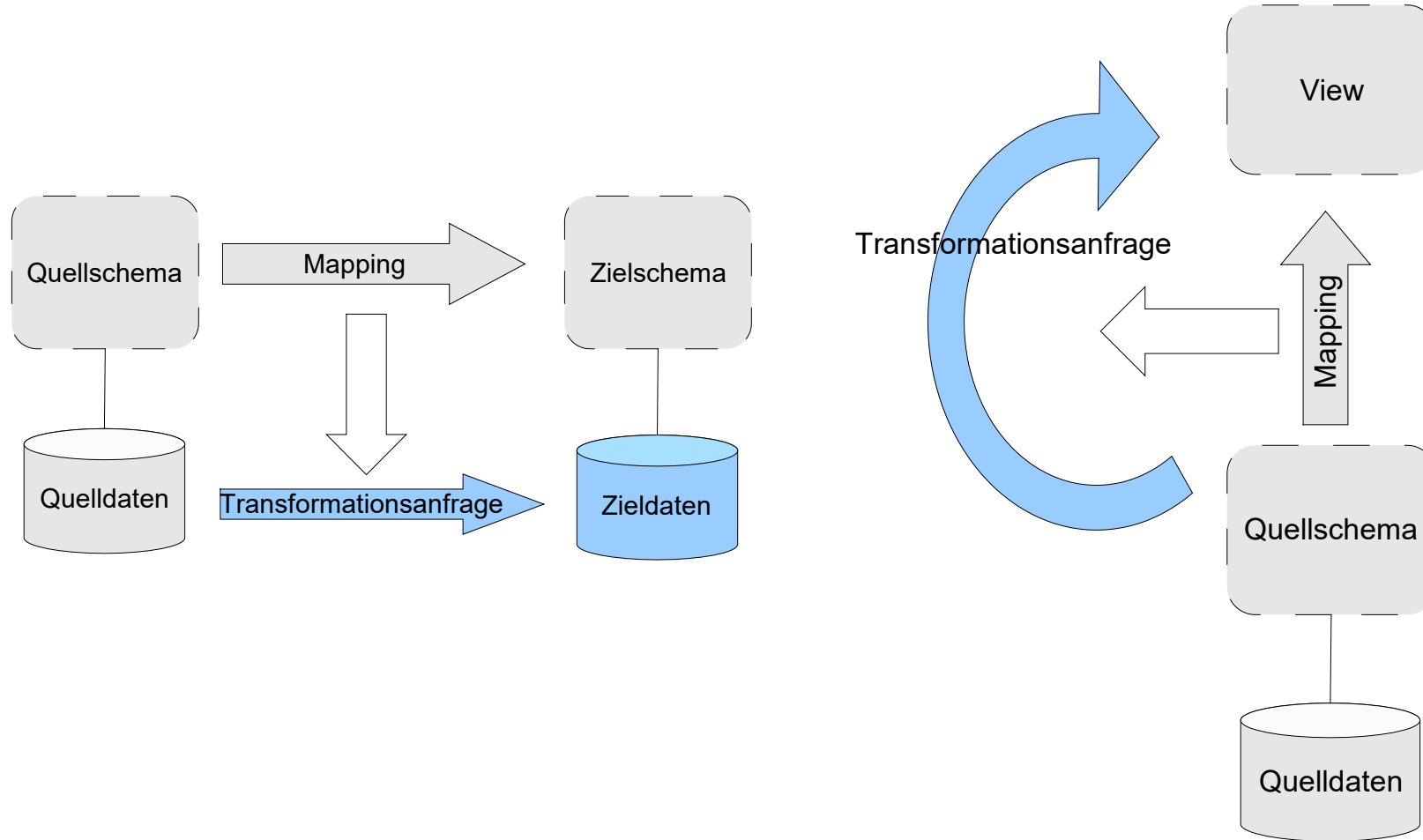
- Beim Entwurf des integrierten Systems
  
- Bottom-up
  - Ausgelöst durch den Bedarf, mehrere (alle) Quellen integriert anzufragen
  - Schemaintegration ist nötig.
  - Änderungen schwierig, da neu integriert werden muss.
  - Typisches Szenario: Data Warehouse
  
- Top-down
  - Ausgelöst durch globalen Informationsbedarf
  - Vorteilhaft bei labilen Quellen
  - Schemaintegration nicht nötig, bzw. leichter

## Schemaintegration vs. Schema Mapping

---

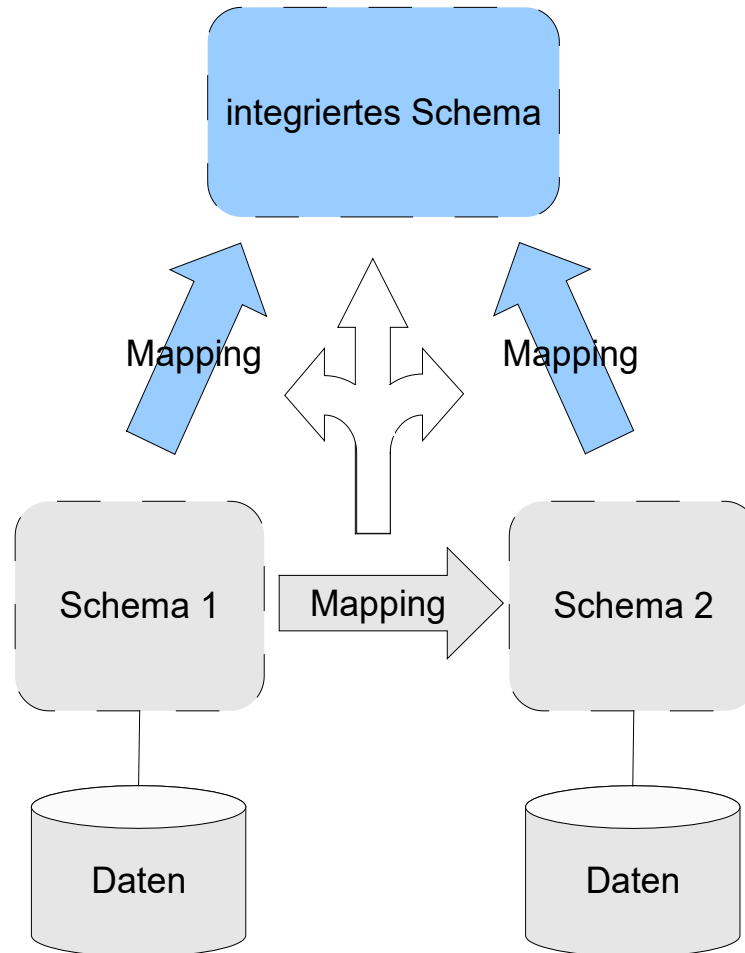
- Beide Problemlösungen müssen strukturelle und semantische Heterogenität überwinden.
- Aber:
  - Schemaintegration liefert Schema Mapping „frei Haus“.
  - Zielschema hat keine eigene Semantik.
  - Schemaintegration ist unflexibel.
- Deshalb nun: Schema Mapping

# Verwendung von Schema Mappings: Datentransformation



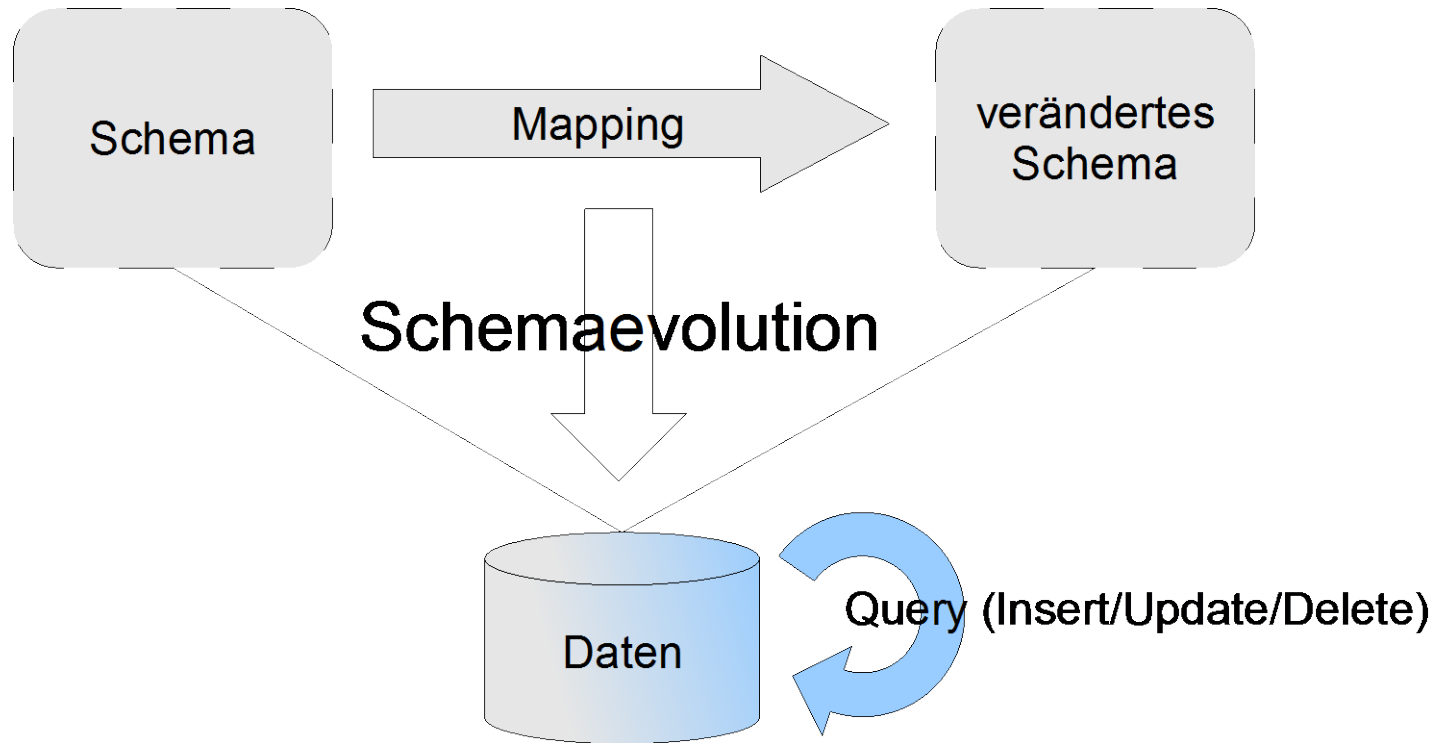
Felix Naumann  
Information Integration  
Winter 2019/20

# Verwendung von Schema Mappings: Schemaintegration



Felix Naumann  
Information Integration  
Winter 2019/20

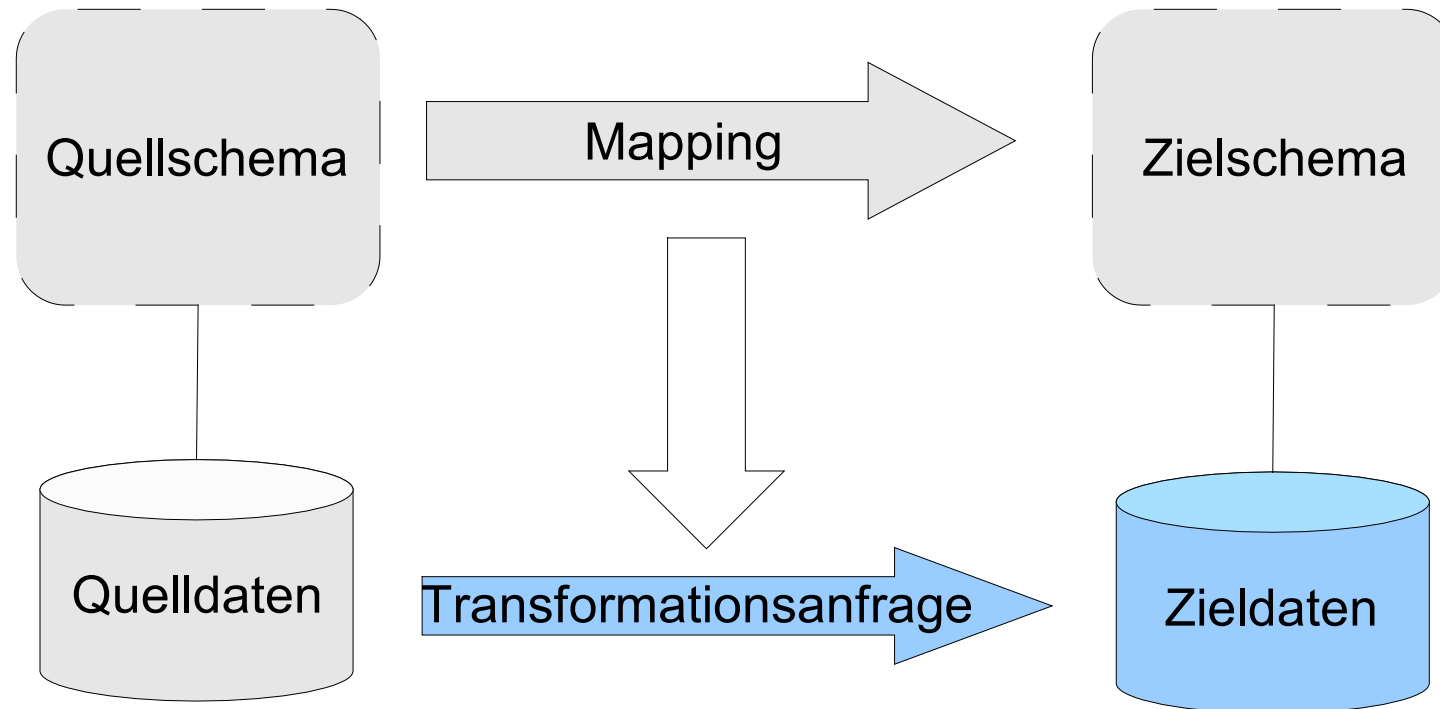
# Verwendung von Schema Mappings: Schemaevolution





## Verwendung von Schema Mappings

- Im weiteren: Datentransformation
  - Materialisierte Integration
  - Virtuelle Integration



## Überblick

1. Motivation
- 2. Schema Mapping**
3. Schema Matching
4. Mapping Interpretation
5. Mapping Werkzeuge



Felix Naumann  
Information Integration  
Winter 2019/20

## Schema Mapping – Begriffe

---

- (Inter-Schema) **Korrespondenz**
  - Eine Zuordnung eines oder mehrerer Elemente eines (Quell-) Schemas zu einem oder mehreren Elementen eines anderen (Ziel-) Schemas
  - Auch: Value-correspondence
- (High-level) **Mapping**
  - Eine Menge von Korrespondenzen zwischen zwei Schemata.
- (Low-Level) **Logisches Mapping**
  - Logische Übersetzung eines oder mehrerer Mappings, die
    - den Integritätsbedingungen beider Schemata gehorcht und
    - die Intention des Nutzers widerspiegelt.
- **Interpretation**
  - Übersetzung eines Mappings in ein oder mehrere logische Mappings
  - Übersetzung eines logischen Mappings in eine Transformationsanfrage
- **Transformationsanfrage**
  - Anfrage in einer Anfragesprache (z.B. SQL), die Daten des Quellschemas in die Struktur des Zielschemas überführt

## Wdh: Schematische Heterogenität

---

### ■ Struktur

#### □ Modellierung

- Relation vs. Attribut
- Attribut vs. Wert
- Relation vs. Wert

#### □ Benennung

- Relationen
- Attribute

#### □ Normalisiert vs. Denormalisiert

#### □ Geschachtelt vs. Fremdschlüssel

High-order Mappings (später und SchemaSQL)

hier

## Motivation

---

- Datentransformation zwischen heterogenen Schemata
  - Altes aber immer wiederkehrendes Problem
  - Üblicherweise schreiben Experten komplexe Anfragen oder Programme
    - Zeitintensiv
    - Experte für die Domäne, für Schemata und für Anfrage
    - XML macht alles noch schwieriger: XML Schema, XQuery
  
- Idee: Automatisierung
  - Gegeben: Zwei Schemata
  - Gesucht: High-level Mapping zwischen den beiden Schemata
  
  - Gegeben: Zwei Schemata und Mapping
  - Gesucht: Anfrage zur Datentransformation

## Motivation – Probleme

---

- Generierung der „richtigen“ Anfrage unter Berücksichtigung der Schemata und des Mappings
- Garantie, dass die transformierten Daten dem Zielschema entsprechen
- Effiziente Datentransformation
  - Für Materialisierung (Ausführung, inkrementell)
  - Für virtuelle Integration (query-unfolding)

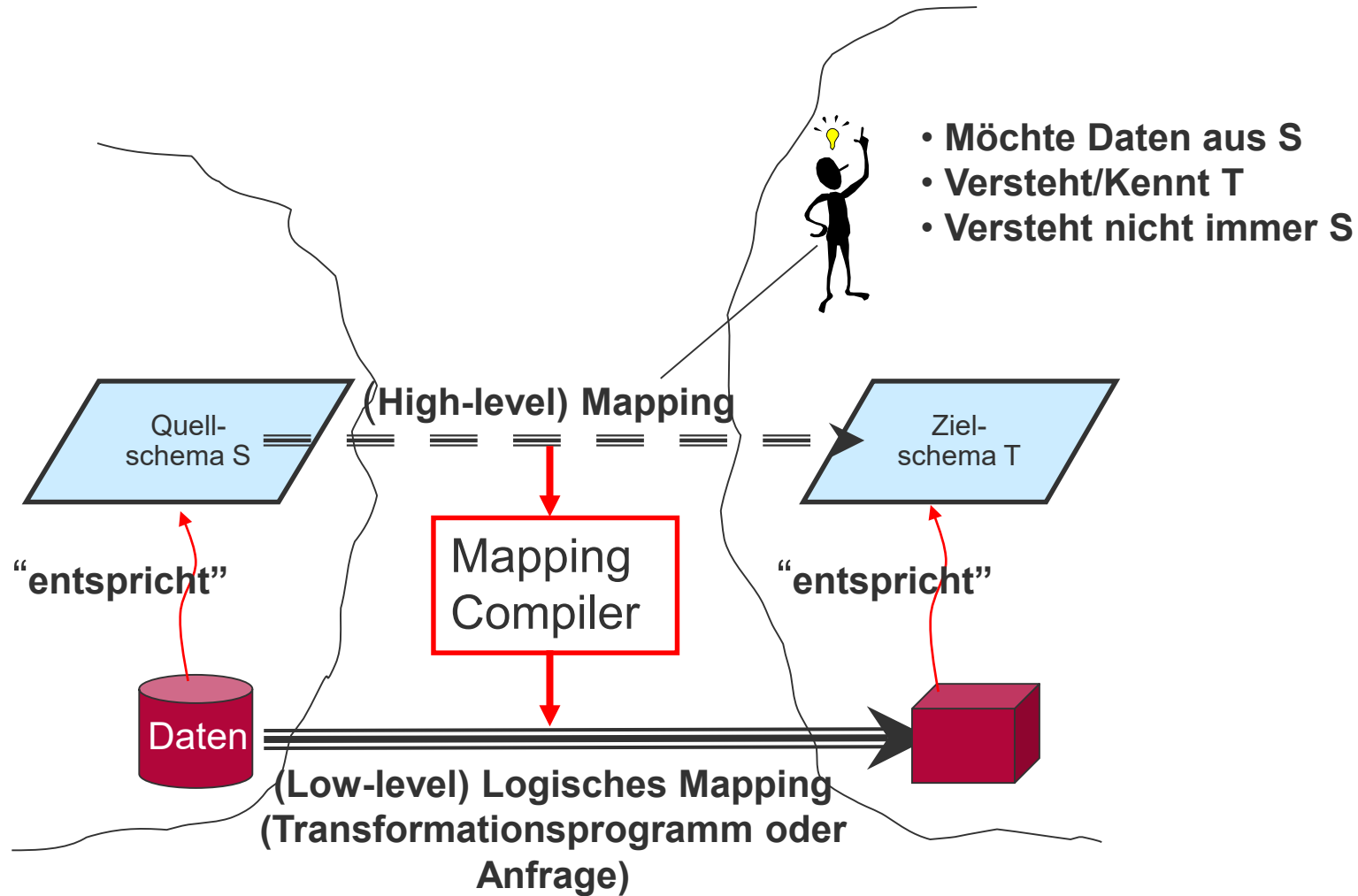
Hier: Nur Effektivität, nicht Effizienz

## Motivation – Weitere Probleme

---

- Geschachtelte Strukturen unterstützen
  - Geschachteltes, relationales Modell
  - XML
  - Geschachtelte Integritätsbedingungen
- Korrespondenzen
  - Nutzerfreundlich
  - Automatische Entdeckung (Schema Matching)
- Intention des Nutzers erkennen und repräsentieren
- Semantik der Daten erhalten
  - Assoziationen entdecken & erhalten
  - Schemata und deren Integritätsbedingungen nutzen
- Neue Datenwerte erzeugen
- Korrekte Gruppierungen erzeugen
- ...

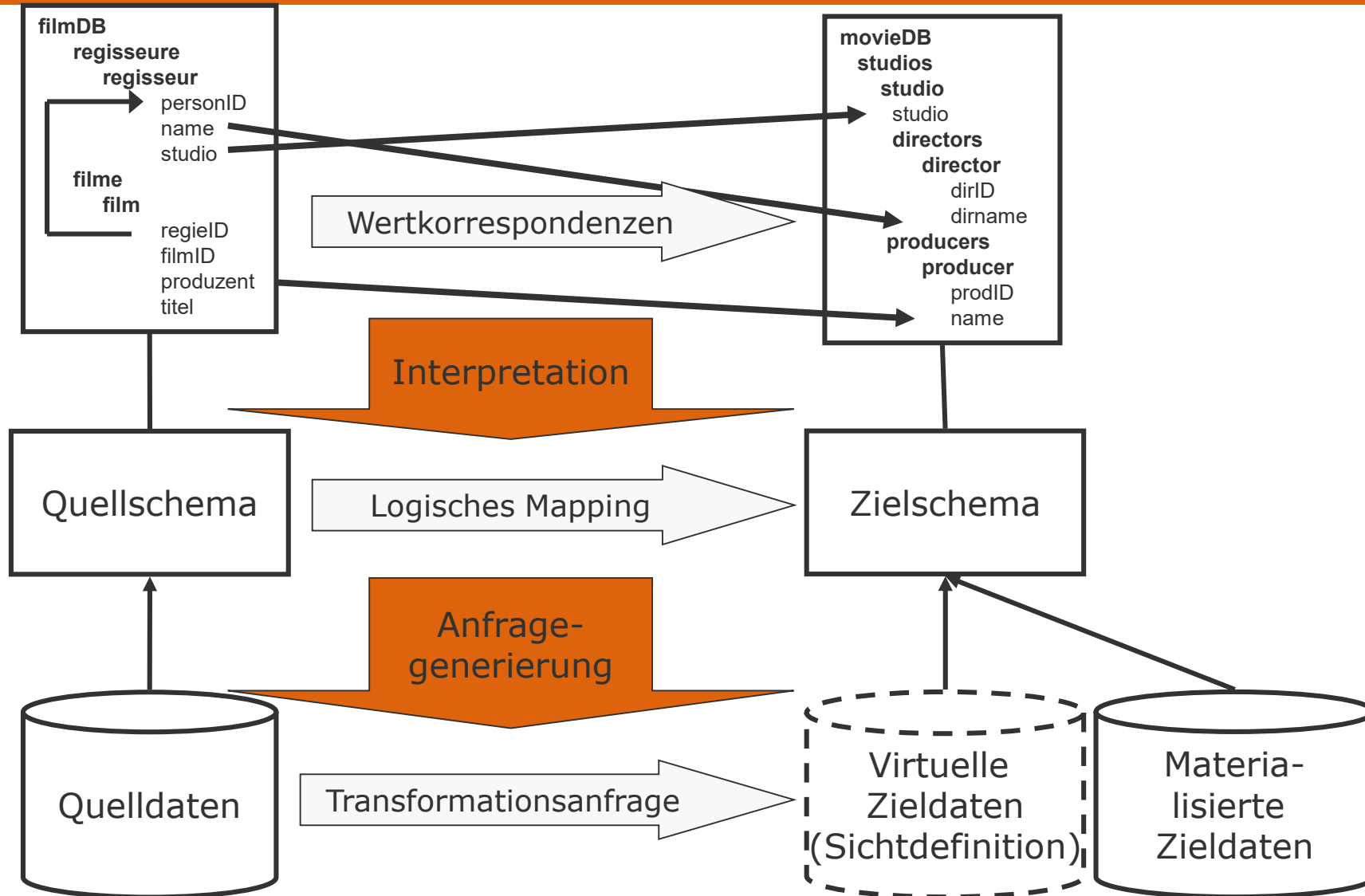
# Schema Mapping im Kontext



Felix Naumann  
Information Integration  
Winter 2019/20

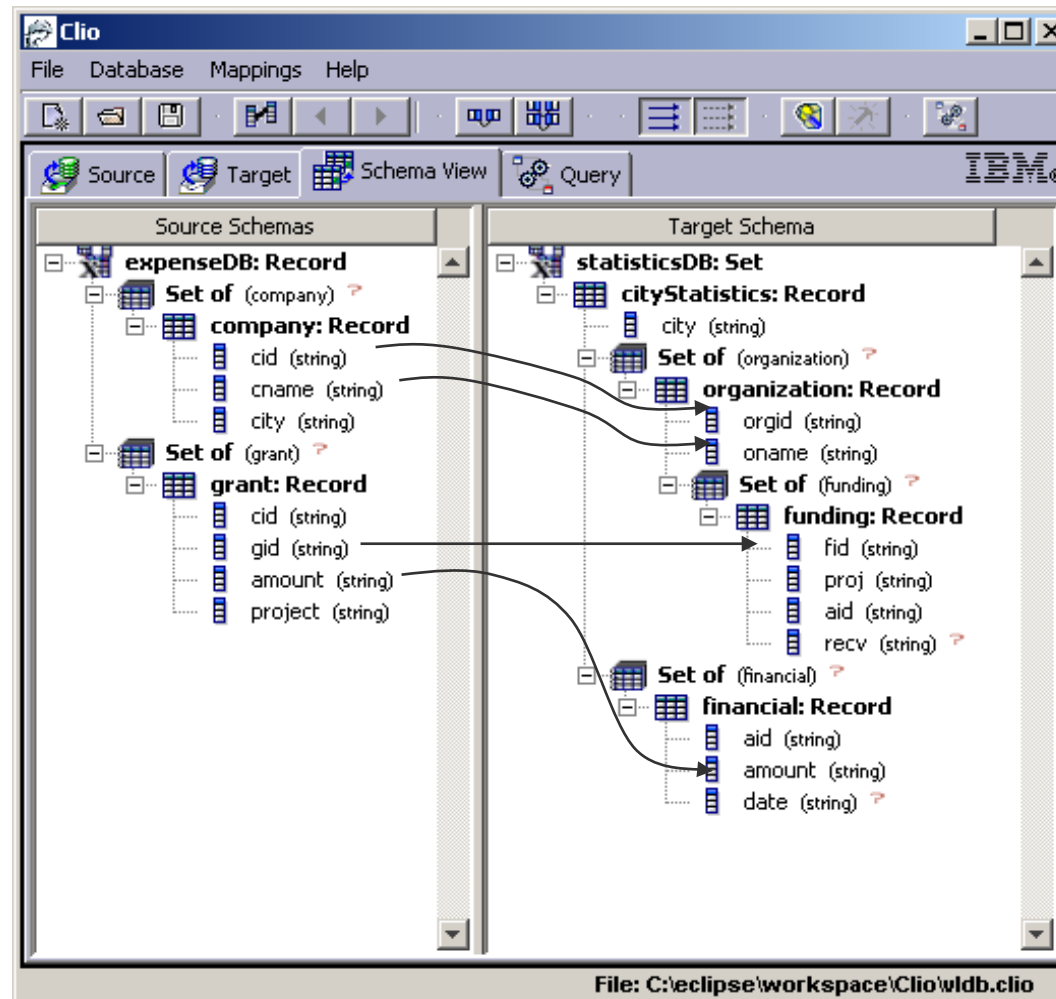


# Schema Mapping im Kontext



# Schema Mapping im Kontext

- Schema Matching & Korrespondenzen
- Schema Mapping
- Mapping Interpretation
- Datentransformation



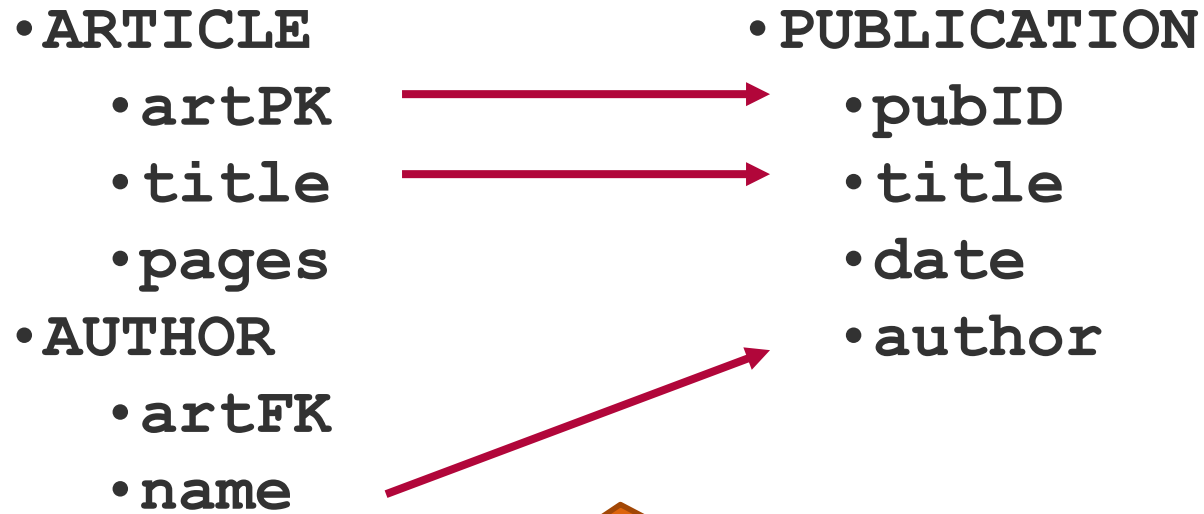
Felix Naumann  
Information Integration  
Winter 2019/20

## Wdh: Schematische Heterogenität – Beispiel

---

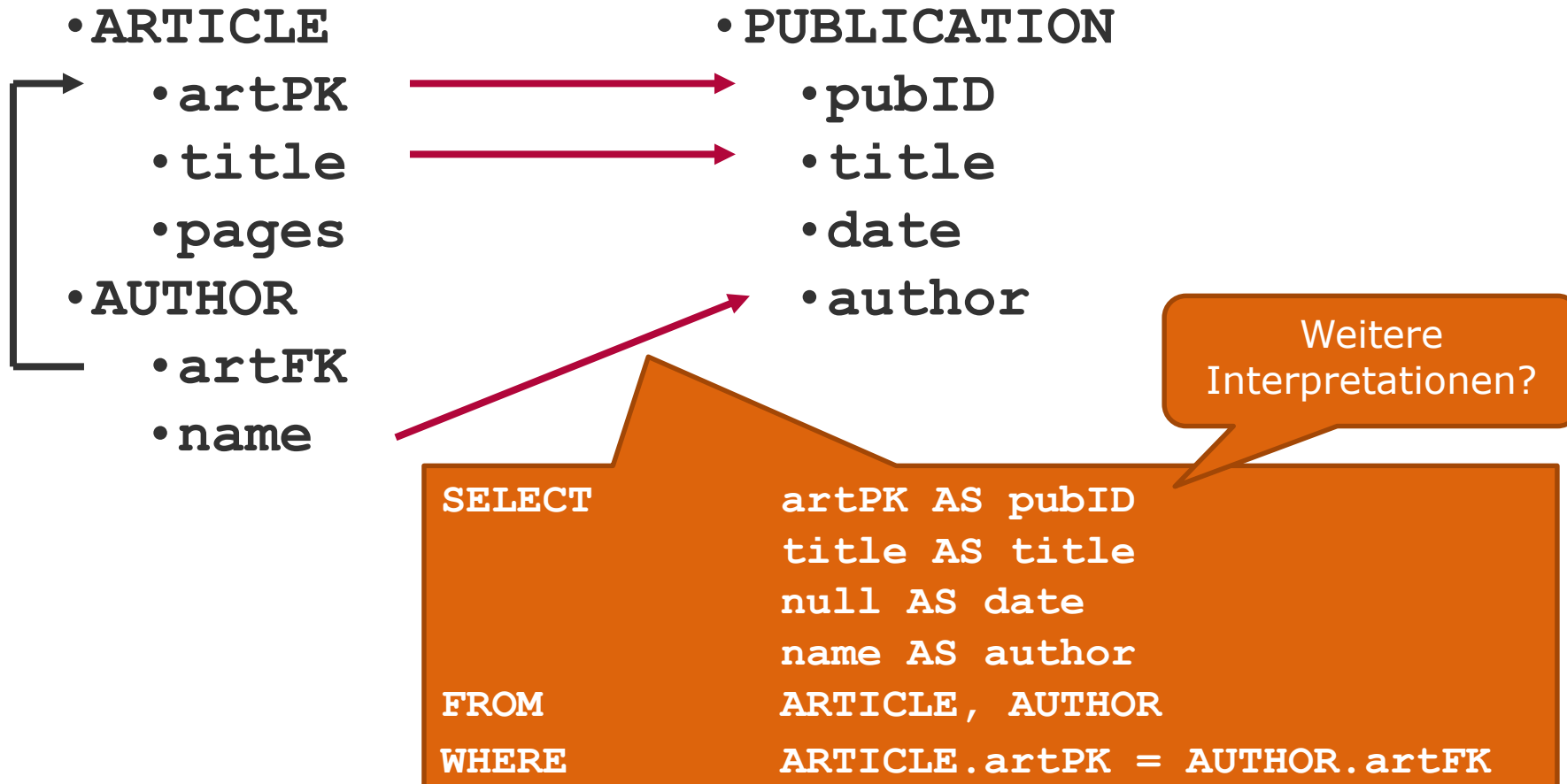
- Normalisiert vs. Denormalisiert
  - 1:n Assoziationen zwischen Werten wird unterschiedlich dargestellt
    - Durch Vorkommen im gleichen Tupel
    - Durch Schlüssel-Fremdschlüssel Beziehung
  
- Lösung: Schema Mapping

# Schema Mapping Beispiel

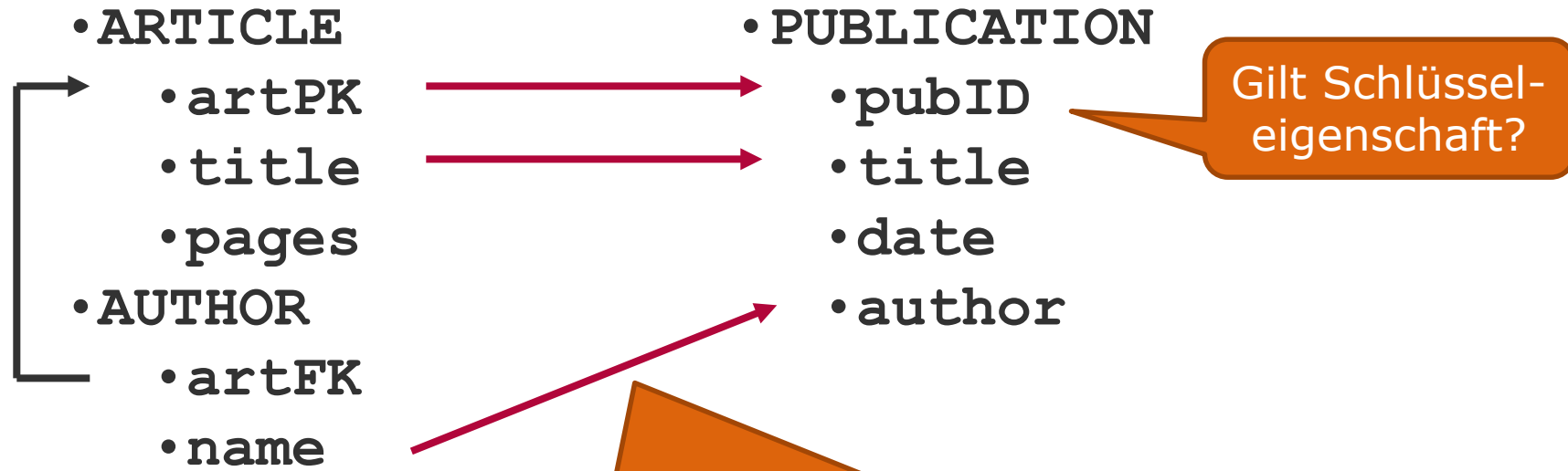


```
SELECT artPK AS pubID UNION SELECT null AS pubID
      title AS title       null AS title
      null AS date         null AS date
      null AS author       name AS author
FROM ARTICLE              FROM AUTHOR
```

# Schematische Heterogenität – Lösungen



# Schematische Heterogenität – Lösungen



```
SELECT      artPK AS pubID
            title AS title
            null AS date
            name AS author
FROM        ARTICLE LEFT OUTER JOIN AUTHOR
ON          ARTICLE.artPK = AUTHOR.artFK
```

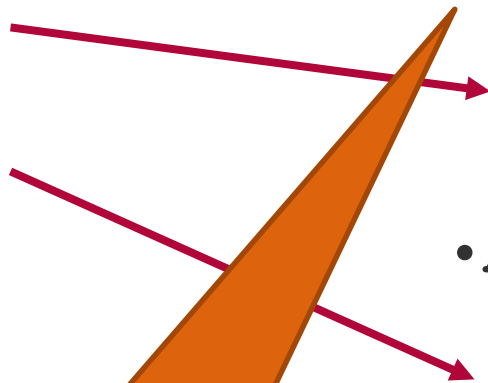
# Wdh: Schematische Heterogenität – Lösungen

• PUBLICATION

- title
- date
- author

• ARTICLE

- artPK
- title
- pages
- AUTHOR
- artFK
- name



**DISTINCT**

```
SELECT SK(title) AS artPK
       title AS title
       null AS pages
FROM PUBLICATION
```

```
SELECT SK(title) AS artFK
       author AS name
FROM PUBLICATION
```

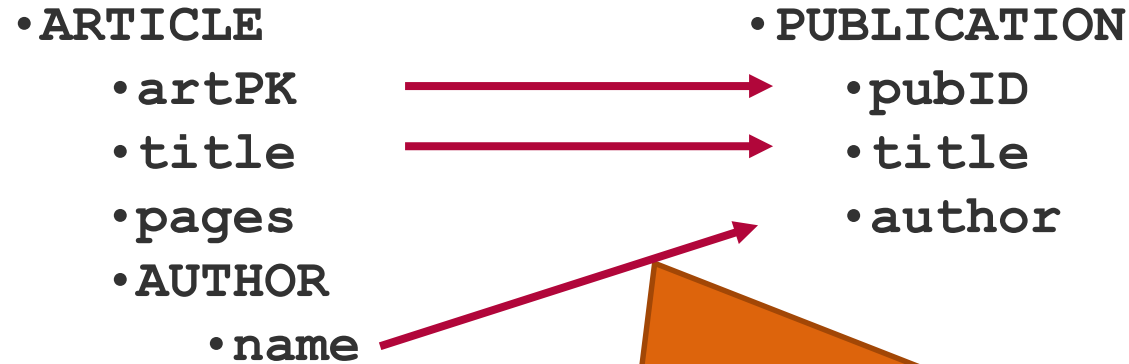
## Wdh: Schematische Heterogenität – Beispiel

---

- Geschachtelt vs. Flach
  - 1:n Assoziationen werden unterschiedlich dargestellt
    - Als geschachtelte Elemente
    - Als Schlüssel-Fremdschlüssel Beziehung
- Lösung: Schema Mapping

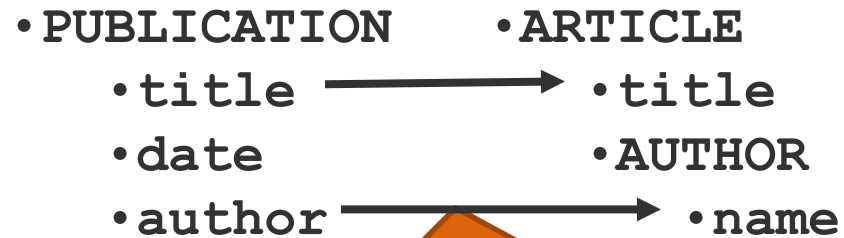


# Wdh: Schematische Heterogenität – Lösungen



```
LET $doc0 := document("articles.xml") RETURN
<root> { distinct-values (
  FOR $x0 IN $doc0/authorDB/ARTICLE, $x1 IN $x0/AUTHOR
  RETURN
    <publication>
    <pubID> { $x0/artPK/text() } </pubID>
    <title> { $x0/title/text() } </title>
    <author> { $x1/name/text() } </author>
    </publication> )
} </root>
```

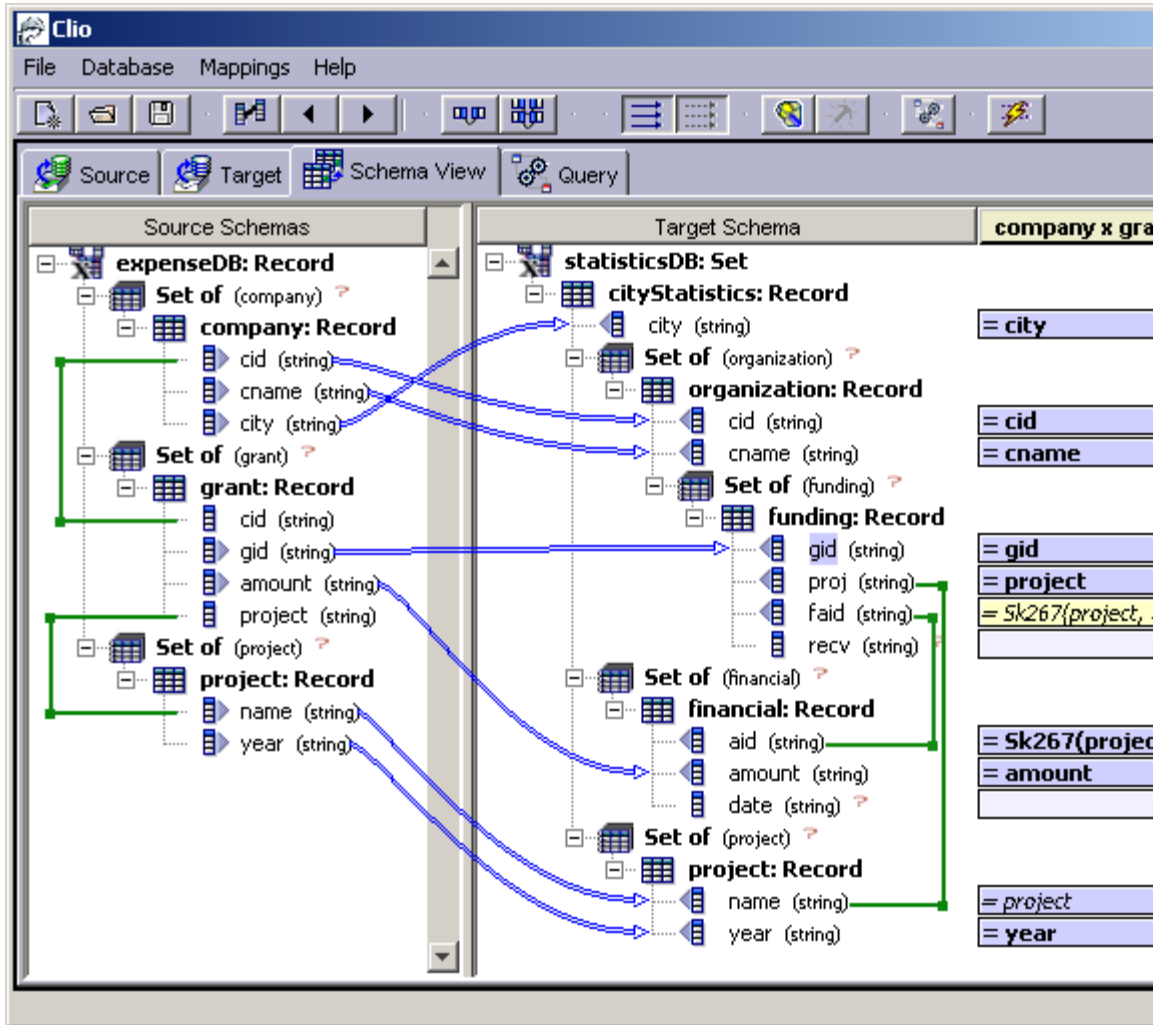
# Wdh: Schematische Heterogenität – Lösungen



```

LET $doc0 := document("publication.xml")
RETURN
<articles> { distinct-values (
  FOR $x0 IN $doc0/dblp/publication RETURN
    <ARTICLE>
      <title> { $x0/title/text() } </title>
      { distinct-values (
        FOR $x0L1 IN $doc0/dblp/publication
        WHERE $x0/title/text() = $x0L1/title/text()
        RETURN
          <AUTHOR>
            <name> { $x0L1/author/text() } </name>
          </AUTHOR> ) }
    </ARTICLE> ) } </articles>

```



```

XQuery
RETURN
  $x0L1/project/text() = $x1L1/name/text() AND
  $x2L1/cid/text() = $x0L1/cid/text() AND
  $x2/city/text() = $x2L1/city/text()
RETURN
  <organization>
    <cid> $x0L1/cid/text() </cid>,
    <cname> $x2L1/cname/text() </cname>,
    distinct (
      FOR
        $x0L2 IN $doc/expenseDB/grant ,
        $x1L2 IN $doc/expenseDB/project ,
        $x2L2 IN $doc/expenseDB/company
      WHERE
        $x0L2/project/text() = $x1L2/name/text() AND
        $x2L2/cid/text() = $x0L2/cid/text() AND
        $x2L1/cname/text() = $x2L2/cname/text() AND
        $x2L1/city/text() = $x2L2/city/text() AND
        $x0L1/cid/text() = $x0L2/cid/text()
      RETURN
        <funding>
          <gid> $x0L2/gid/text() </gid>,
          <proj> $x0L2/project/text() </proj>,
          <faid> "Sk267(", $x0L2/project/text(), ", ",
            </funding> )
        </organization> ),
    distinct (
      FOR
        $x0L1 IN $doc/expenseDB/grant ,
        $x1L1 IN $doc/expenseDB/project ,
        $x2L1 IN $doc/expenseDB/company
      WHERE
    
```

Jetzt zunächst: Schema Matching

Felix Naumann  
Information Integration  
Winter 2019/20

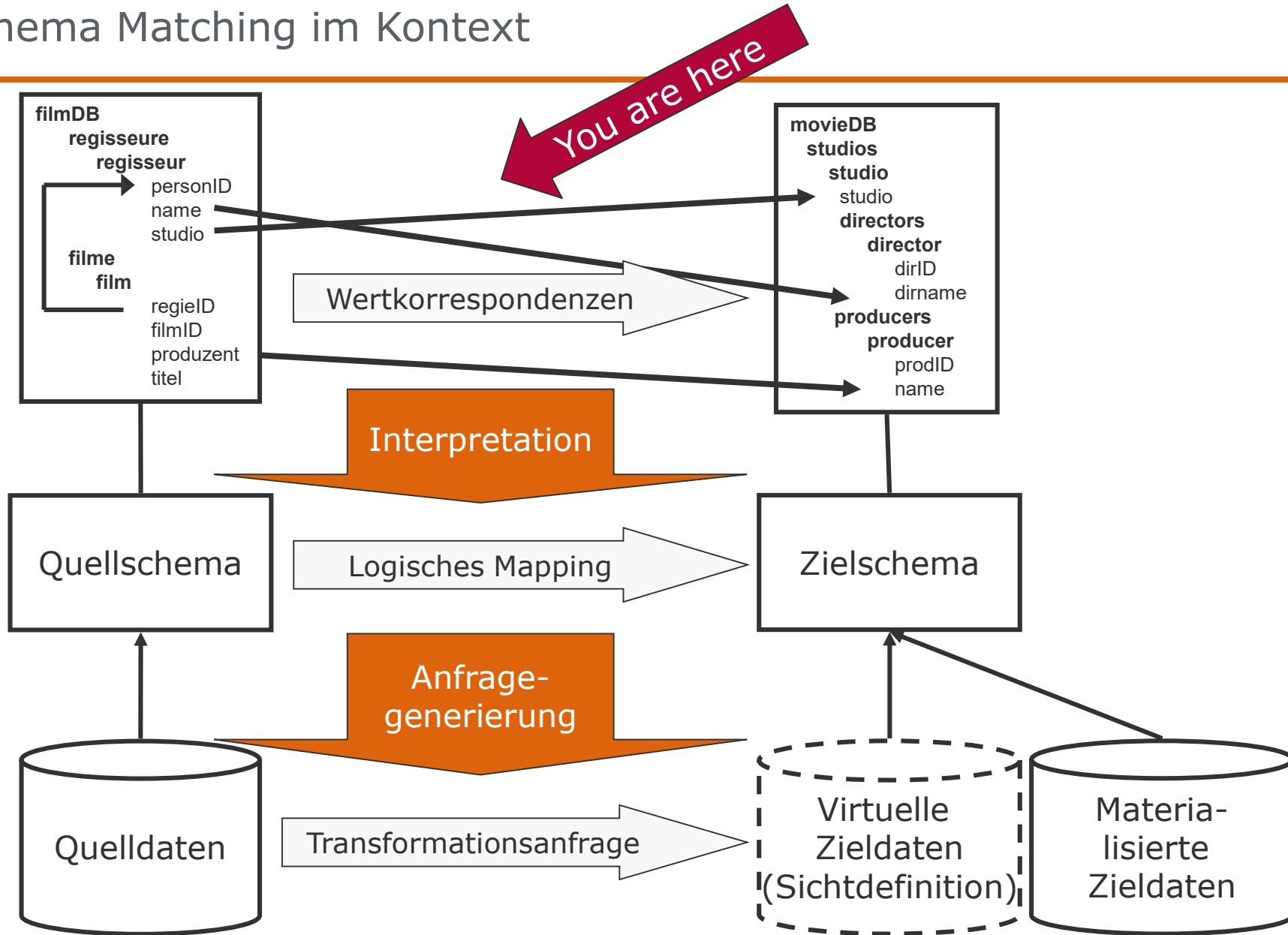
# Überblick

1. Motivation
2. Schema Mapping
3. Schema Matching
  - **Klassifikation von Schema Matching Methoden**
  - Erweiterungen
  - Globales Matching
4. Mapping Interpretation
5. Mapping Werkzeuge



Felix Naumann  
Information Integration  
Winter 2019/20

# Schema Matching im Kontext



# Schema Matching – Motivation

- Große Schemata

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

- Unübersichtliche Schemata

- Tiefe Schachtelungen
- Fremdschlüssel
- Bildschirm nicht breit genug
- XML Schema

- Fremde Schemata

- Unbekannte Synonyme

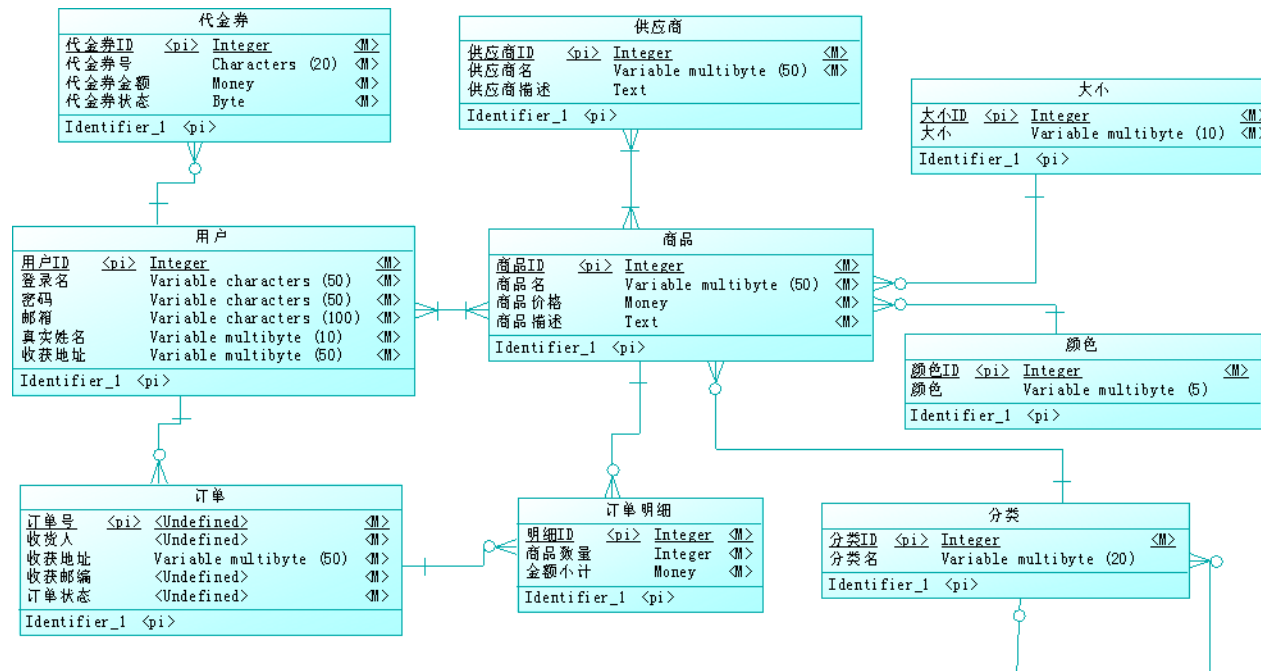
- Irreführende Schemata

- Unbekannte Homonyme

- Fremdsprachliche Schemata

- Kryptische Schemata

- |Attributnamen| ≤ 8 Zeichen
- |Tabellennamen| ≤ 8 Zeichen



Felix Naumann  
Information Integration  
Winter 2019/20

File Schemas Mappings Help

Source Target Schema View Query

Source schemas

- author: Record (personType)
  - name (string)
- PubMed: Record
  - value (unsignedInt) ?
  - status (NMTOKEN) ?
  - medlineID (unsignedInt) ?
- Set [0,\*] ?
  - ev: Record (evidenceRefType)
    - type (string) ?
    - href (anyURI)
    - role (anyURI) ?
    - title (string) ?
    - label (string) ?
    - value() (string) ?
- citedInBook: Record (bookType)
  - linktype (string) ?
  - role (anyURI) ?
  - title (string) ?
  - volume (unsignedInt) ?
  - year (gYear)
  - first (unsignedInt)
  - last (unsignedInt)
  - ISSN (string) ?
  - publisher (string)
  - city (string)
  - country (string) ?
- Record
  - title (string)
  - bookTitle (string)
  - editors: Record (editorsType)
    - Set [1,\*]
      - editor: Record (personType)
        - name (string)
    - authors: Record (authorsType)
      - Set [1,\*]
        - author: Record (personType)
          - name (string)
    - Set [0,\*] ?
      - ev: Record (evidenceRefType)
        - type (string) ?
        - href (anyURI)
        - role (anyURI) ?
        - title (string) ?
        - label (string) ?
        - value() (string) ?
  - Set [0,\*] ?
    - ev: Record (evidenceRefType)
      - type (string) ?
      - href (anyURI)
      - role (anyURI) ?
      - title (string) ?
      - label (string) ?
      - value() (string) ?
  - observations: Record (observationType)
    - linktype (string) ?

Target Schema

Man beachte die Scrollbar!

Man beachte die Schachtelungstiefe!

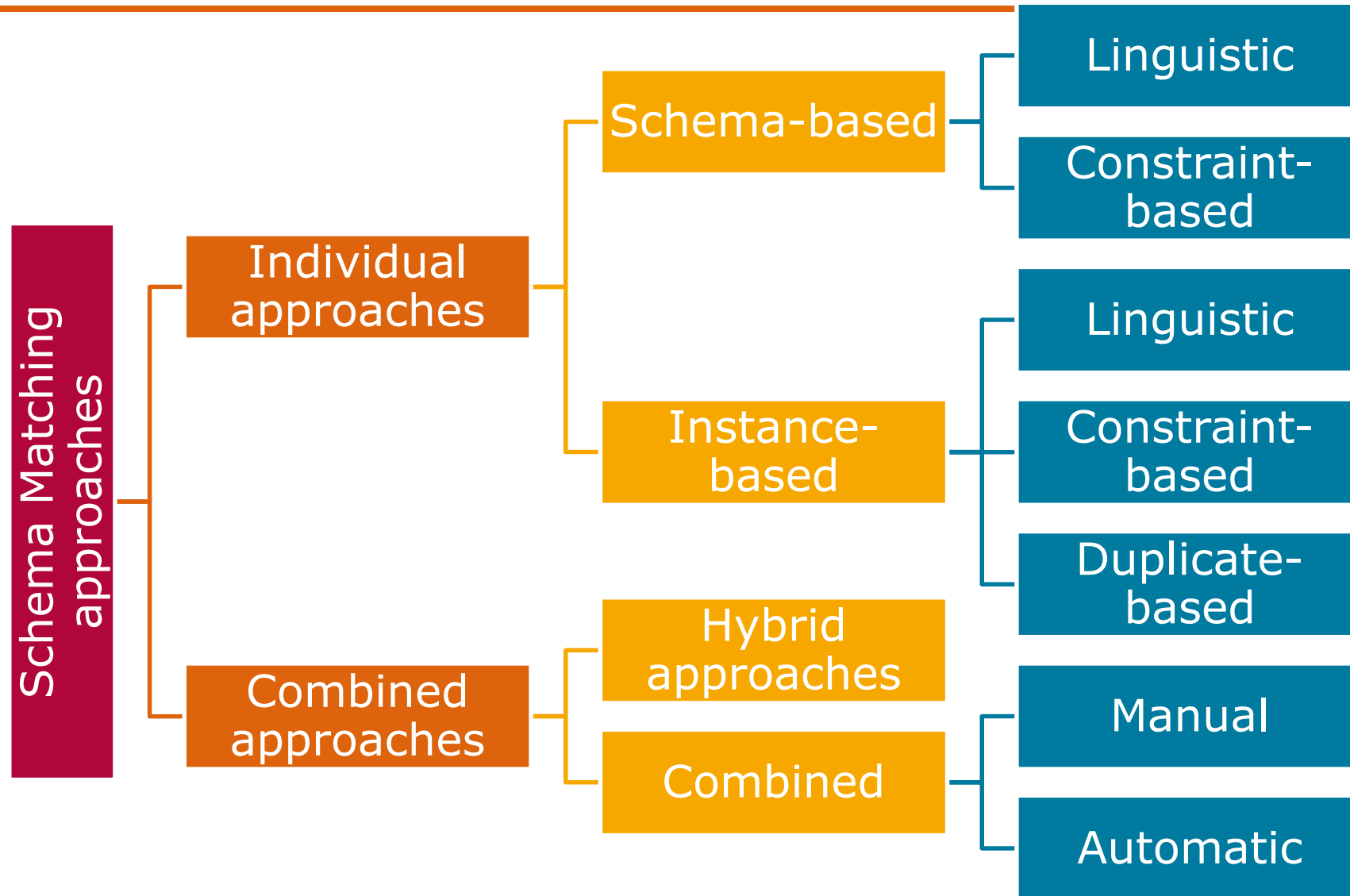
## Schema Matching – Motivation

---

- Die Folgen
  - Falsche Korrespondenzen (false positives)
  - Fehlende Korrespondenzen (false negatives)
  - Frustration
    - User verlieren sich im Schema
    - User verstehen Semantik der Schemata nicht



# Schema Matching Classification [RB01]



## Schema Matching Klassifikation

---

- Schema Matching basierend auf
  - Namen der Schemaelemente (*label-based*)
  - Darunterliegende Daten (*instance-based*)
  - Struktur des Schemas (*structure-based*)
  - Andere Matcher (*composite*)

## Schema Matching – Label-based

---

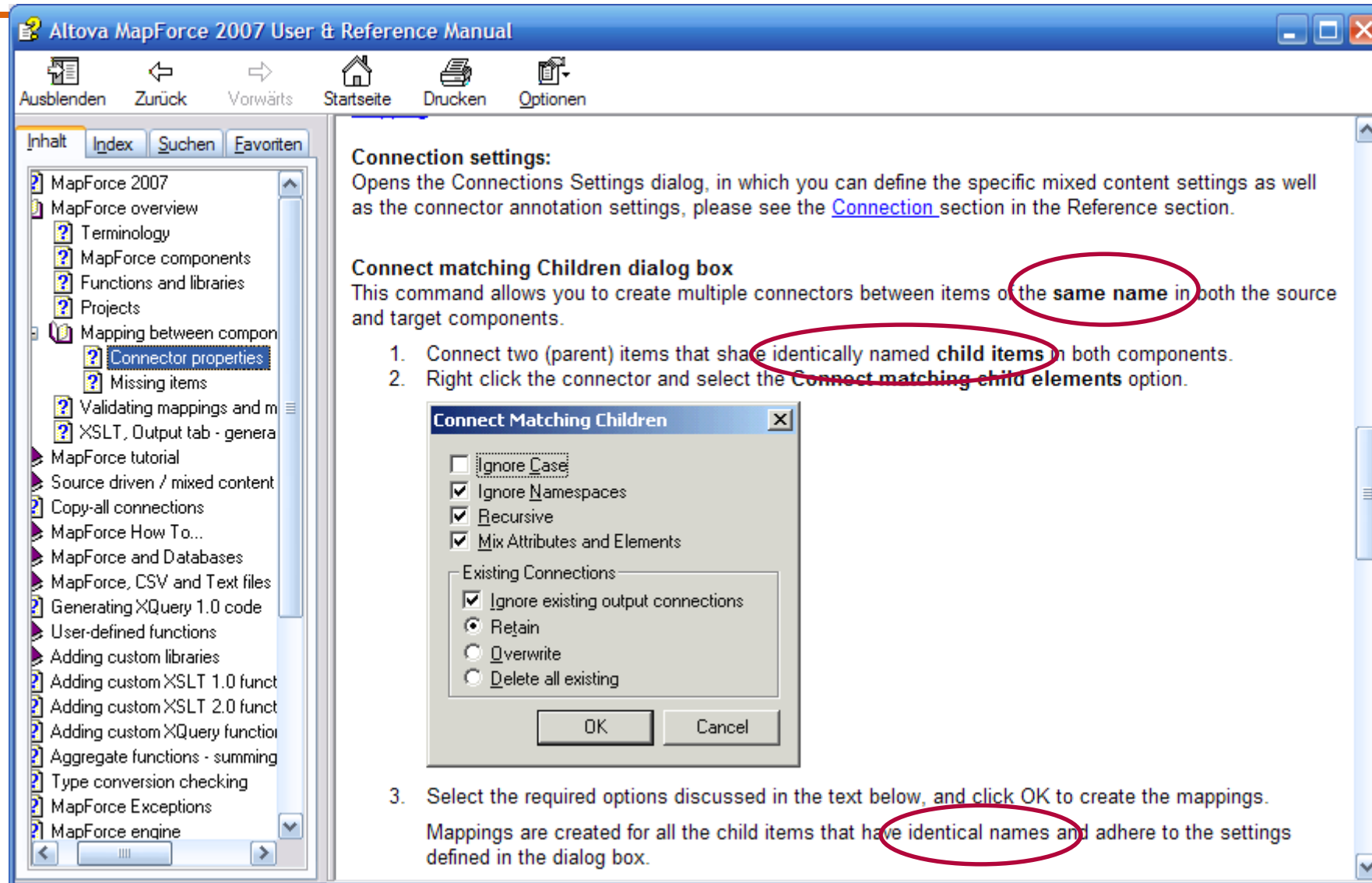
- Gegeben zwei Schemata mit Attributmengen A und B
- Kernidee:
  - Bilde Kreuzprodukt aller Attribute aus A und B.
  - Für jedes Paar vergleiche Ähnlichkeit bezgl. Attributnamen (Label).
    - Z.B. Edit-distance
  - Ähnlichste Paare sind Matches
- Probleme:
  - Effizienz
  - Auswahl der besten Matches (globales Matching – später)
    - Iterativ?
    - Stable Marriage?
  - Synonyme und Homonyme werden nicht erkannt

## Schema Matching – Label-based

---

- Stand der Technik in kommerziellen Produkten
  - Label-based
  - Namensgleichheit
  - Kein globales Matching
  - Keine Ähnlichkeitsmaße
  - Kein Instanz-basiertes Matching

# Altova MapForce 2007



**Altova MapForce 2007 User & Reference Manual**

Ausblenden Zurück Vorwärts Startseite Drucken Optionen

Inhalt Index Suchen Favoriten

- MapForce 2007
- MapForce overview
  - Terminology
  - MapForce components
  - Functions and libraries
  - Projects
- Mapping between compon
  - Connector properties**
  - Missing items
- Validating mappings and m
- XSLT, Output tab - genera
- MapForce tutorial
- Source driven / mixed content
- Copy-all connections
- MapForce How To...
- MapForce and Databases
- MapForce, CSV and Text files
- Generating XQuery 1.0 code
- User-defined functions
- Adding custom libraries
- Adding custom XSLT 1.0 funct
- Adding custom XSLT 2.0 funct
- Adding custom XQuery functio
- Aggregate functions - summing
- Type conversion checking
- MapForce Exceptions
- MapForce engine

**Connection settings:**  
 Opens the Connections Settings dialog, in which you can define the specific mixed content settings as well as the connector annotation settings, please see the [Connection](#) section in the Reference section.

**Connect matching Children dialog box**  
 This command allows you to create multiple connectors between items of the **same name** in both the source and target components.

1. Connect two (parent) items that share **identically named child items** in both components.
2. Right click the connector and select the **Connect matching child elements** option.

**Connect Matching Children**

Ignore Case

Ignore Namespaces

Recursive

Mix Attributes and Elements

Existing Connections

Ignore existing output connections

Retain

Overwrite

Delete all existing

OK Cancel

3. Select the required options discussed in the text below, and click OK to create the mappings.  
 Mappings are created for all the child items that have **identical names** and adhere to the settings defined in the dialog box.

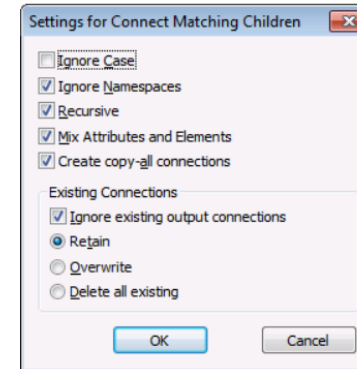
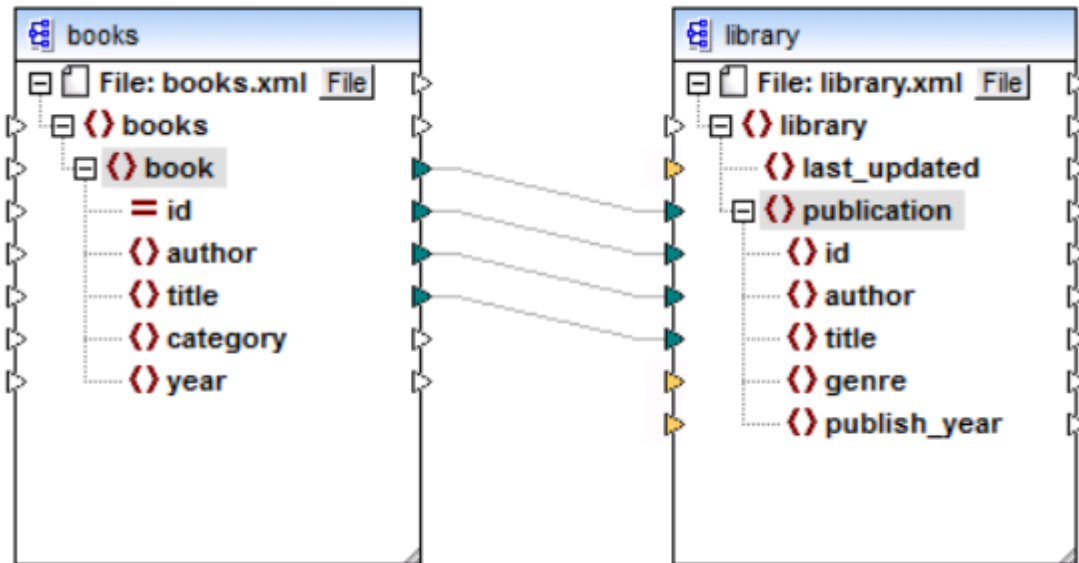
Felix Naumann  
 Information Integration  
 Winter 2019/20

# Altova MapForce 2015


## Step 4: Make the connections

For each <book> in the source XML file, we want to create a new <publication> in the target XML file. We will therefore create a mapping connection between the <book> element in the source component and the <publication> element in the target component. To create the mapping connection, click the output connector (the small triangle) to the right of the <book> element and drag it to the input connector of the <publication> element in the target.

When you do this, MapForce may automatically connect all elements which are children of <book> in the source file to elements having the same name in the target file; therefore, four connections are being created simultaneously. This behavior is called "Auto Connect Matching Children" and it can be disabled and customized if necessary.



Select the required options (see the table below), and click OK. Connections are created for all the child items that have identical names and adhere to the settings defined in the dialog box.

**Note:** The settings you define here are applied when connecting two items if the **Toggle auto connect of children** () toolbar button is active.

<i>Ignore Case</i>	Ignores the case of the child item names.
<i>Ignore Namespaces</i>	Ignores the namespaces of the child items.
<i>Recursive</i>	Creates new connections between any matching items recursively. That is, a connection is created no matter how deep the items are nested in the hierarchy, as long as they have the same name.
<i>Mix Attributes and Elements</i>	When enabled, allows connections to be created between attributes and elements which have the same name. For example, a connection is created if two "Name" items exist, even though one is an element, and the other is an attribute.
<i>Create copy-all connections</i>	This setting is active by default. It creates (if possible) a connection of type "Copy-all" between source and target items.
<i>Ignore existing output connections</i>	Creates additional connections for any matching items, even if they already have outgoing connections.
<i>Retain</i>	Retains existing connections.
<i>Overwrite</i>	Recreates connections according to the settings defined. Existing connections are discarded.

Felix Naumann  
Information Integration  
Winter 2019/20

## Schema Matching – Instance-based

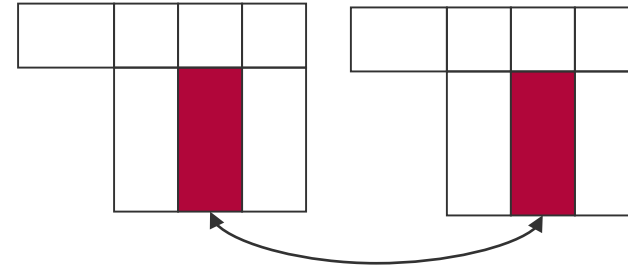
---

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

# Instanz-basiertes Schema Matching

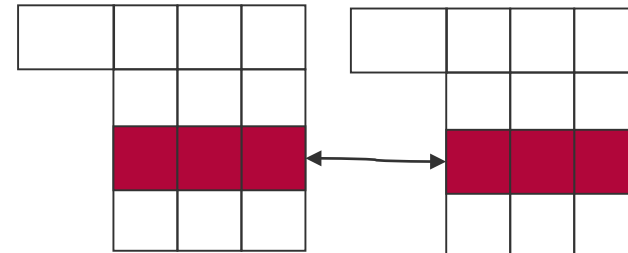
- **Herkömmliche Lösung: Vertikal**

- Vergleich von Spalten
- = **Attribut-Klassifikation**
- [ICDE'02] u.v.a.m.



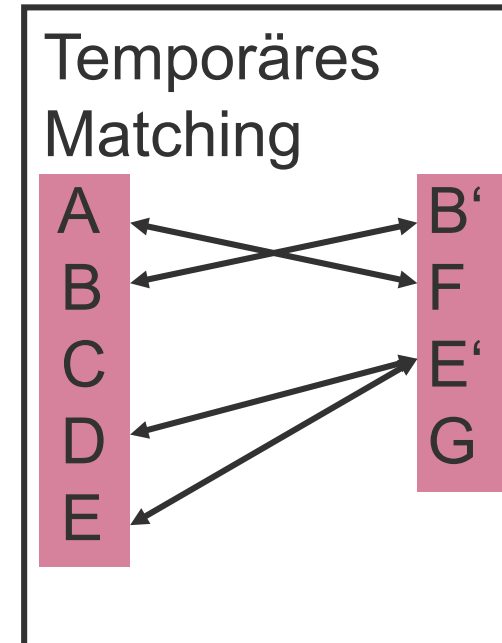
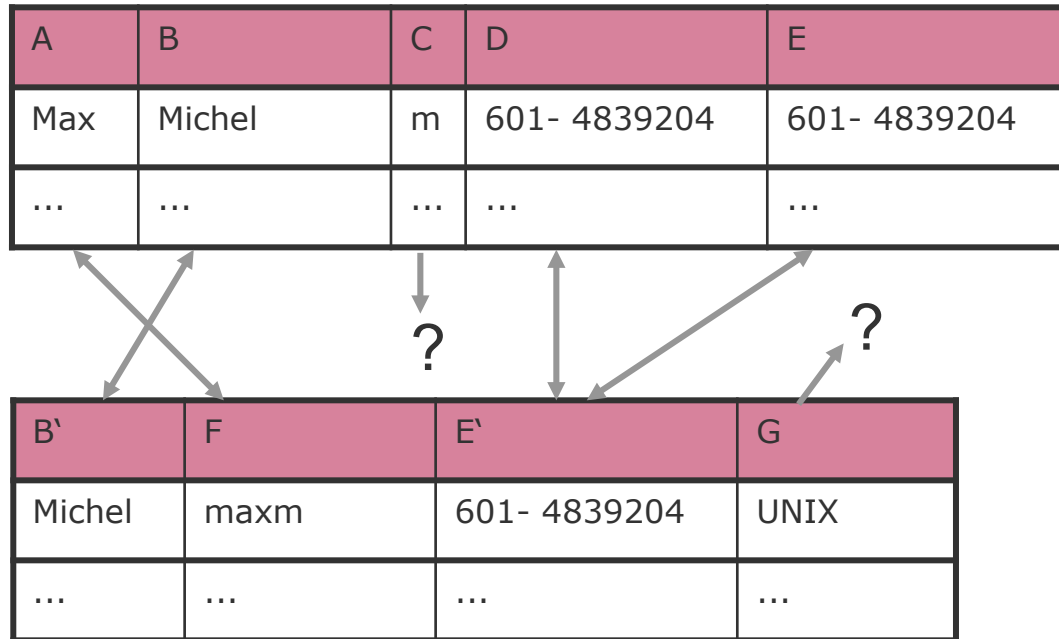
- **Neue Lösung: Horizontal**

- Vergleich von Zeilen
- = **Duplikaterkennung**
- trotz fehlender Attribut-korrespondenzen
- [ICDE'05]





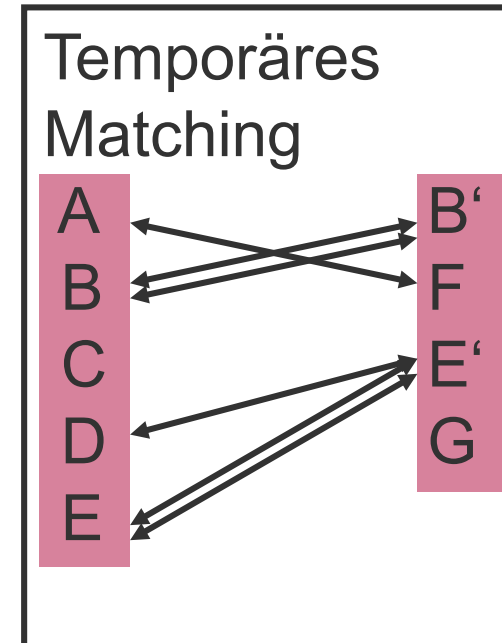
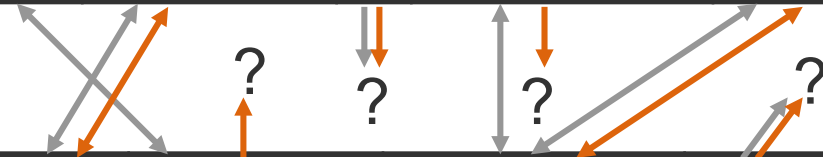
# Duplicate-driven Schema Matching



# Duplicate-driven Schema Matching

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	beer	541- 8127164	WinXP



## Annahmen

- Es existieren Daten in beiden DBs.
- Es existieren (wenigstens ein paar) Duplikate in beiden DBs.

## Schema Matching – Structure-based

---

- Gegeben zwei Schemata mit Elementmengen A und B.
- Kernidee
  - Nutze (komplexe) Struktur des Schemas aus.
  - Hierarchieebene
  - Elementtyp (Attribut, Relation, ...)
  - Nachbarschaftsbeziehungen

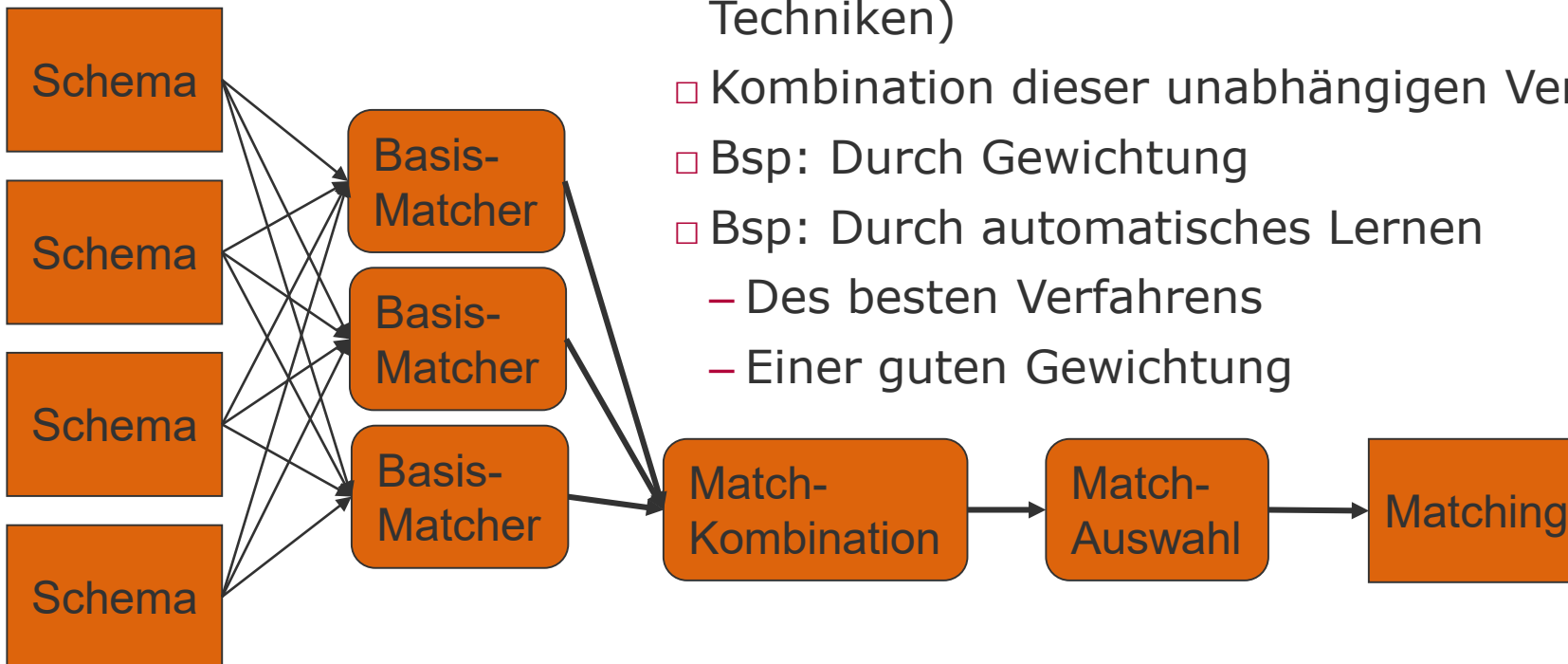
## Schema Matching – Structure-based

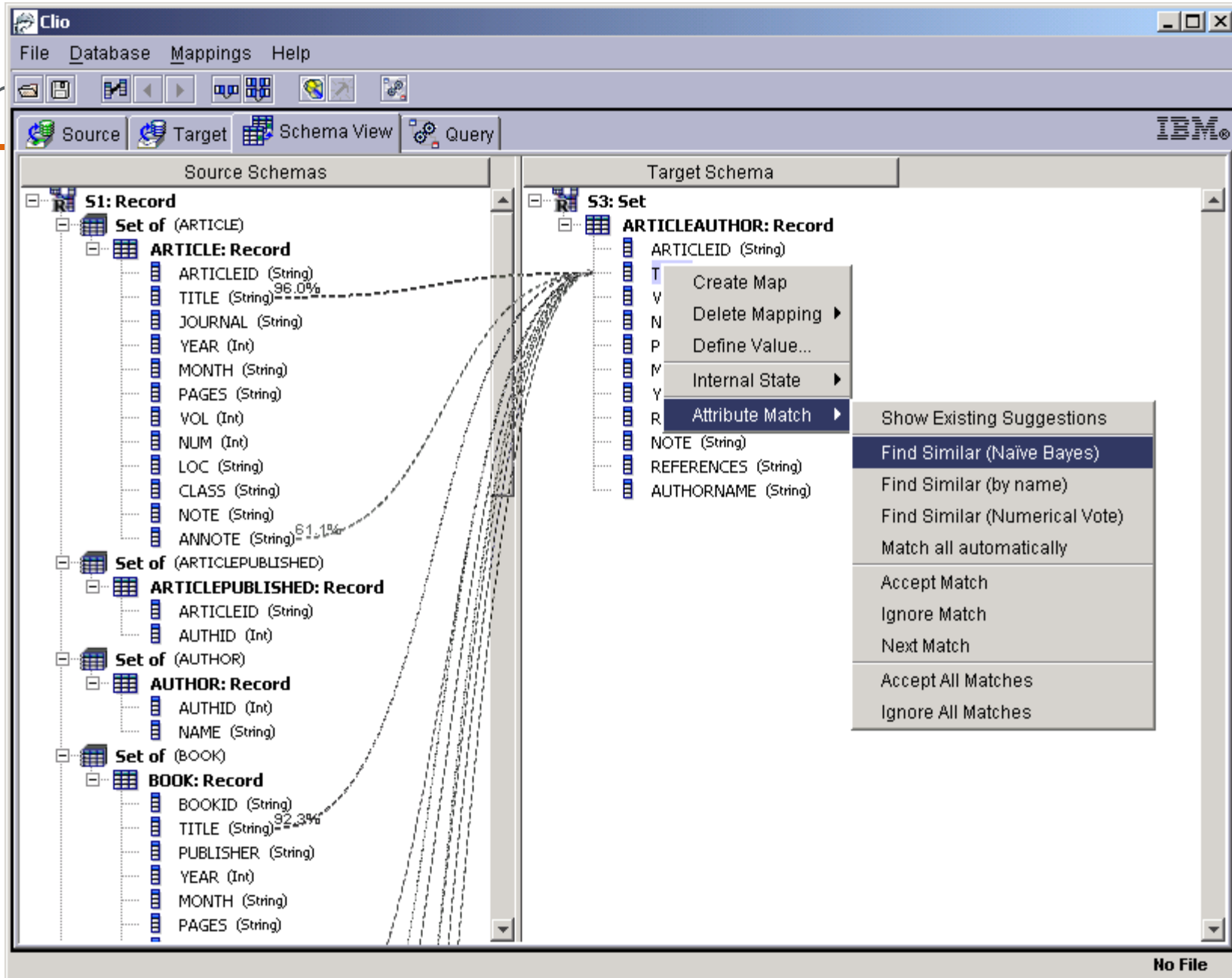
---

- Beispiel: Similarity Flooding nach [MGMR02]
  - Gegeben initiale Ähnlichkeit zwischen Schemaelementen (z.B. durch edit-distance oder durch Analyse der darunterliegenden Daten)
  - Lasse Ähnlichkeiten „abfärben“ auf die Nachbarn
    - Nachbarn sind durch Struktur definiert
    - Sind alle Nachbarn von  $x$  und  $y$  ähnlich zueinander, sind (vielleicht) auch  $x$  und  $y$  ein match.
  - Analogie: Man „flutet“ das Netzwerk der Ähnlichkeiten bis ein Gleichgewicht erreicht ist.

# Schema Matching – Mischformen

- Hybrid
  - Gleichzeitige Anwendung mehrerer Techniken
  - Bsp: Instance-based + Datentypvergleich
- Composite
  - Repertoire bekannter Techniken (inkl. hybrider Techniken)
  - Kombination dieser unabhängigen Verfahren
  - Bsp: Durch Gewichtung
  - Bsp: Durch automatisches Lernen
    - Des besten Verfahrens
    - Einer guten Gewichtung





# Überblick

1. Motivation
2. Schema Mapping
3. Schema Matching
  - Klassifikation von Schema Matching Methoden
  - **Erweiterungen**
  - Globales Matching
4. Mapping Interpretation
5. Mapping Werkzeuge



Felix Naumann  
Information Integration  
Winter 2019/20

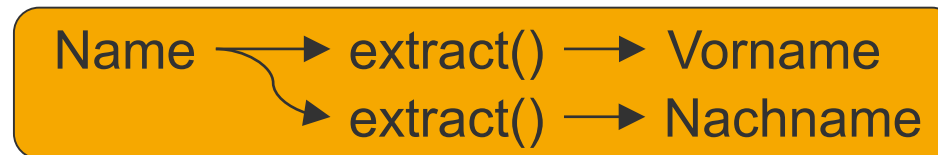
# Schema Matching – Erweiterungen

- n:1 und 1:n Matches
  - Viele Kombinationsmöglichkeiten
  - Viele Funktionen denkbar
    - Mathematische Operatoren, Konkatenation, etc.
  - Parsingregeln
- n:m Matching?
- Matching in komplexen Schemata
  - Ziel: Finde logisches Mapping, nicht nur Korrespondenzen

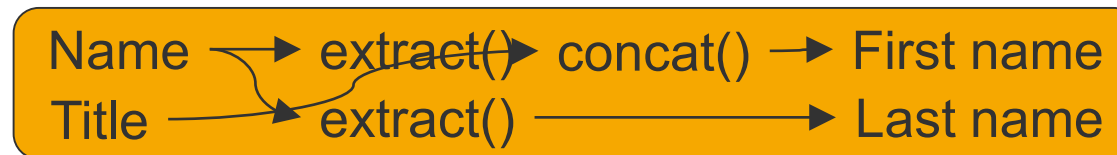
## n:1 Matching



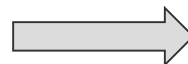
## 1:n Matching



## m:n matching



**Sigmund Freud** | **Prof. Dr.**



**Prof. Dr. Sigmund** | **Freud**



- Finden von 1:1 und komplexen Matchings
- Durchsucht den Raum aller möglichen Matches anhand von spezialisierten Searcher Modulen

- text searcher: `concat(Name, Vorname)`
- numeric searcher: `preis = preis + Mwst`
- date searcher: `datum = concat(Monat, „.“, Jahr)`

- Beam search:

- Anzahl kombinierter Attribute schrittweise erhöhen (lattice)
- In jedem Schritt nur mit k besten Ergebnissen weiterrechnen

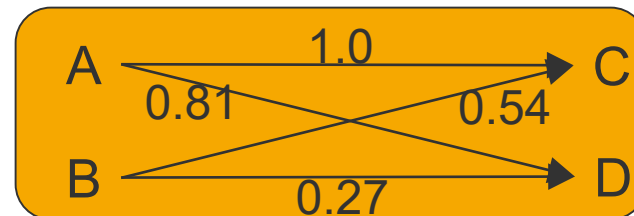
- Setzt frühzeitig Domänenwissen ein (Integritätsbedingungen)

- Attribute des Quellschemas stehen (nicht) in Beziehung
- Ein Attribut des Zielschemas muss einer Bedingung genügen
- Mehrere Attribute des Zielschemas stehen (nicht) in Beziehung

```
name = CONCAT(first_name, last_name)
test_economy_rate = 6 * t_runs_given / t_balls
ODI_overs = 0.1667 * o_balls
Marital_status = f(person_marital_stats)
                    Single=f(SIN) Married=f(MAR) Divorced=f(DIV)
fireplace = 1 if house_description contains fireplace
ODI_debut = (o_debut_day-o_debut_month-o_debut_year)
```

## Schema Matching – Erweiterungen

- Globales matching (gleich)
  - Matche nicht nur einzelne Attribute (oder Attributmengen)
  - Sondern komplette Tabellen oder komplette Schemata
  - Stable Marriage Problem



## Schema Matching – Weitere Anwendungen

---

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

## Schema Matching Kritik

---

- Few (public) use cases
- No good benchmarks
  - Too many and too toy-ish
  - Minimum requirement: 2 interesting schemata + mapping
- Few available prototypes
- No commercial implementation
  - To speak of
- Real schemata are too large
  - Toy examples work fine
  - Screen not wide or long enough
  
- The YAM Effect
  - YAM – Yet Another Matcher

- Past goal: Improved precision and recall
  - Big productivity gains are unlikely
  
- Better goals
  - Return top-k, not best overall match
  - Avoid the tedium. Manage work.
    - Help with scrolling
  - HCI – handle large schemata
  - User studies – what would improve productivity?

# Überblick

1. Motivation
2. Schema Mapping
3. Schema Matching
  - Klassifikation von Schema Matching Methoden
  - Erweiterungen
  - **Globales Matching**
4. Mapping Interpretation
5. Mapping Werkzeuge



Felix Naumann  
Information Integration  
Winter 2019/20

## Schema Matching – Stable Marriage

---

- Gegeben
  - $n$  Frauen (Attribute in Schema A) und  $m$  Männer (Attribute in Schema B)
- Monogamie
  - Je eine Frau kann nur mit je einem Mann verheiratet sein: nur 1:1 matches
- Jede Frau hat eine Rangliste der Männer und umgekehrt
  - Bei Schema Matching
    - Attribut-Ähnlichkeit gemäß eines der vorigen Verfahren
    - Rangliste ist (normalerweise) symmetrisch
- Gesucht: Paarung (globales Matching), so dass niemals gilt
  - $f_1$  heiratet  $m_1$ ,  $f_2$  heiratet  $m_2$ ,
  - aber  $f_1$  bevorzugt  $m_2$  und  $m_2$  bevorzugt  $f_1$  (instabil!)

## Stable Marriage – Beispiel

---

Männer (1-4)

1: B, D, A, C

2: C, A, D, B

3: B, C, A, D

4: D, A, C, B

Frauen (A-D)

A: 2, 1, 4, 3

B: 4, 3, 1, 2

C: 1, 4, 3, 2

D: 2, 1, 4, 3

Beispiel aus: David Toth,

["The Stable Marriage Problem: More Marital Happiness than Reality TV"](#)

April 25, 2003, Connecticut College, New London, CT, USA,



## Stable Marriage – Beispiel

---

Männer (1-4)

1: B, D, A, C

2: C, A, D, B

3: B, C, A, D

4: D, A, C, B

Frauen (A-D)

A: 2, 1, 4, 3

B: 4, 3, 1, 2

C: 1, 4, 3, 2

D: 2, 1, 4, 3

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

## Stable Marriage – Beispiel

---

Männer (1-4)

1: B, D, A, C

2: C, A, D, B

3: B, C, A, D

4: D, A, C, B

Frauen (A-D)

A: 2, 1, 4, 3

B: 4, 3, 1, 2

C: 1, 4, 3, 2

D: 2, 1, 4, 3

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

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

## Stable Marriage – Beispiel

---

Männer (1-4)

1: ~~B~~, D, A, C

2: C, A, D, B

3: B, C, A, D

4: D, A, C, B

Frauen (A-D)

A: 2, 1, 4, 3

B: 4, 3, 1, 2

C: 1, 4, 3, 2

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)

## Stable Marriage – Beispiel

---

Männer (1-4)

1: ~~B~~, D, A, C

2: C, A, D, B

3: B, C, A, D

4: D, A, C, B

Frauen (A-D)

A: 2, 1, 4, 3

B: 4, 3, 1, 2

C: 1, 4, 3, 2

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)

# Stable Marriage – Beispiel

Männer (1-4)

1: ~~B~~, D, A, C

2: C, A, D, B

3: B, C, A, D

4: ~~D~~, A, C, B

Frauen (A-D)

A: 2, 1, 4, 3

B: 4, 3, 1, 2

C: 1, 4, 3, 2

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)

## Stable Marriage – Beispiel

Männer (1-4)

1: ~~B~~, D, A, C

2: C, A, D, B

3: B, C, A, D

4: ~~D~~, A, C, B

Frauen (A-D)

A: 2, 1, 4, 3

B: 4, 3, 1, 2

C: 1, 4, 3, 2

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)

# Vier stabile Paare!

## Stable Marriage – Diskussion

---

- *Stable Roommates Problem:*  
[https://de.wikipedia.org/wiki/Stable\\_Roommates\\_Problem](https://de.wikipedia.org/wiki/Stable_Roommates_Problem)
  - Keine Unterscheidung von Männern und Frauen
- Unterschiedliche Schemagrößen
  - Vorschläge müssen vom kleineren Schema kommen
  - Oder: Dummy Schema-elemente mit geringen Werten / rankings

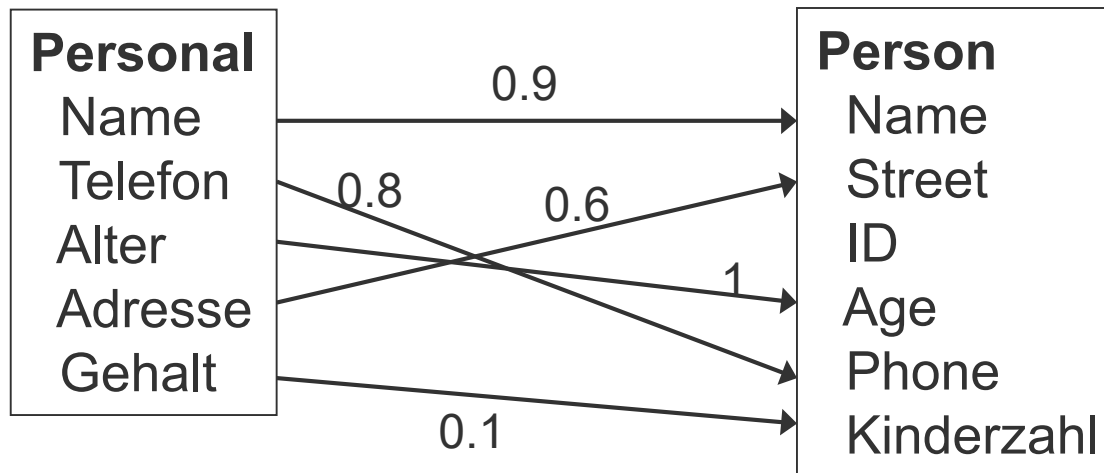
## Maximum Weighted Matching

---

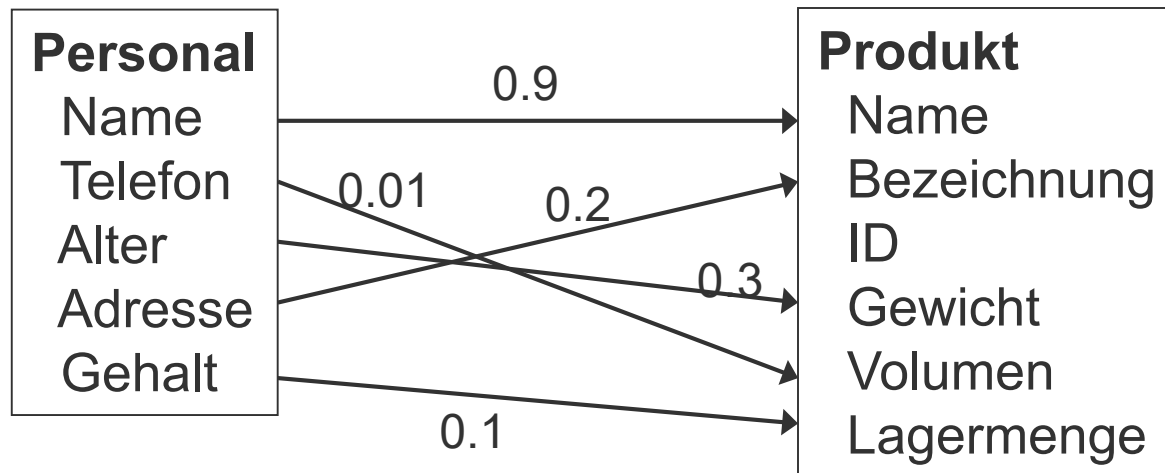
- Alternative zu Stable Marriage
  
- Suche Matching mit maximalem Gewicht in bipartiten Graphen
  - Bipartit:
    - Knoten in zwei Klassen (Quelle & Ziel)
    - Kanten nur zwischen Knoten verschiedener Klassen (Korrespondenzen)
  - Maximiere Summe der einzelnen Gewichte/Ähnlichkeiten
  
- $O(n^3)$  („Ungarische Methode“)



# Diskussion: Globales Matching



# Diskussion: Globales Matching



Felix Naumann  
Information Integration  
Winter 2019/20

## Zusammenfassung – Schema Matching

---

- Schema Matching basierend auf
  - Namen der Schemaelemente (label-based)
  - Darunterliegende Daten (instance-based)
  - Struktur des Schemas (structure-based)
  - Mischformen, Meta-Matcher
- High-order Matching
- n:m Matching
- Globales Matching

## Überblick

1. Motivation
2. Schema Mapping
3. Schema Matching
- 4. Mapping Interpretation**
5. Mapping Werkzeuge

Aus [FHP+02] und VLDB 2002 Vortragsfolien

Felix Naumann  
Information Integration  
Winter 2019/20

## Mapping – Das Problem

---

- Gegeben: Zwei Schemata
  - Unabhängig voneinander erzeugt
  - Relational
  - Geschachtelt
  - Mit Integritätsbedingungen (Schlüssel/Fremdschlüssel)
  - Stellen teilweise unterschiedliche Daten dar
- Gegeben: Eine Menge von Korrespondenzen zwischen den Schemata
- Gesucht: Anfrage, die Daten des einen in Daten des anderen Schemas transformiert, wobei
  - Semantik des Quellschemas erhalten bleibt,
  - Integritätsbedingungen des Zielschemas berücksichtigt werden,
  - und möglichst alle Korrespondenzen berücksichtigt werden.

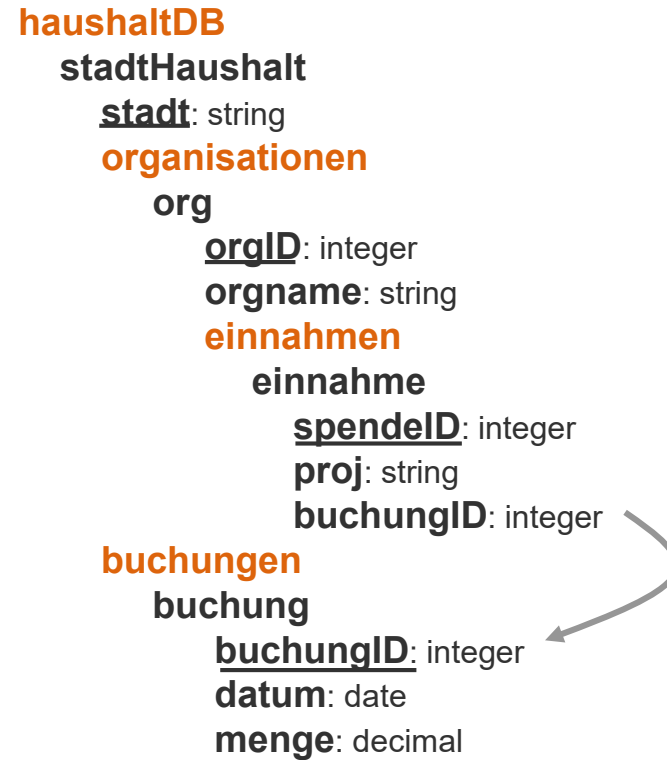
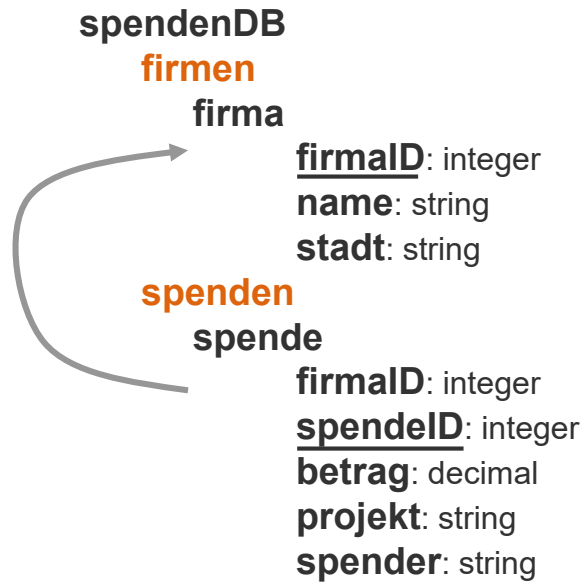
# Relationale vs. XML Schemata

- Relationale Schemata
  - Flach
- XML Schemata
  - Flach oder geschachtelt
- NF2 bzw. NF<sup>2</sup>

Employee

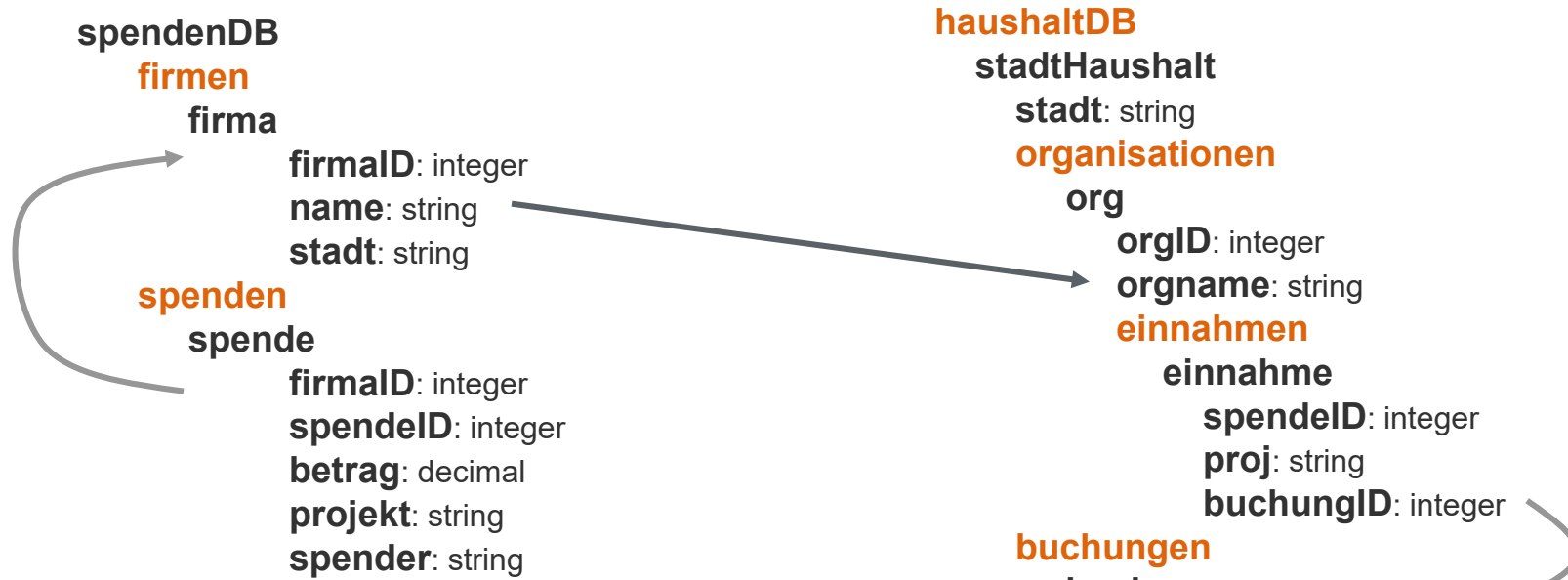
ename	Children		Skills		
	name	dob	type	Exams	
				year	city
Smith	Sam	2/10/84	typing	1984	Atlanta
	Sue	1/20/85		1985	Dallas
			dictation	1984	Atlanta
Watson	Sam	3/12/78	filing	1984	Atlanta
				1975	Austin
				1971	Austin
			typing	1962	Waco

# Mapping – Beispiel



Quelle für Beispiel: [FHP+02]

# Mapping – Beispiel

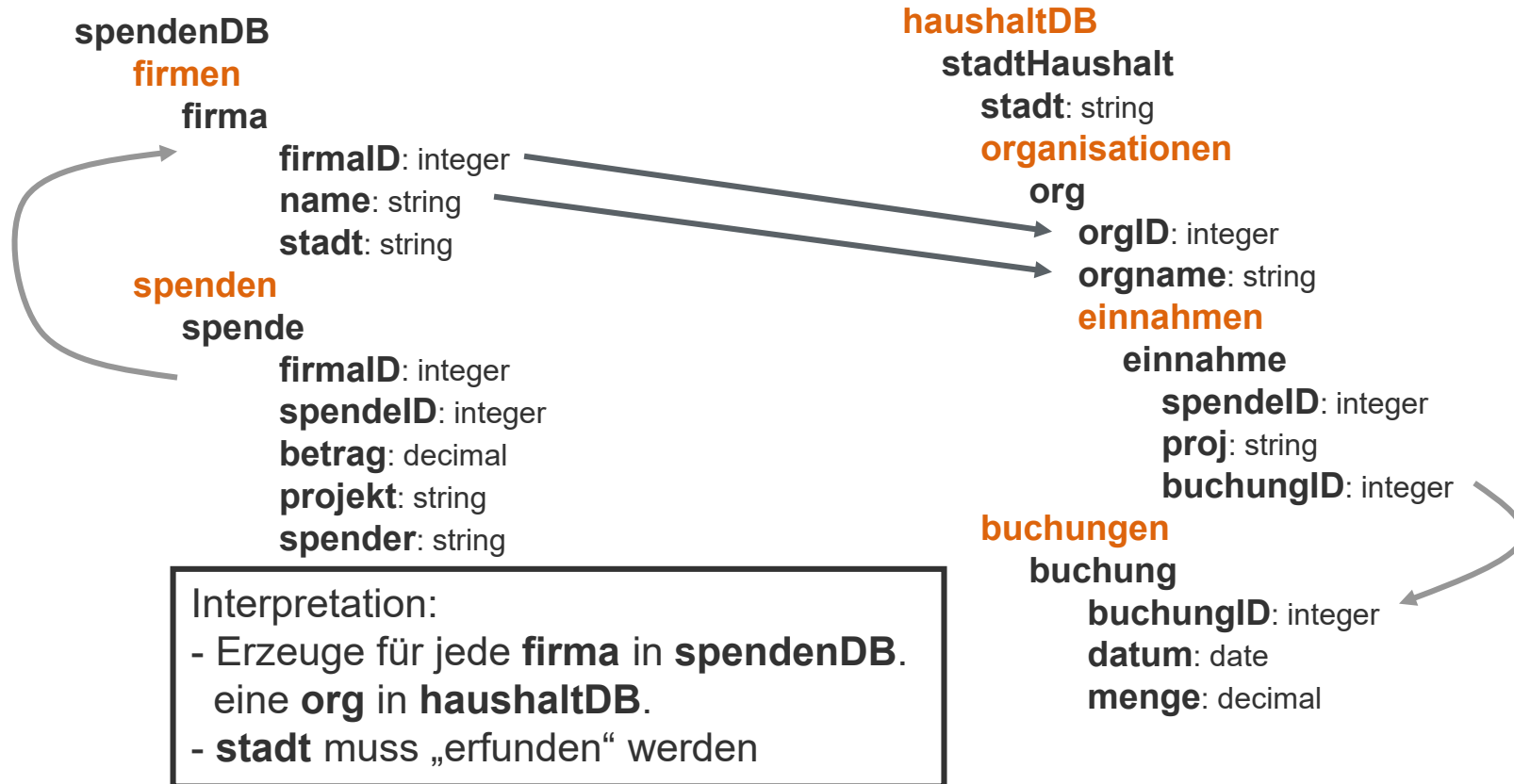


Interpretation:

- Erzeuge für jede **firma** in **spendenDB**. eine **org** in **haushaltDB**.
- **orgID** muss „erfunden“ werden
- **stadt** muss „erfunden“ werden



# Mapping – Beispiel



# „Erfinden“ von Werten

- Zwei Gründe zum Erfinden: „not-null“ und Identität
- „not-null“ Werte
  - Erfundener Wert egal
  - Z.B. „unbekannt“ oder „null“ (oder „Berlin“)
- ID Werte
  - Skolemfunktion:
    - Input: n Werte (beliebige Domäne)
    - Output: bezgl. Input eindeutiger Wert (beliebiger Domäne)
    - Beispiel: Konkatenation aller Inputwerte als String (mit Trenner)

## firma

**firmaID:** integer  
**name:** string  
**stadt:** string

**stadt:** string

**organisationen**

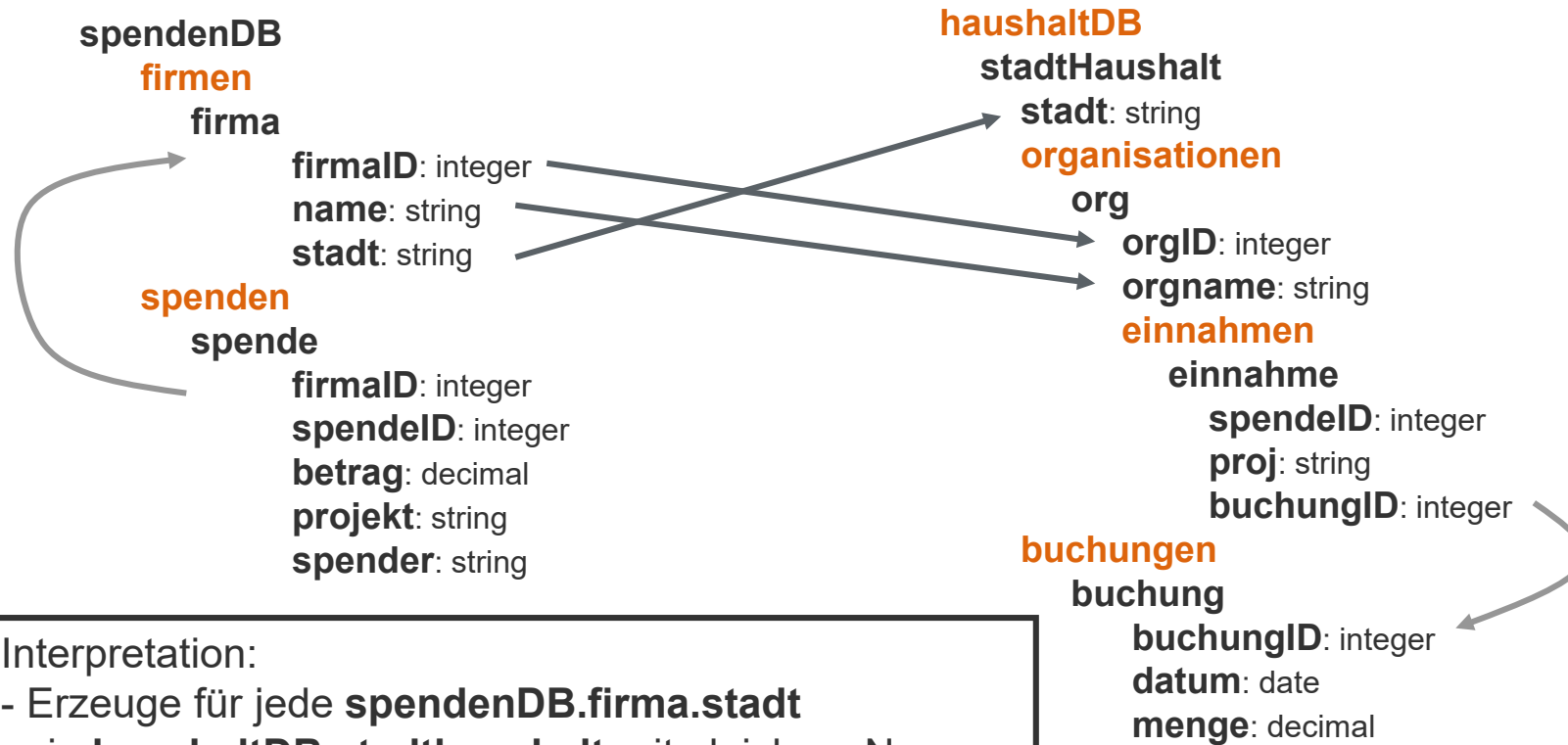
**org**

**orgID:** integer

**orgname:** string

Wert für org.orgID nicht egal, sondern je nach firma.name eindeutig!

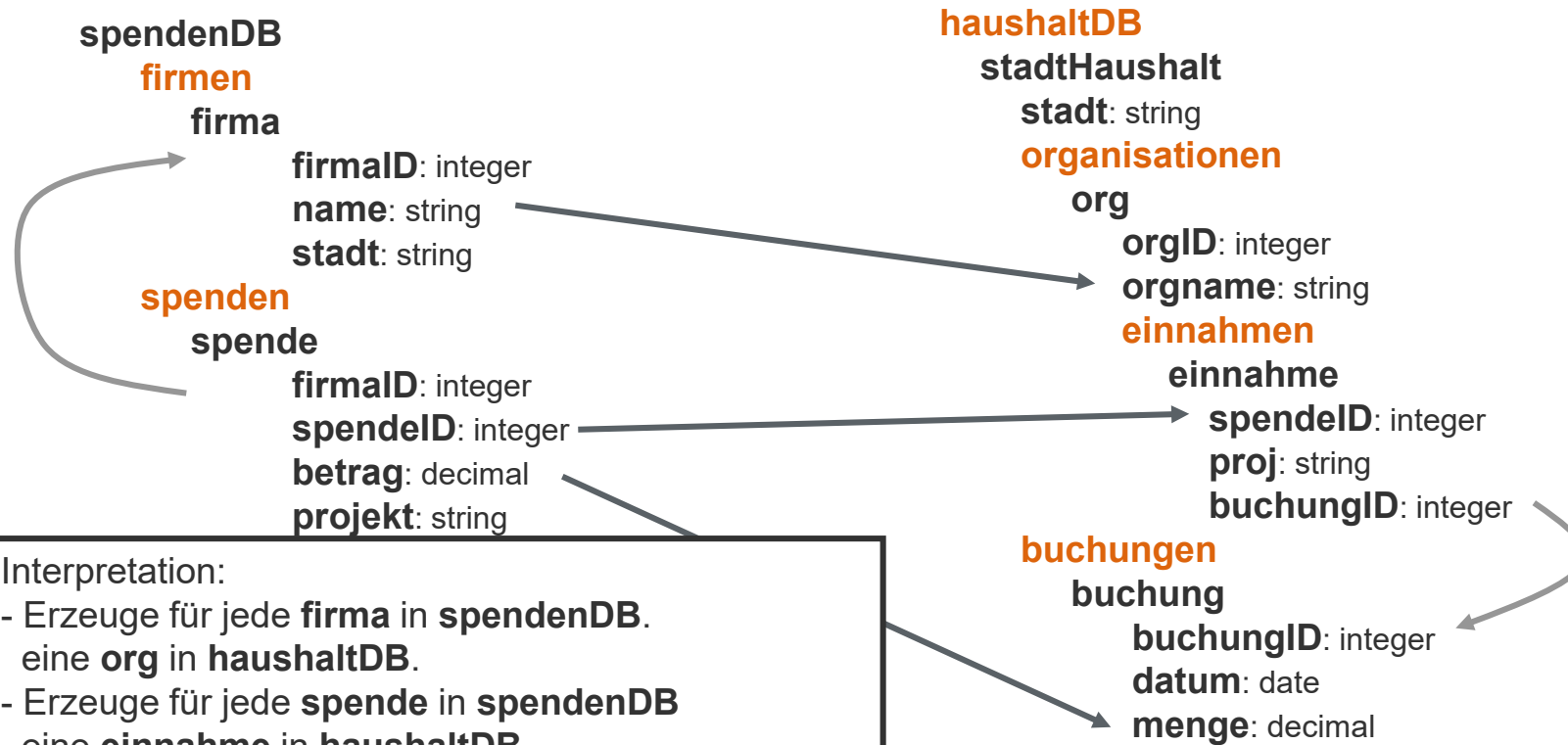
# Mapping – Beispiel



Interpretation:

- Erzeuge für jede **spendenDB.firma.stadt** ein **haushaltDB.stadthaushalt** mit gleichem Namen.
- Gruppriere jede **firma** unter den entsprechenden **stadtHaushalt**.

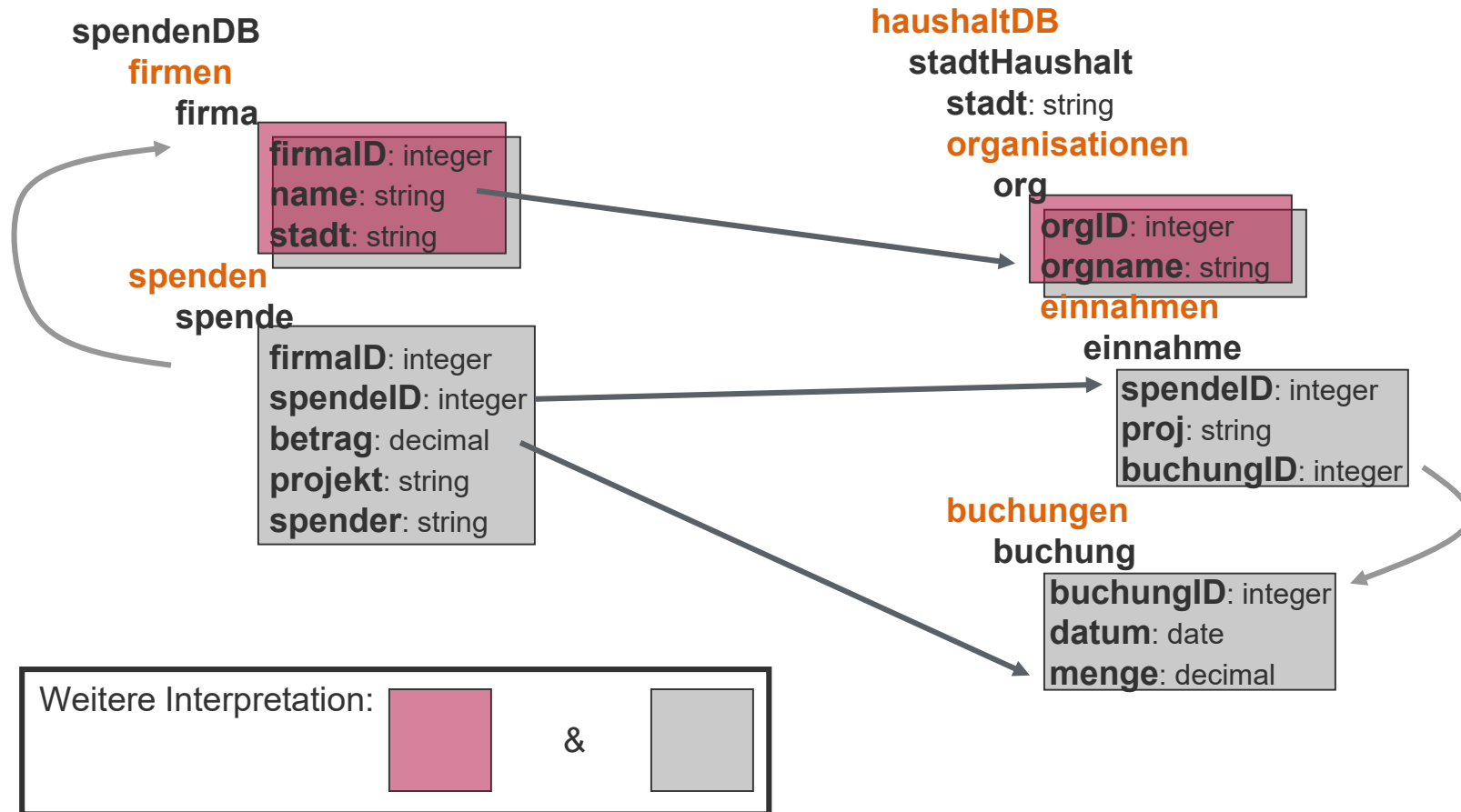
# Mapping – Beispiel



Interpretation:

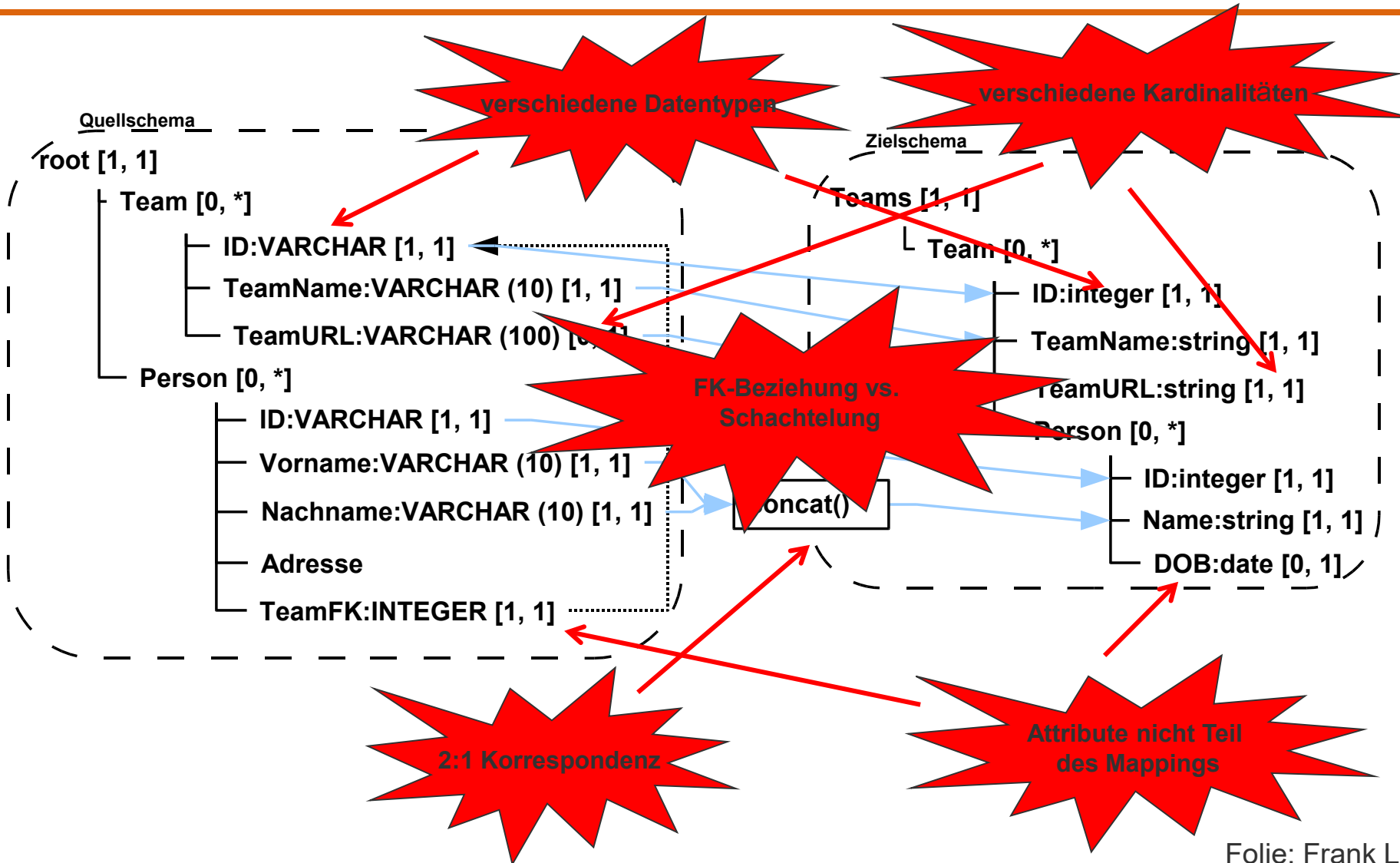
- Erzeuge für jede **firma** in **spendenDB**. eine **org** in **haushaltDB**.
- Erzeuge für jede **spende** in **spendenDB** eine **einnahme** in **haushaltDB**
- Erzeuge für jede **spende** in **spendenDB** eine **buchung** in **haushaltDB**
- Gruppriere korrekt: Schachtelung & Fremdschlüssel!

# Mapping – Beispiel



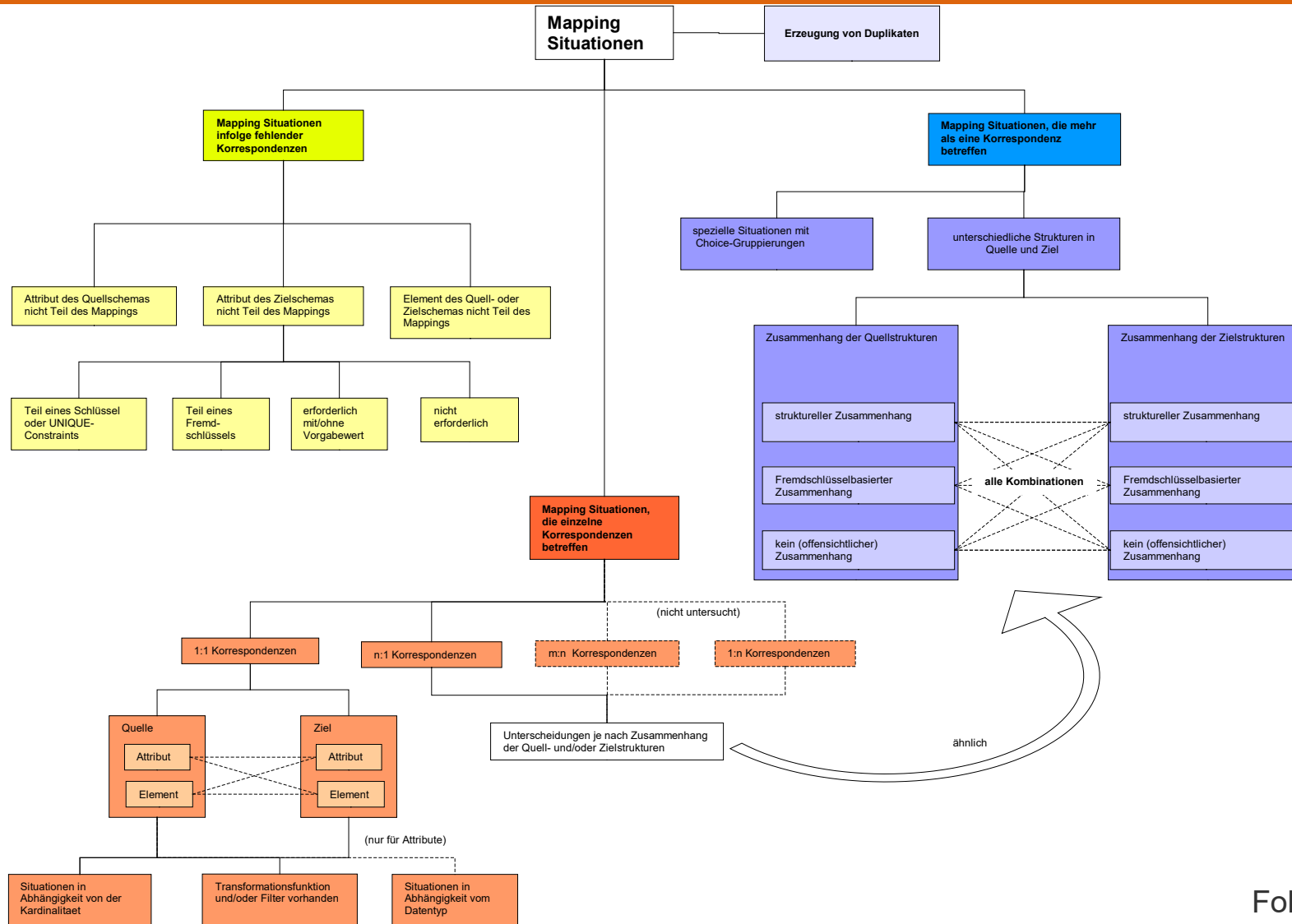
Felix Naumann  
Information Integration  
Winter 2019/20

# Verschiedene Mapping Situationen



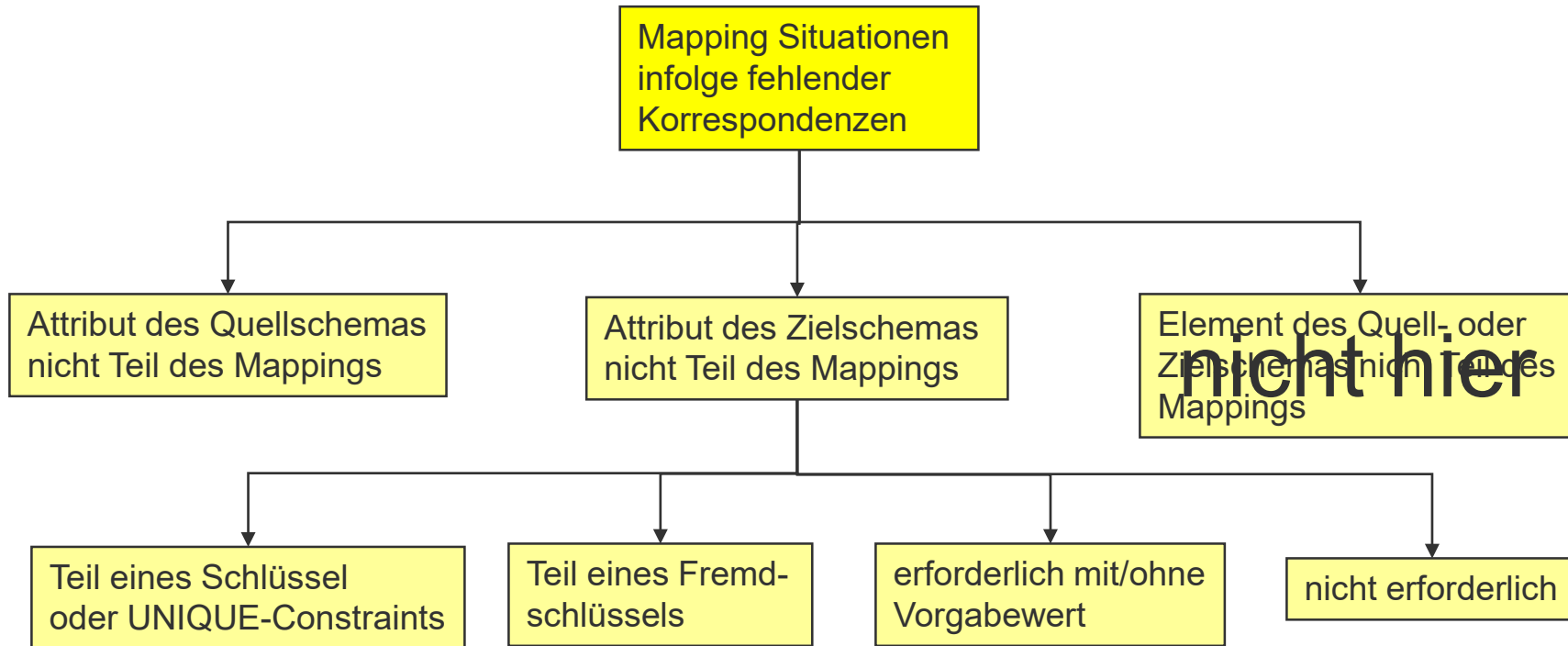
Felix Naumann  
Information Integration  
Winter 2019/20

# Klassifikation von Mapping Situationen



Felix Naumann  
Information Integration  
Winter 2019/20

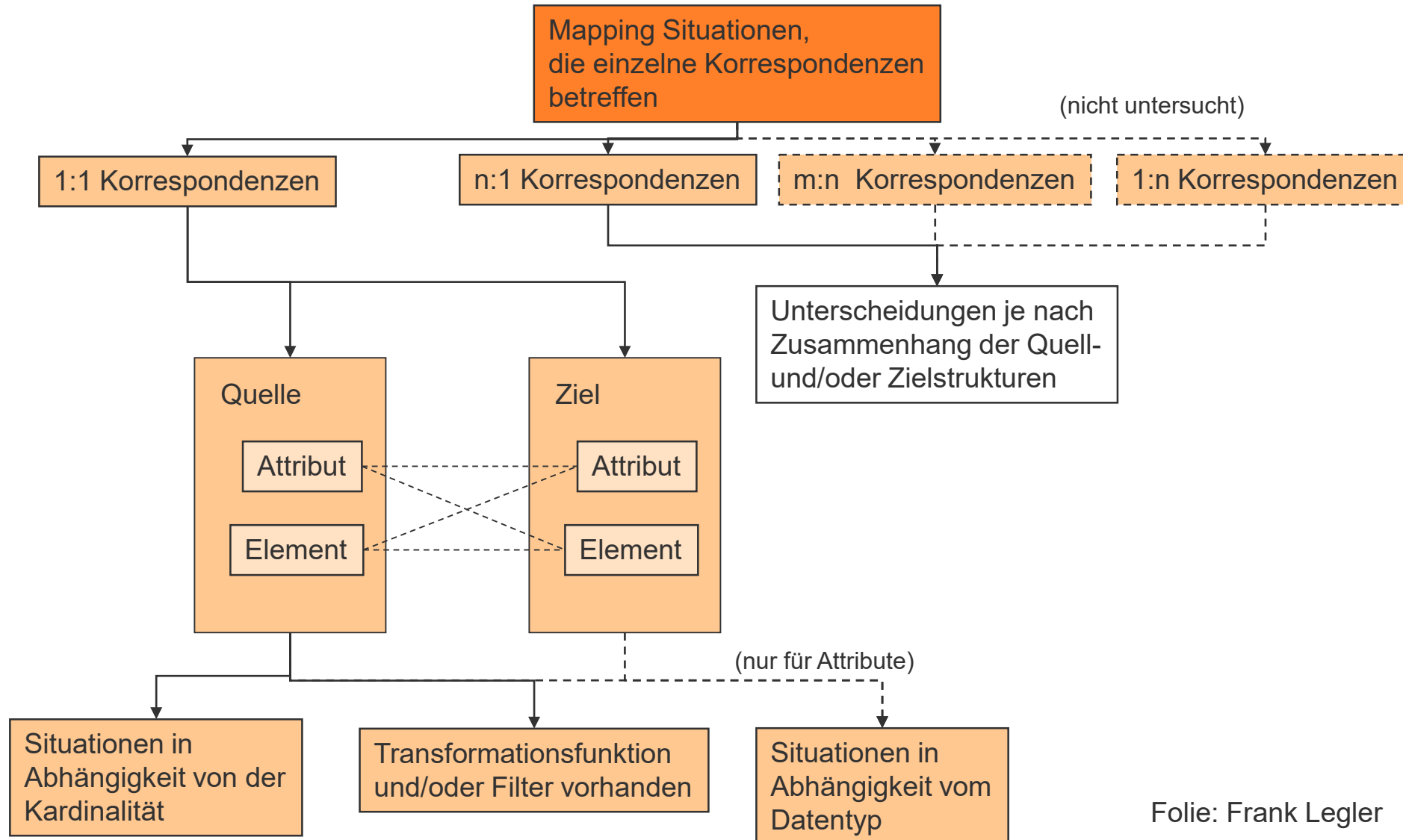
# Klassifikation von Mapping Situationen



Felix Naumann  
Information Integration  
Winter 2019/20

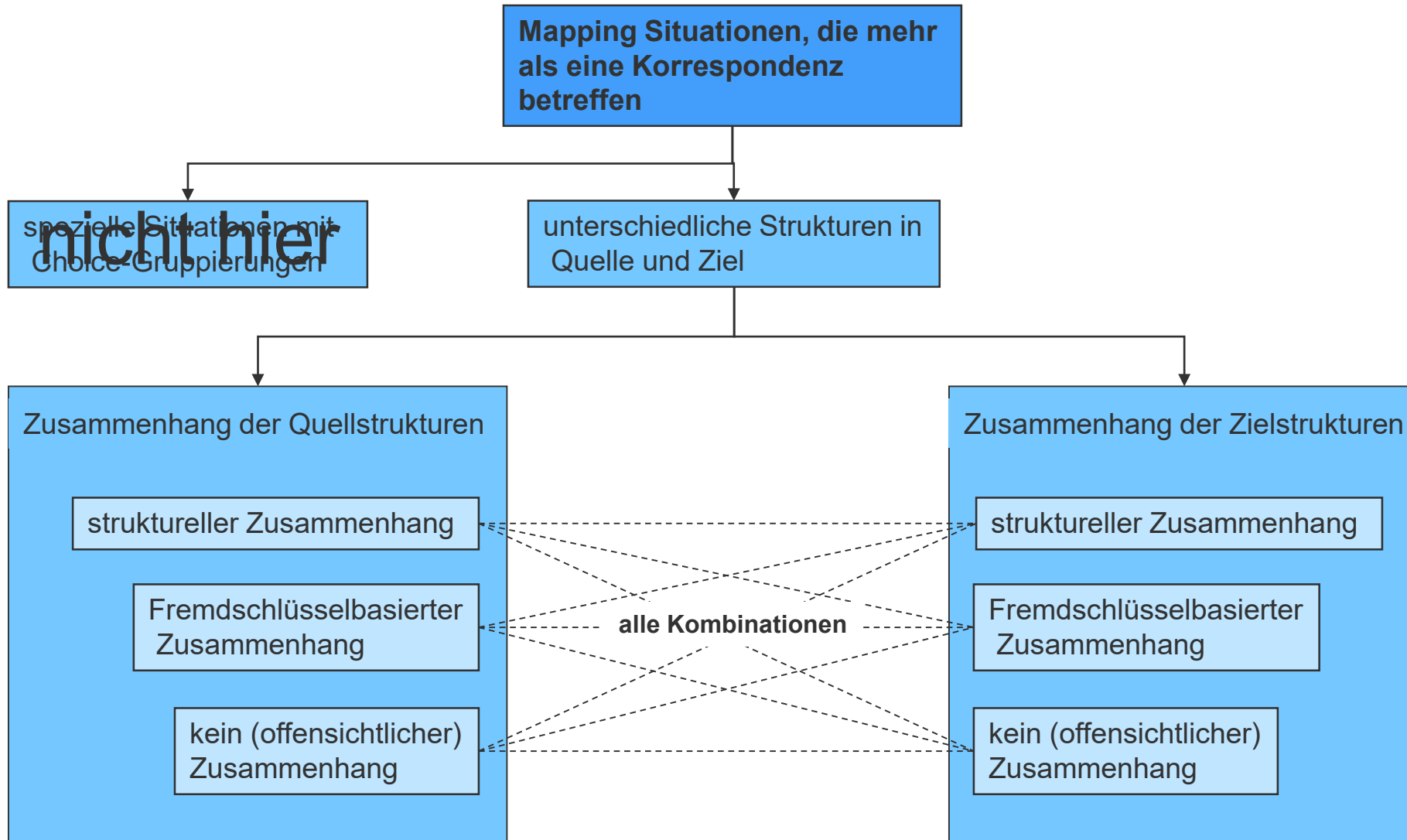


# Klassifikation von Mapping Situationen



Felix Naumann  
Information Integration  
Winter 2019/20

# Klassifikation von Mapping Situationen

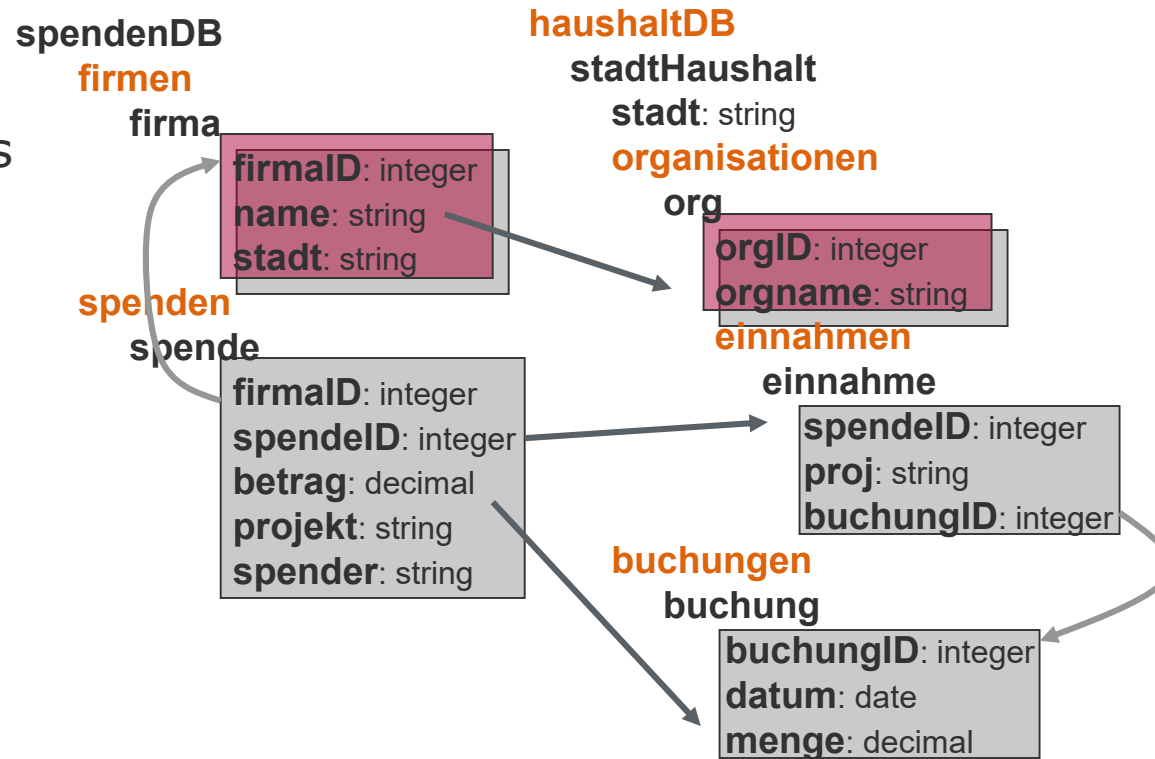


Felix Naumann  
Information Integration  
Winter 2019/20

# Mapping – Algorithmus

■ Drei Schritte

1. Entdeckung von intra-Schema Assoziationen
2. Entdeckung von inter-Schema logischen Mappings
3. Anfrage-erzeugung



Lucian Popa

Felix Naumann  
Information Integration  
Winter 2019/20

# Entdeckung von Assoziationen

## ■ Schritt 1

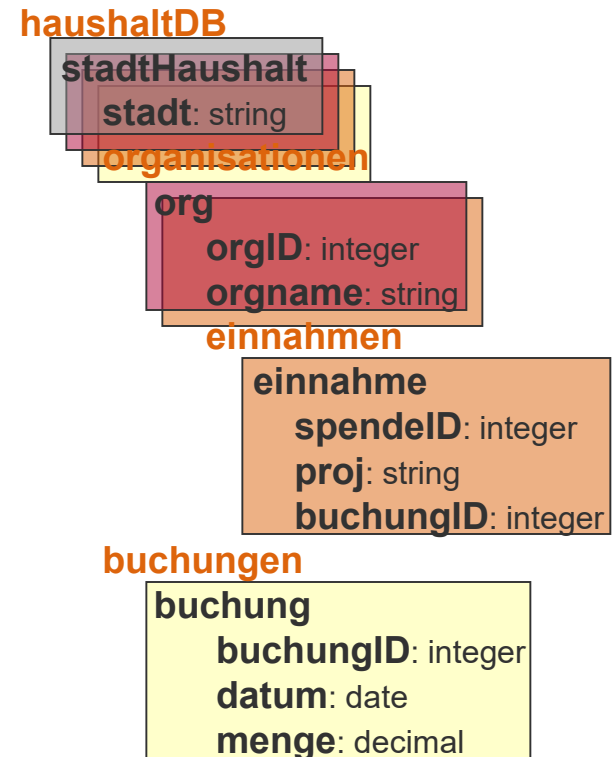
- Intra-schema Assoziationen zwischen Schemaelementen
- Relationale Sichten enthalten maximale Gruppen assoziierter Elemente
- Jede Sicht repräsentiert eine eigene „Kategorie“ an Daten der Datenquelle
- Unabhängig vom Mapping (aber beschränkt auf „gemappte“ Elemente)



Felix Naumann  
Information Integration  
Winter 2019/20

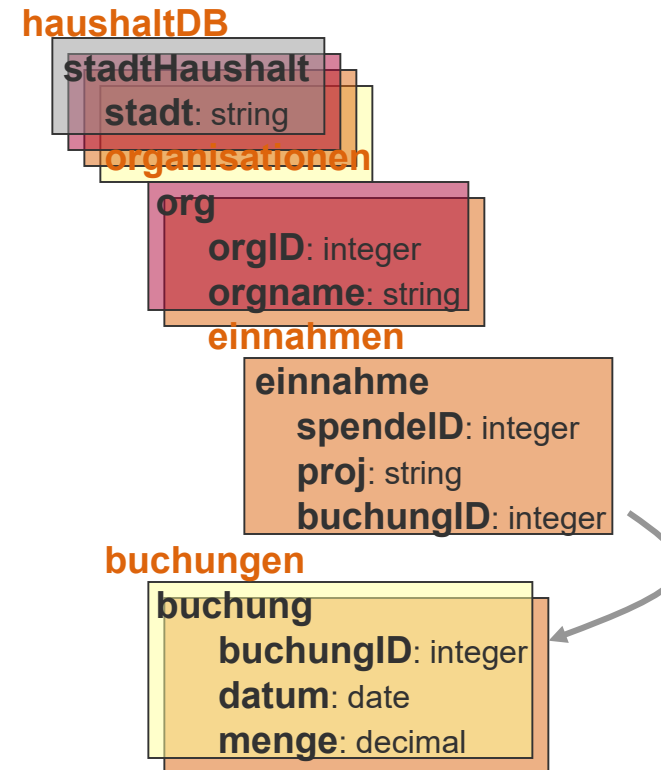
# Entdeckung von Assoziationen

- Start: Alle „primären“ Pfade (*primary paths*)
  - Assoziationen im Schema ohne Integritätsbedingungen
- Relationale Schemata
  - Jede Relation entspricht einem primären Pfad
- Geschachtelte Schemata
  - Attribute einer Ebene
  - Attribute geschachtelter Ebenen



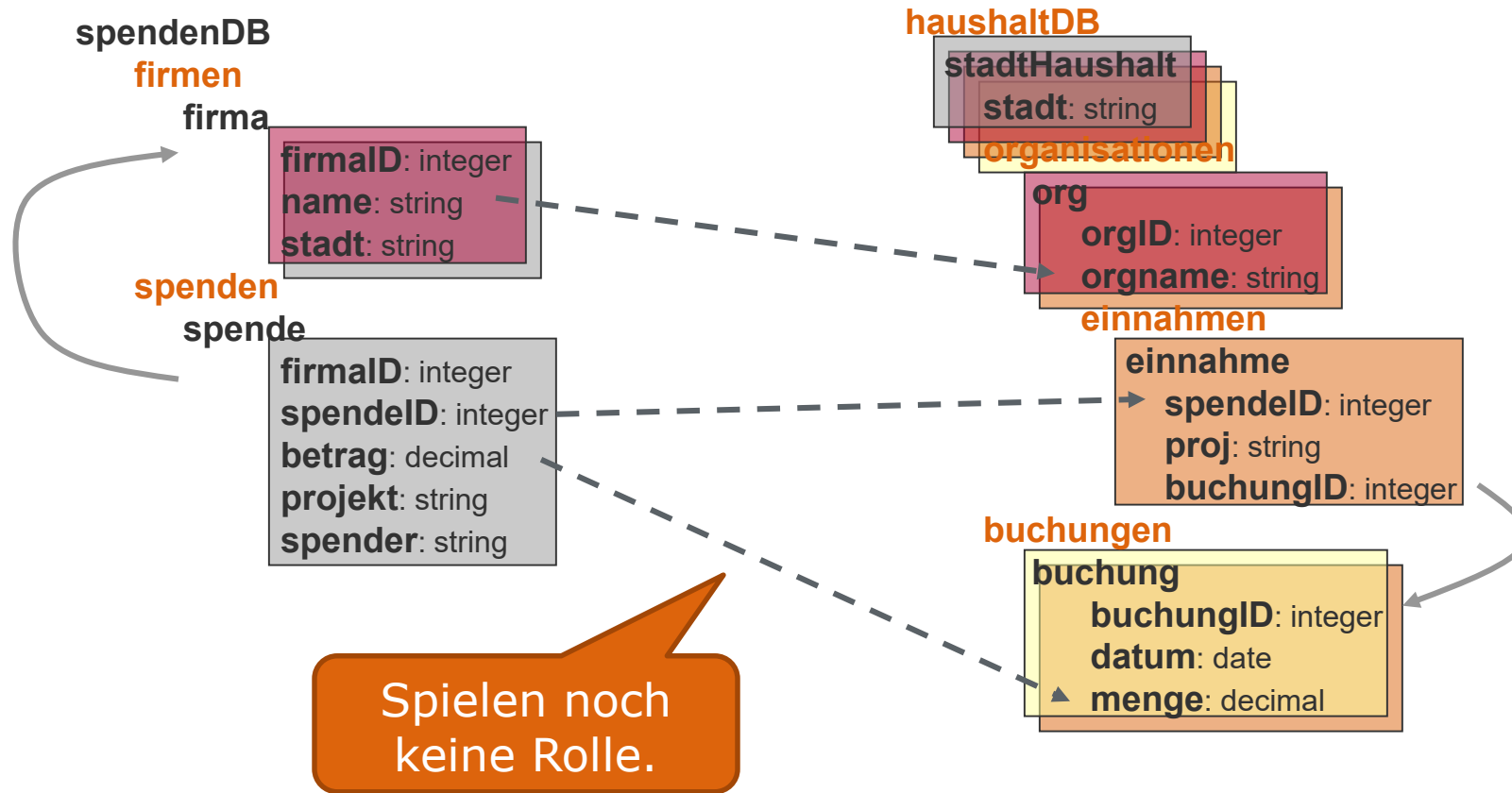
## Entdeckung von Assoziationen

- Betrachte nun Schlüssel / Fremdschlüssel (ICs)
- Logische Relation
  - Erweitere jeden primären Pfad durch „Verfolgen“ der ICs (*chase*)



Felix Naumann  
Information Integration  
Winter 2019/20

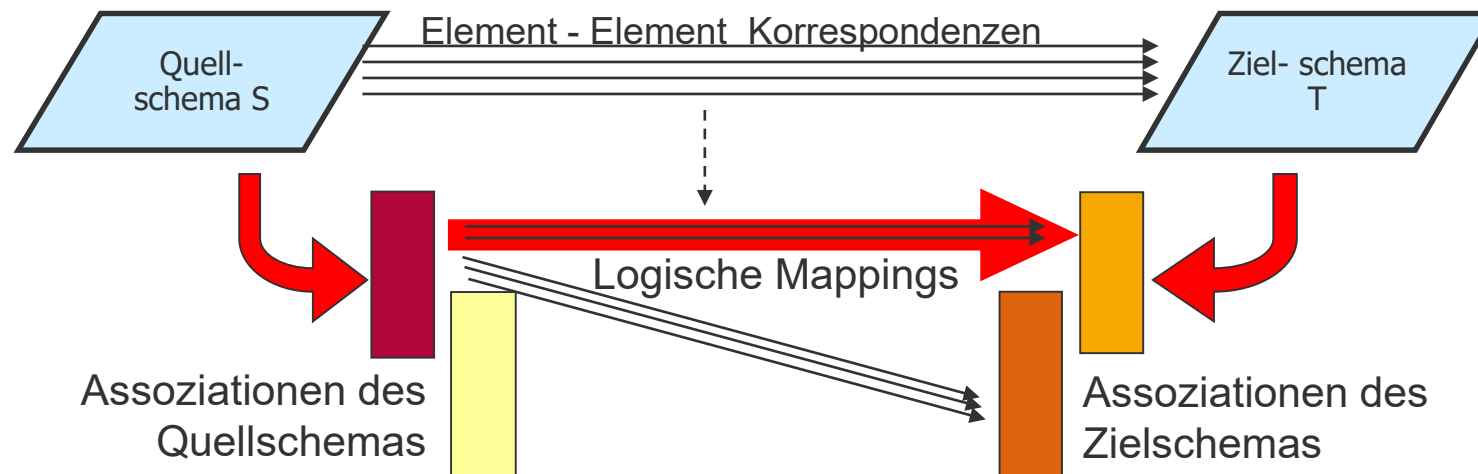
# Entdeckung von Assoziationen



## Entdeckung von logischen Mappings

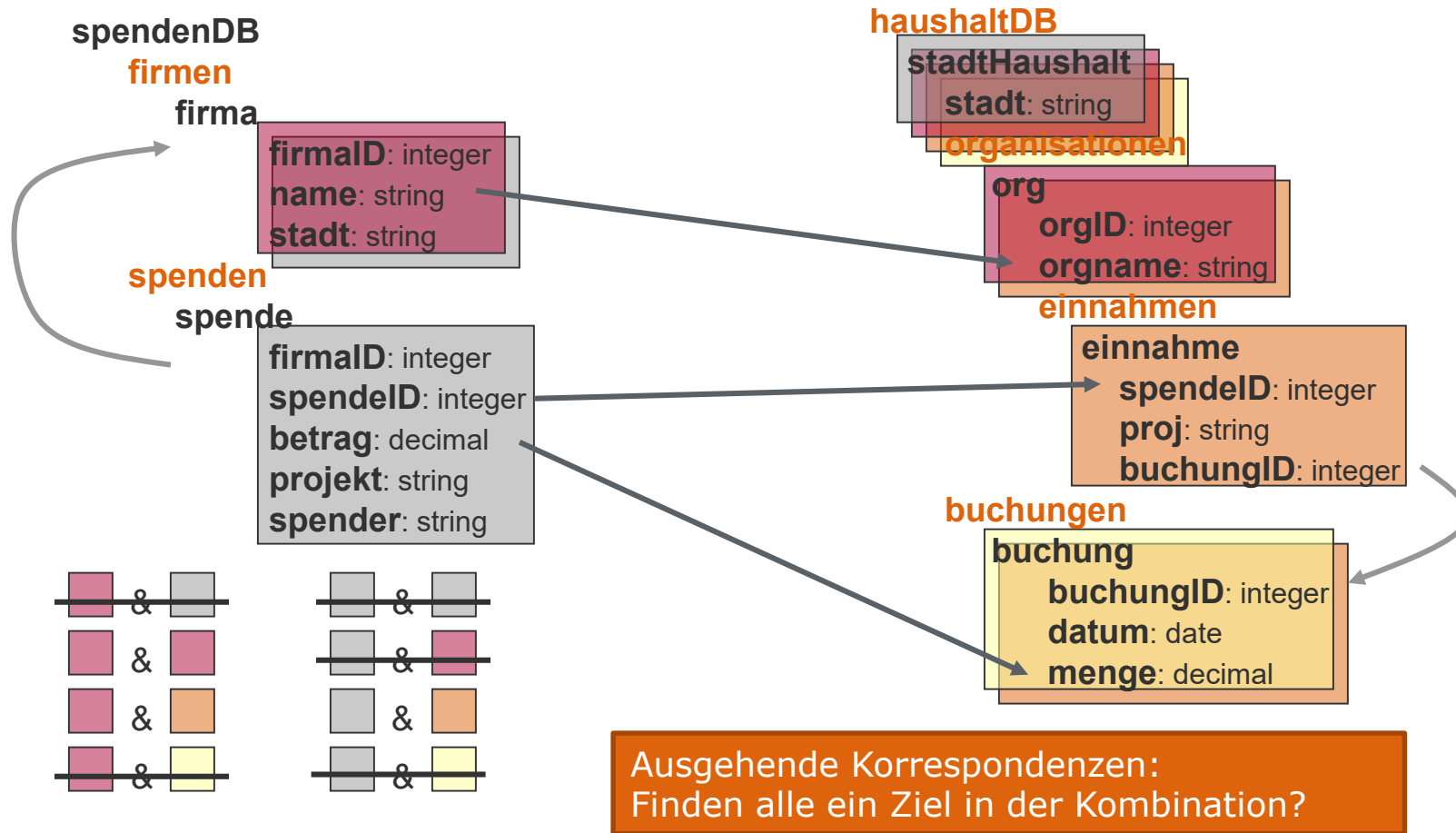
### ■ Schritt 2

- Entdecke logische Mappings zwischen Quell- und Zielschema
- Betrachte alle Kombinationen aus Assoziationen des Quellschemas und Assoziationen des Zielschemas
  - Unter Berücksichtigung aller Korrespondenzen (sofern Korrespondenzen zwischen ihnen überhaupt existieren)

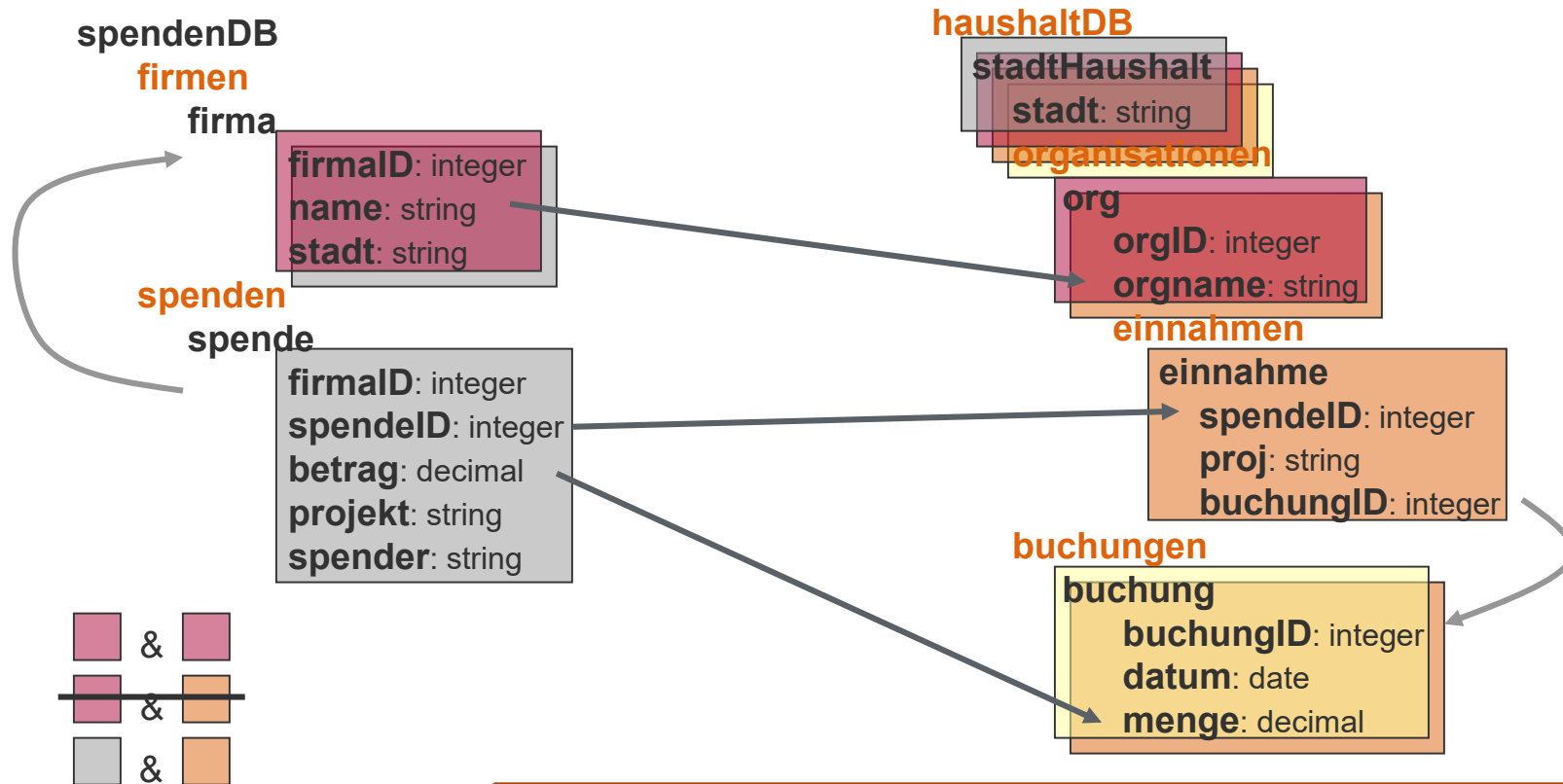




# Entdeckung von logischen Mappings

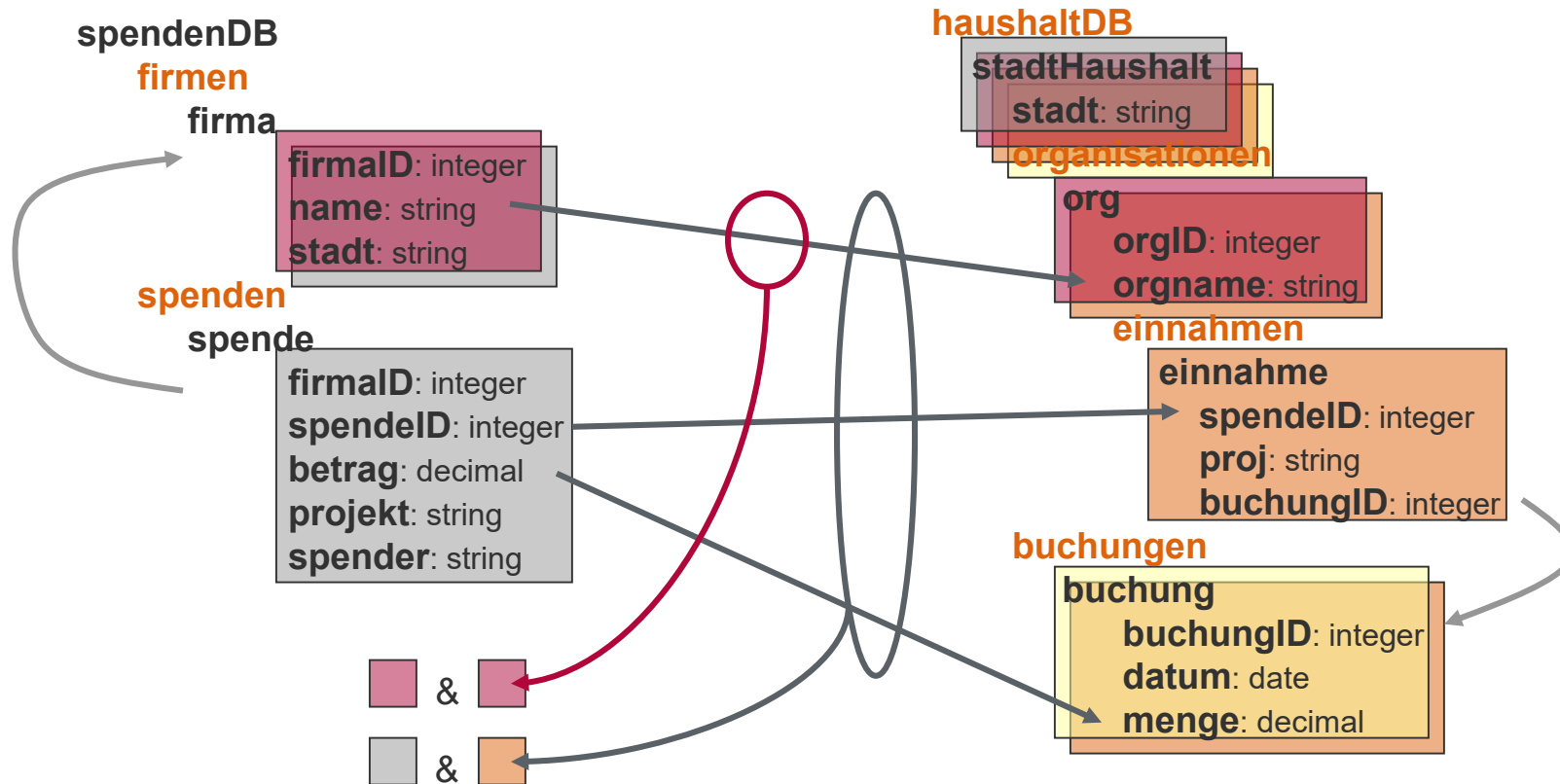


# Entdeckung von logischen Mappings



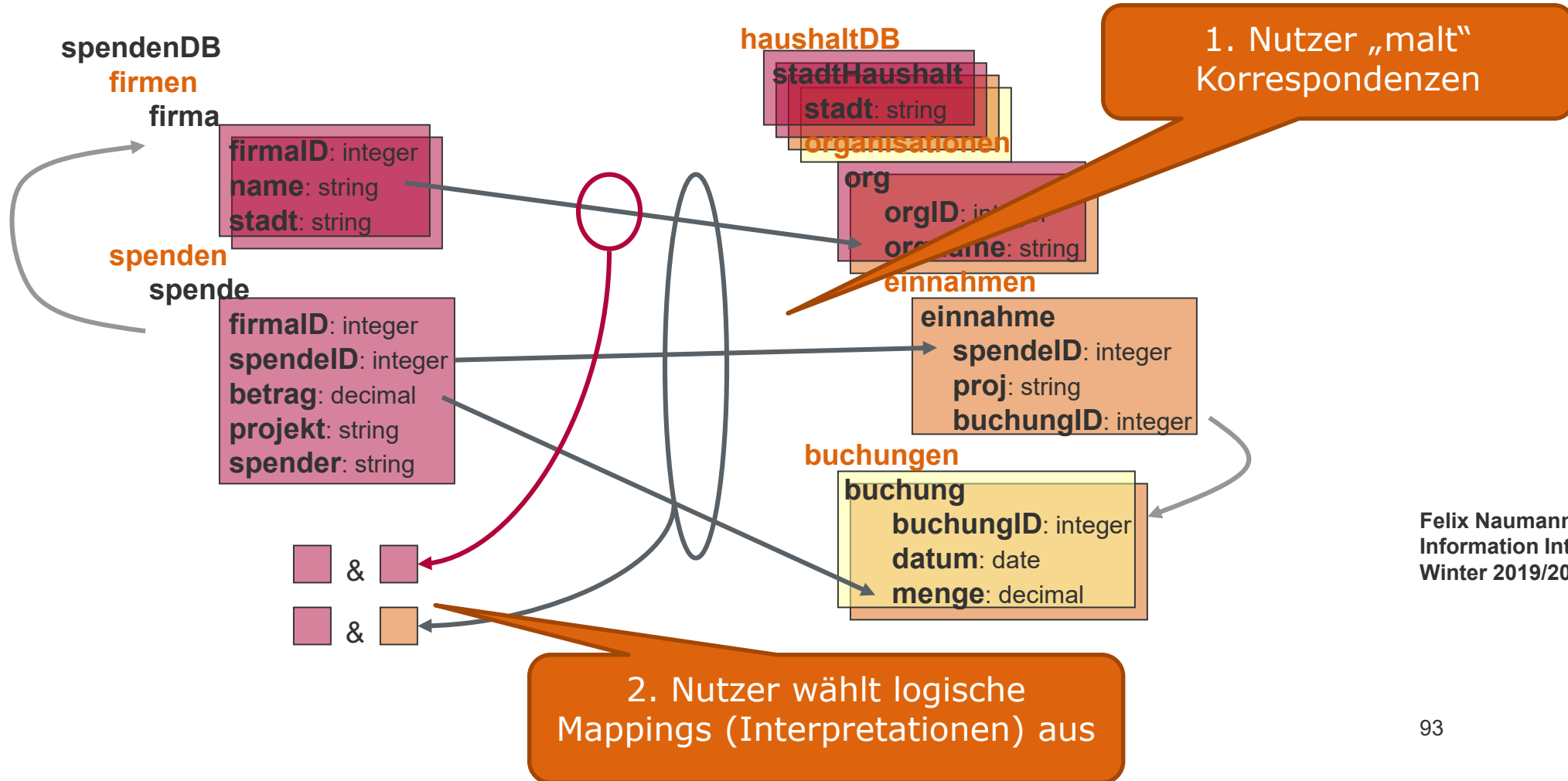
Eingehende Korrespondenzen:  
Stammen sie alle aus Assoziationen der Kombination?

# Entdeckung von logischen Mappings



Felix Naumann  
Information Integration  
Winter 2019/20

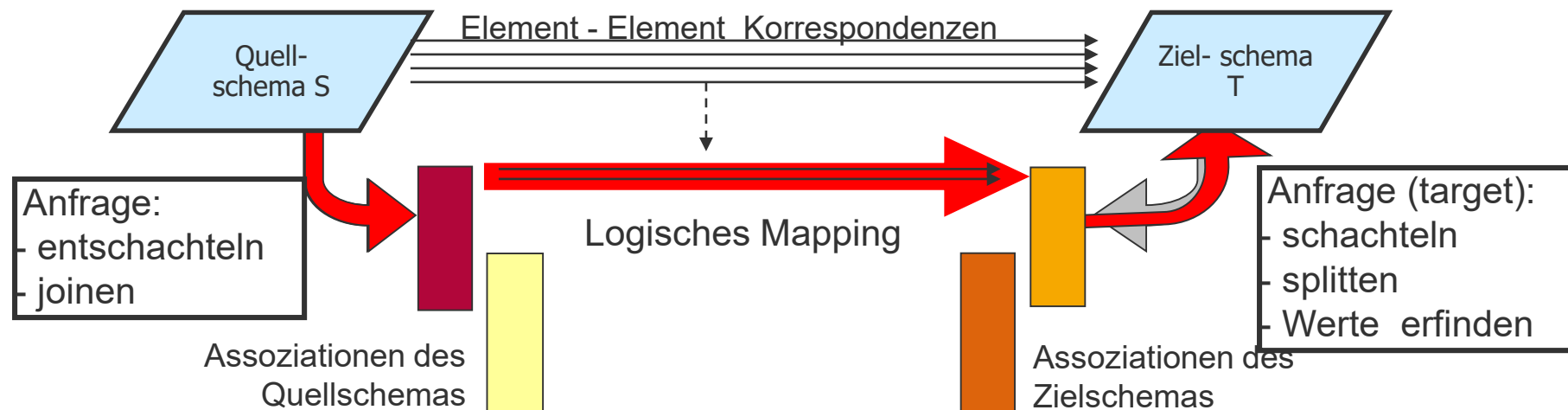
# Input des Nutzers bzw. Domänenexperten



## Erzeugung der Anfragen

### ■ Schritt 3

- Erzeuge für jedes (ausgewählte) logische Mapping eine Anfrage
  - Auswahl und Verknüpfen der entsprechenden Quelldaten
  - Generierung der entsprechenden Zieldaten



## Erzeugung der Anfragen

---

- 2 Probleme
  - Erfinden von Werten
    - NULL-Werte nicht immer ausreichend.
    - Schlüssel und passende Fremdschlüssel müssen erzeugt werden.
  - Schachtelung
    - Es soll nicht eine logische Relation (flach) materialisiert werden, sondern geschachtelte Strukturen.
    - Es muss entsprechend gruppiert werden.

## Erfinden neuer Werte (Wdh.)

- Logische Relation: buchungen
- Wert für buchungID wird nicht gefüllt
  - Entweder: Egal
  - Oder: Not-null: Dann Default-Wert, z.B. „null“
  - Oder ID: Dann eindeutigen Werte erzeugen
  - Skolemfunktion, basierend auf allen Werten des Mappings dieser logischen Relation

spendenDB  
**spenden**  
 spende

firmaID: integer  
 spendeID: integer  
 projekt: string  
 betrag: decimal  
 spender: string

haushaltDB

**einnahmen**

einnahme

spendeID: integer

proj: string

buchungID: integer

**buchungen**

buchung

buchungID: integer

datum: date

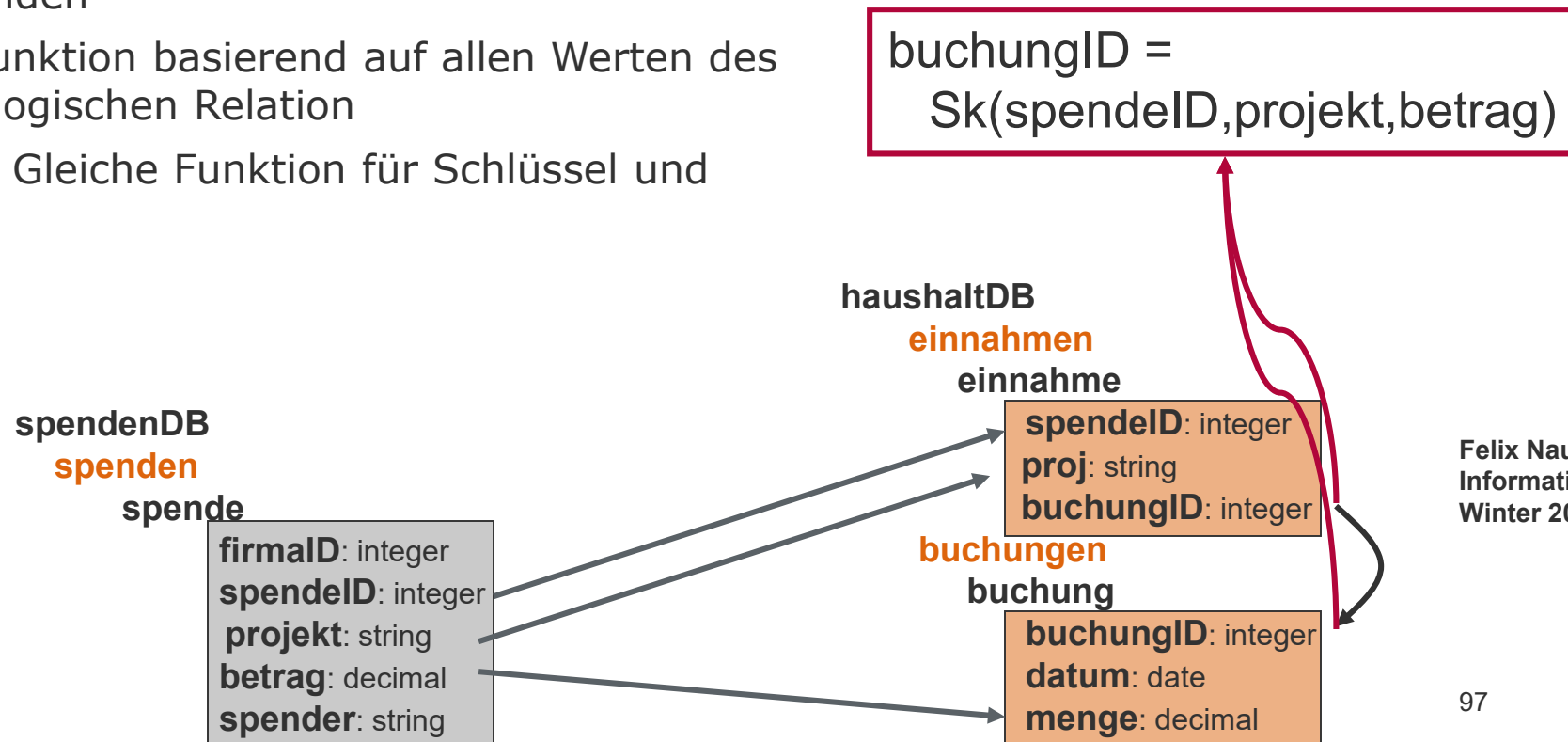
menge: decimal

**buchungID = Sk(betrag)**

Felix Naumann  
 Information Integration  
 Winter 2019/20

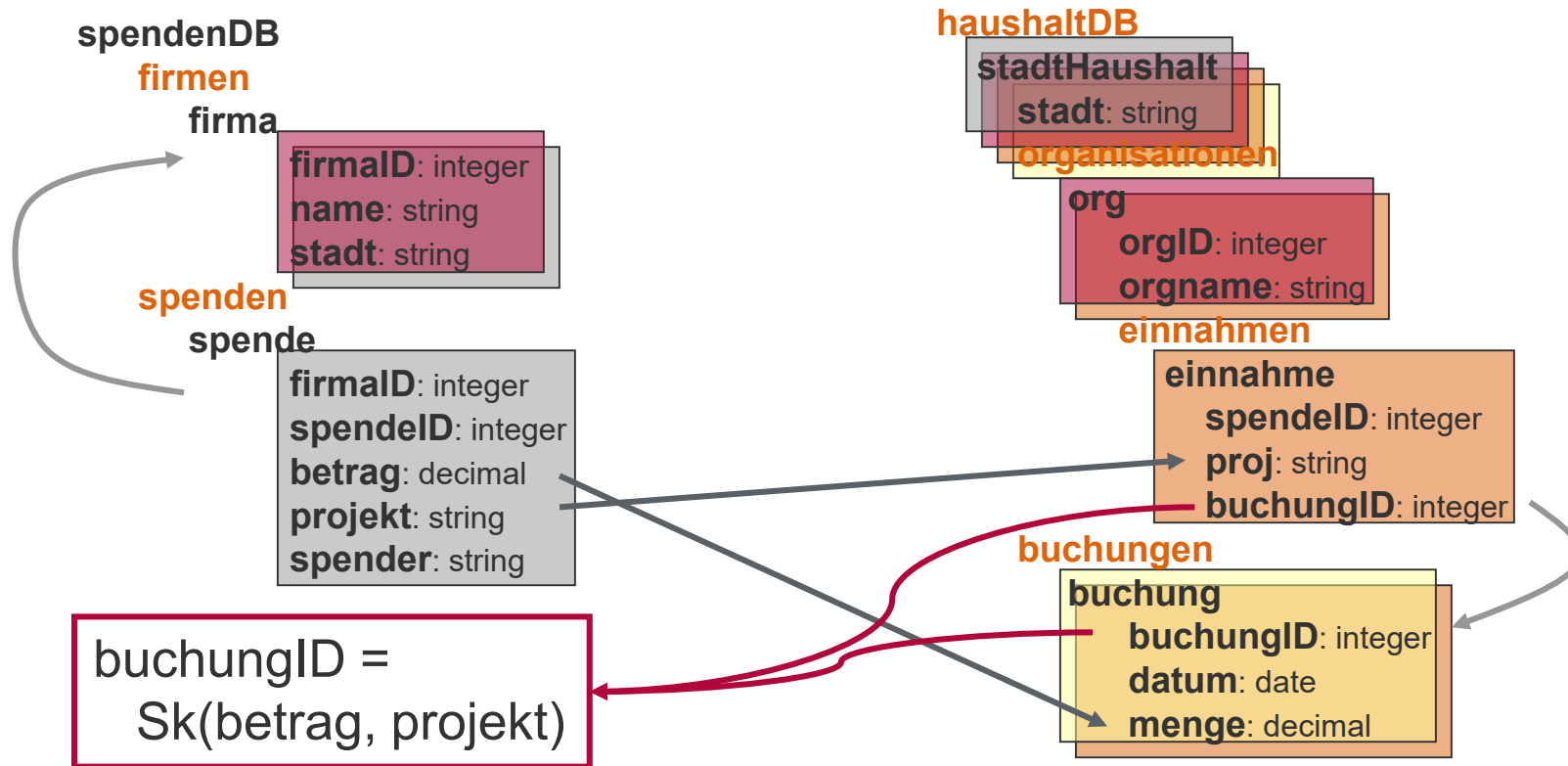
# Erfinden neuer Werte

- Logische Relation: einnahmen, buchungen
- buchungID-Attribute haben keine Korrespondenz
  - Assoziationen gingen verloren
  - Also neue ID erfinden
  - Wieder: Skolemfunktion basierend auf allen Werten des Mappings dieser logischen Relation
  - Trick wie gehabt: Gleiche Funktion für Schlüssel und Fremdschlüssel

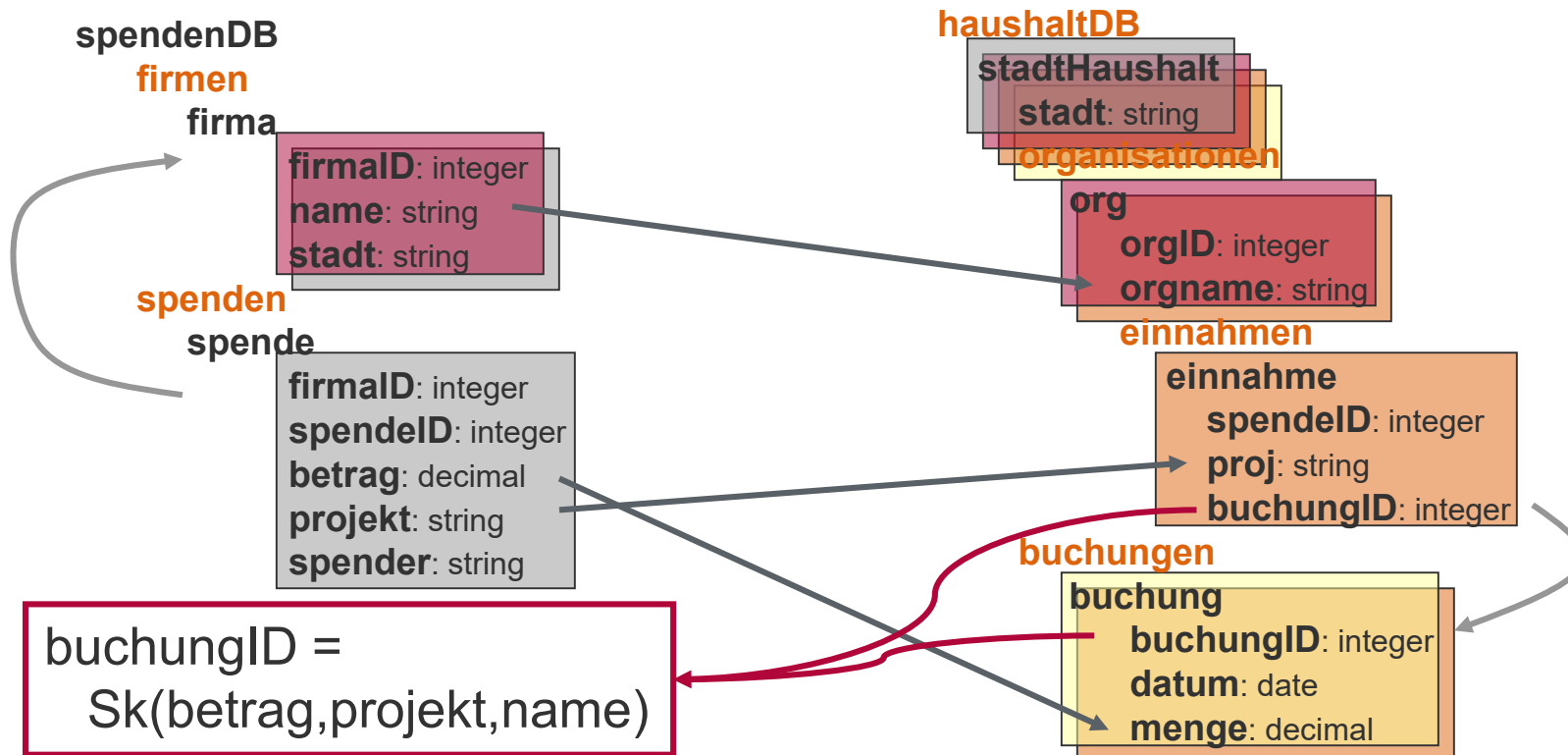




# Erfinden neuer Werte



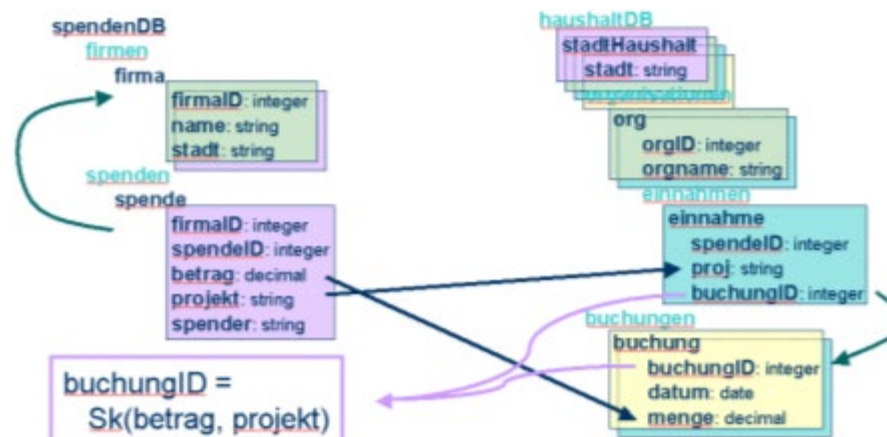
# Erfinden neuer Werte



Jetzt erst recht: Warum nicht  $\text{Sk}(\text{betrag}, \text{projekt}, \text{name}, \text{firmalD})$ ?

## Erfinden neuer Werte

- Gegenfrage: Warum nicht gleiche ALLE ungemappten Werte?
- Vier Antworten (von Lucian Popa)
  1. Prinzipiell ginge das, aber
  2. Unerklärliche „Duplikate“ könnten entstehen
    - 2 Spenden, mit gleichem Betrag und gleichem Projekt aber von unterschiedlichen Firmen
  3. firmenID könnte für Nutzer völlig uninteressant sein.
    - Mapping hätte dann die Rolle einer Projektion, in der Duplikate fehl am Platz wären
    - Sie wären sogar völlig unerklärlich für den Nutzer.
  4. Minimalität
    - Beide Varianten sind in Ordnung, aber eine ist kleiner.



# Gruppierung

- Alle Attribute erhalten Werte
- Aber:
  - Assoziationen könnten verloren gehen.
  - Neue (falsche) Assoziationen könnten erzeugt werden.
- Deshalb: Gruppierung notwendig
- Trick: Virtuelle ID Generierung mittels Skolemfunktion basierend auf allen (gemapten) Werten hierarchisch über der aktuellen Relation.
  - Im Beispiel: Jedes Firmen-Tupel erhält ID Sk(land, stadt)
  - Jedes weitere Tupel aus firmen mit gleichen Werten für stadt und land errechnet gleiche ID und wird unter gleichem Element geschachtelt.



## Erzeugung der Anfragen

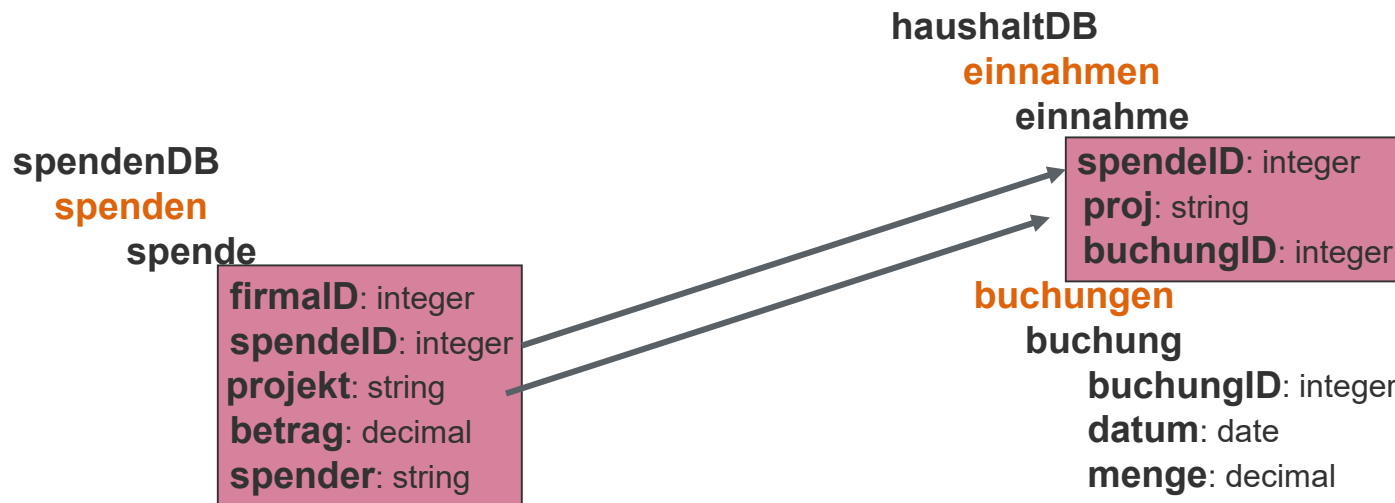
---

- Implementationsspezifisch
- Abhängig von Datenmodell
  - Relational → Relational: SQL
  - Relational → XML: SQL/XML
    - Schachtelung und tagging des Ergebnisses
  - XML → Relational: XQuery oder XSLT
    - Tags weglassen
  - XML → XML: XQuery oder XSLT
- In Clio
  - Erzeugung proprietärer Regeln
  - Dann: Übersetzung der Regeln in jeweilige Anfragesprache

# Erzeugung der Anfragen

## ■ Erzeugung proprietärer Regeln in Clio

```
□ for    x in spenden
  let    a = x.spende.spendeID, b = x.spende.projekt
  return <einnahme = <spendeID = a, proj = b, buchungID = null>>
        in einnahmen
```

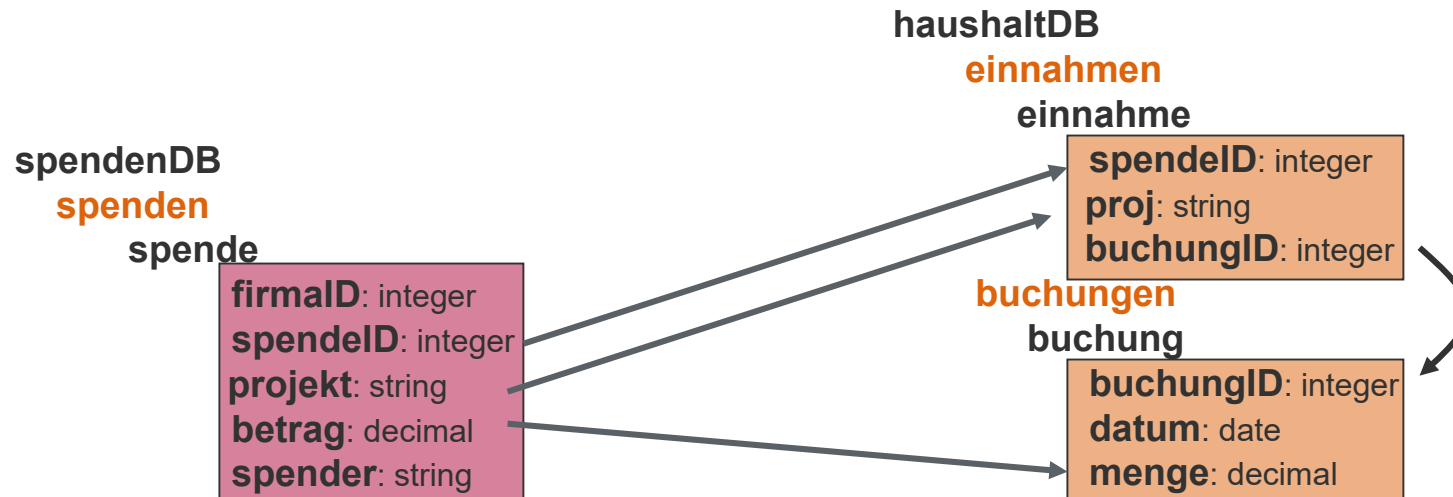


## Erzeugung der Anfragen

- Erzeugung proprietärer Regeln in Clio

```

□ for    x in spenden
  let    a = x.spende.spendeID, b = x.spende.projekt, c = x.spende.betrag
  return <einnahme = <spendeID = a, proj = b, buchungID = Sk(a,b,c)>>
          in einnahmen,
          <buchung = <buchingID = Sk(a,b,c), datum = null, menge = c>>
          in buchungen
  
```



# Erzeugung der Anfragen – XQuery

```

for x in spenden
let a = x.spende.spendeID, b = x.spende.projekt, c = x.spende.betrag
return <einnahme = <spendeID = a, proj = b, buchungID = Sk(a,b,c)>> in einnahmen,
       <buchung = <buchungID = Sk(a,b,c), datum = null, menge = c>> in buchungen

```

```

LET $doc0 := document("input XML file goes here")
RETURN <haushaltDB> {
  distinct-values (
    FOR $x0 IN $doc0/spendenDB/spende
    RETURN <einnahme>
      <spendeID> { $x0/spendeID/text() } </spendeID>
      <proj> { $x0/projekt/text() } </proj>
      <buchungID> { "Sk32(", $x0/betrag/text(), ", ", $x0/spendeID/text(), ", ", $x0/projekt/text(), ")" } </buchungID>
    </einnahme> ) }
  { distinct-values (
    FOR $x0 IN $doc0/spendenDB/spende
    RETURN <buchung>
      <buchungID> { "Sk32(", $x0/betrag/text(), ", ", $x0/spendeID/text(), ", ", $x0/projekt/text(), ")" } </buchungID>
      <menge> { $x0/betrag/text() } </menge>
    </buchung> ) }
</haushaltDB>

```



# Erzeugung der Anfragen – SQL

```

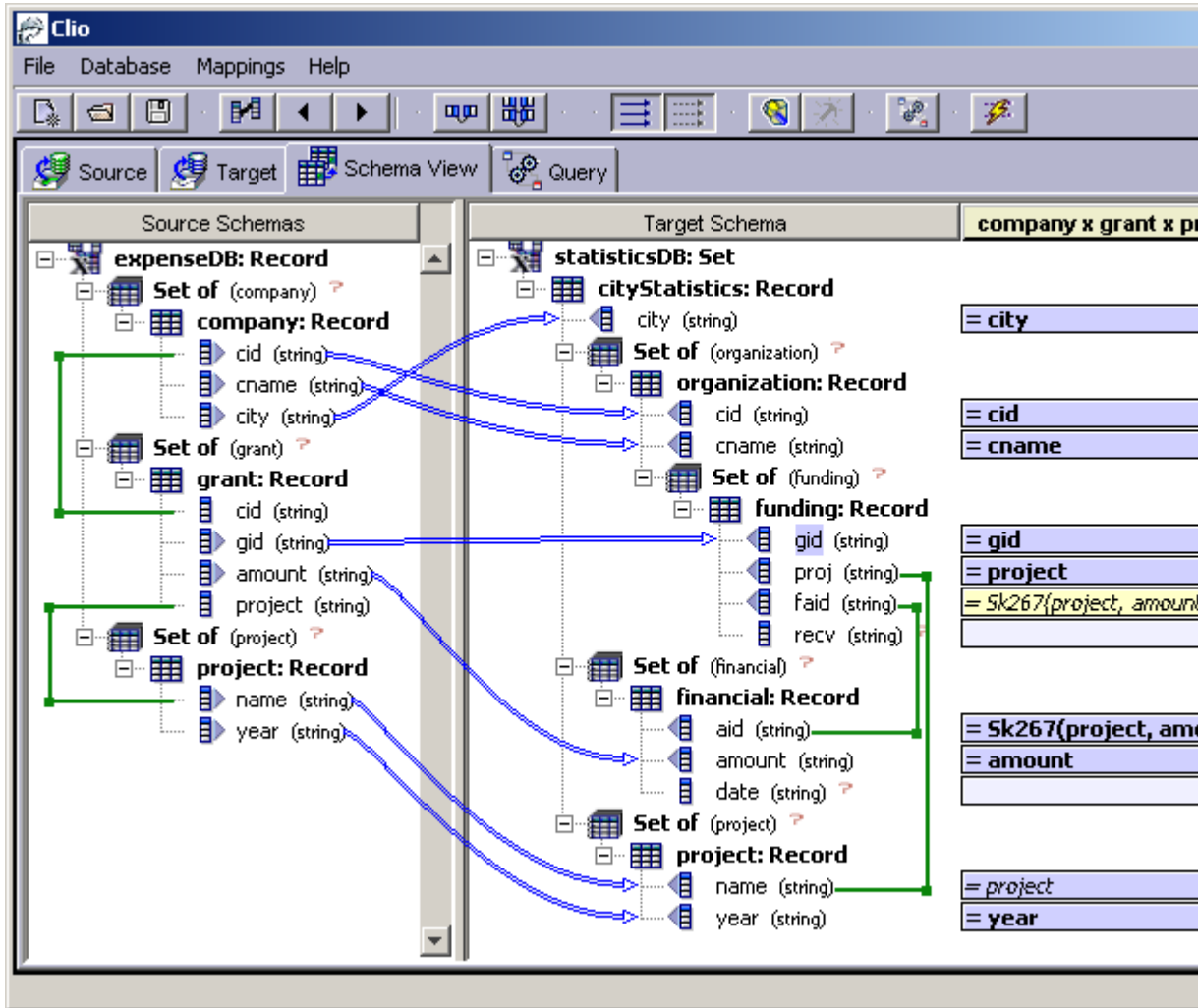
for x in spenden
let a = x.spende.spendeID, b = x.spende.projekt, c = x.spende.betrag
return <einnahme = <spendeID = a, proj = b, buchungID = Sk(a,b,c)>> in einnahmen,
       <buchung = <buchungID = Sk(a,b,c), datum = null, menge = c>> in buchungen

```

```

-- CREATE VIEW einnahme AS
SELECT
  x0.spendeID AS spendeID,
  x0.projekt AS proj,
  RTRIM('Sk32(' || CHAR(x0.betrag) || ',' || CHAR(x0.spendeID) || ',' || CHAR(x0.projekt) || ')') AS buchungID
FROM
  spendenDB.spende x0
-----
-- CREATE VIEW buchung AS
SELECT
  RTRIM('Sk32(' || CHAR(x0.betrag) || ',' || CHAR(x0.spendeID) || ',' || CHAR(x0.projekt) || ')') AS buchungID,
  x0.betrag AS menge
FROM
  spendenDB.spende x0

```



```

Clio
File Database Mappings Help
Source Target Schema View Query
XQuery XSL SQL Query

company x grant x pr

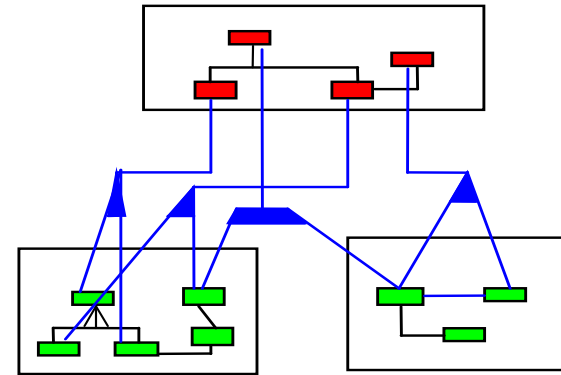
RETURN
<organization>
  <cid> $x0L1/cid/text() </cid>,
  <cname> $x2L1/cname/text() </cname>,
  distinct (
    FOR
      $x0L2 IN $doc/expenseDB/grant ,
      $x1L2 IN $doc/expenseDB/project ,
      $x2L2 IN $doc/expenseDB/company
    WHERE
      $x0L2/project/text() = $x1L2/name/text() AND
      $x2L2/cid/text() = $x0L2/cid/text() AND
      $x2L1/cname/text() = $x2L2/cname/text() AND
      $x2L1/city/text() = $x2L2/city/text() AND
      $x0L1/cid/text() = $x0L2/cid/text()
    RETURN
      <funding>
        <gid> $x0L2/gid/text() </gid>,
        <proj> $x0L2/project/text() </proj>,
        <faid> "SK267(", $x0L2/project/text(), ", ",
        </funding> )
      </organization> ),
  distinct (
    FOR
      $x0L1 IN $doc/expenseDB/grant ,
      $x1L1 IN $doc/expenseDB/project ,
      $x2L1 IN $doc/expenseDB/company
    WHERE
  
```

Preview Execute Query Copy to Clipboard

# Vorschau: Global as View / Local as View

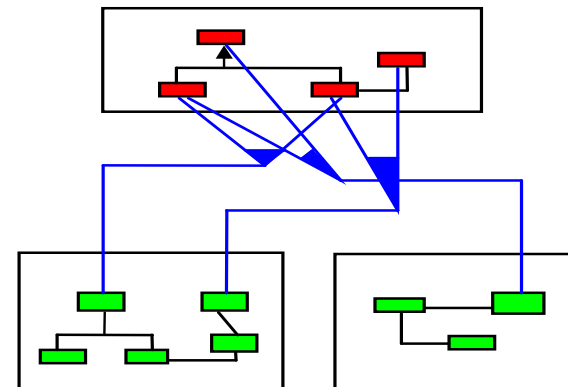
- Global as View (GaV)

- Relationen des globalen Schemas werden als Sichten auf die lokalen Schemata der Quellen ausgedrückt.



- Local as View (LaV)

- Relationen der Schemata der Quellen werden als Sichten auf das globale Schema ausgedrückt.



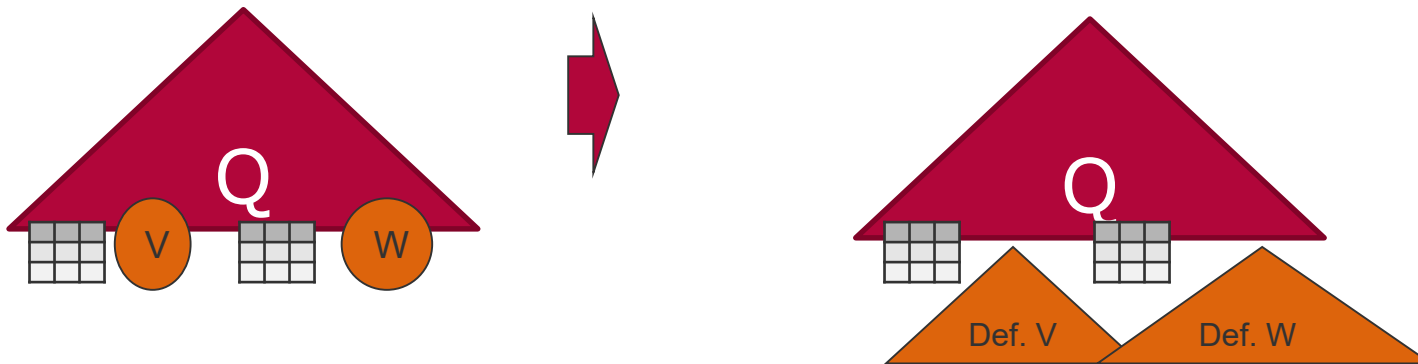
## Einschub: Sichten im relationalen Modell

---

- Relationen aus **CREATE TABLE** Ausdrücken existieren tatsächlich (materialisiert, physisch) in der Datenbank.
- Die Daten aus Sichten (*views*) existieren nur virtuell.
  - Sichten entsprechen Anfragen, denen man einen Namen gibt. Sie wirken wie physische Relationen.
- **CREATE VIEW ParamountFilme AS**  
**SELECT Titel, Jahr**  
**FROM Filme**  
**WHERE StudioName = ,Paramount`;**
- Semantik
  - Bei jeder Anfrage an die Sicht wird die SQL Anfrage der Sicht ausgeführt.
  - Die ursprüngliche Anfrage verwendet das Ergebnis als Relation.

## Anfrageplanung mit Sichten

- Baumdarstellung von Anfragen
  - Blätter repräsentieren Relationen
    - Basisrelationen
    - Sichten
  - Ersetzung der Sichten durch die Sichtdefinition
    - Als Subanfrage



## Einbettung in GaV/LaV

---

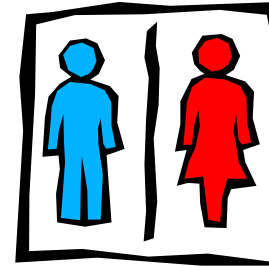
- Ergebnis des Schema Mapping Prozess sind Anfragen
  - Eine oder mehrere Anfragen pro Assoziation
- Relationales Modell:
  - Jede Ziel-Relation entspricht einem primären Pfad
  - Verwerfe Fremdschlüssel
  - Jede Ziel-Relation entspricht einer Assoziation
  - Jede Anfrage (oder Vereinigung von Anfrage) liefert Daten für eine Relation, also wie GaV.
- Schema Mapping leistet aber mehr!
  - Erzeugung von Daten für mehrere Zielrelationen zugleich
  - Unter Berücksichtigung von Integritätsbedingungen in Quell- und Zielschema
  - „Kombination von LaV und GaV“: GLaV

# Schematische Heterogenität

---

- Struktur
    - Modellierung
      - Relation vs. Attribut
      - Attribut vs. Wert
      - Relation vs. Wert
    - Benennung
      - Relationen
      - Attribute
    - Normalisiert vs. Denormalisiert
    - Geschachtelt vs. Fremdschlüssel
    - Geschachtelt vs. Flach
- Higher-order Mappings
- bisher

# High-order Mappings



Männer ( Id, Vorname, Nachname)  
Frauen ( Id, Vorname, Nachname)

Relation vs. Attribut

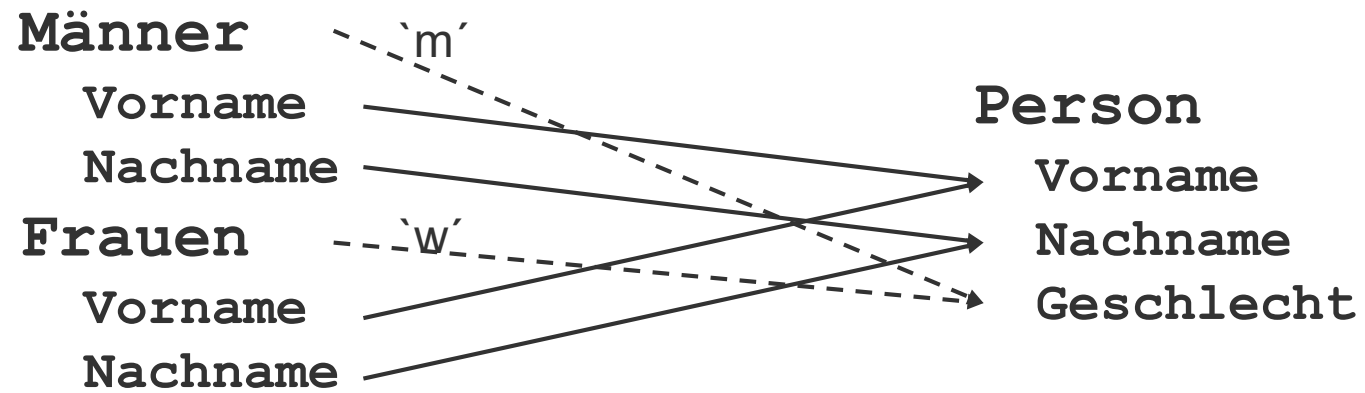
Person ( Id, Vorname, Nachname, männlich, weiblich)

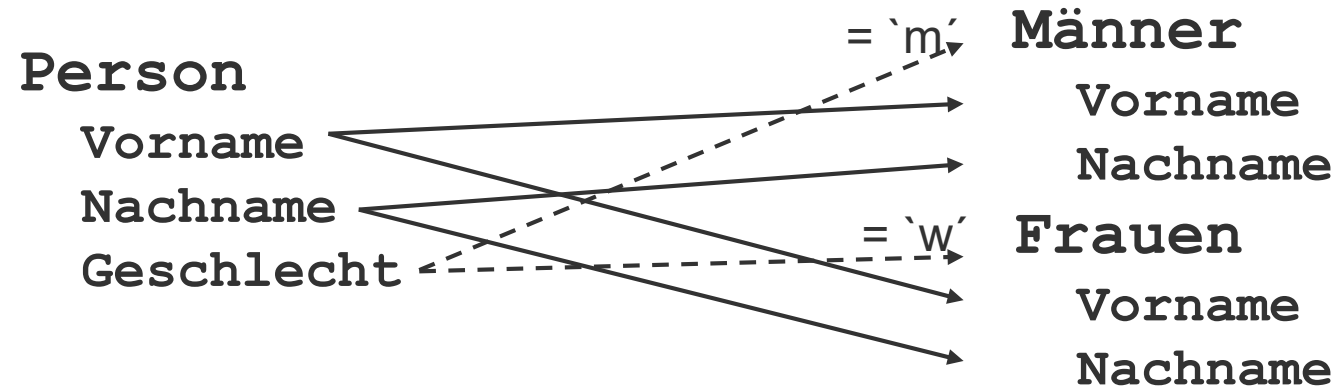
Relation vs. Wert

Person ( Id, Vorname, Nachname, Geschlecht)

Attribut vs. Wert

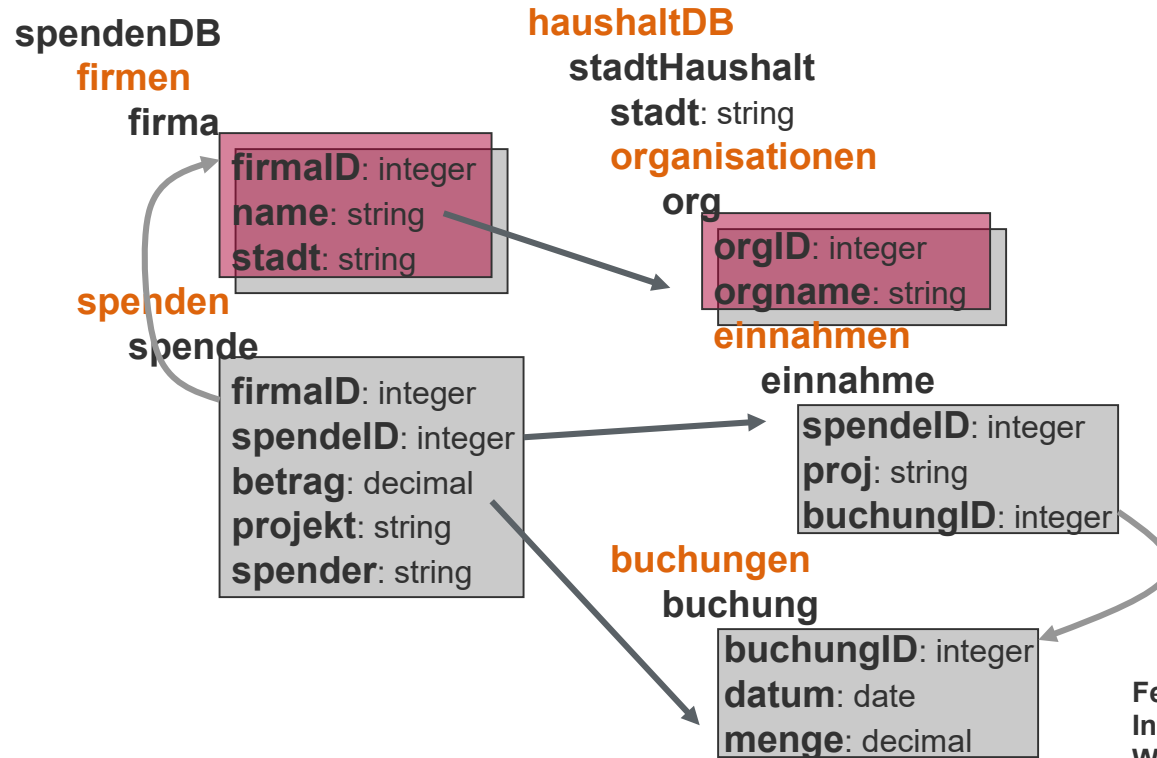






# Zusammenfassung

- Drei Schritte
  1. Entdeckung von intra-Schema Assoziationen
  2. Entdeckung von inter-Schema logischen Mappings
  3. Anfrage-erzeugung



Felix Naumann  
Information Integration  
Winter 2019/20

## Überblick

1. Motivation
2. Schema Mapping
3. Schema Matching
4. Mapping Interpretation
- 5. Mapping Werkzeuge**



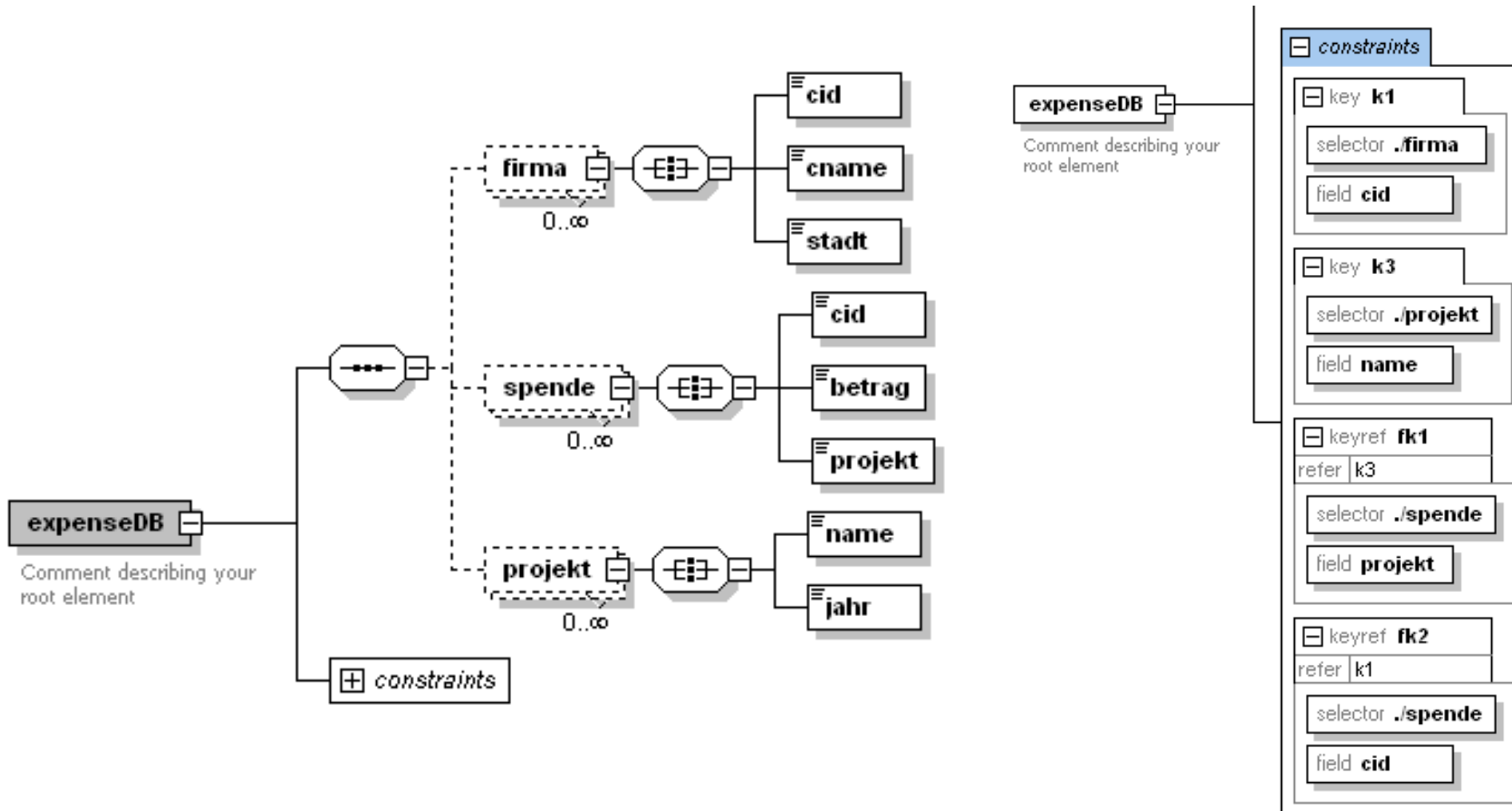
Felix Naumann  
Information Integration  
Winter 2019/20

## Vergleich dreier Tools

---

- Altovas MapForce
- IBMs Clio
- [IBMs WSAD]
- Szenario
  - expenseDB -> statisticsDB
  - Definition des Mappings
  - Generierung der Transformation
  - Vergleich der XML Ergebnisse
- XML Screenshots aus XMLSpy
  - [www.xmlspy.com](http://www.xmlspy.com)

# Szenario - ExpenseDB

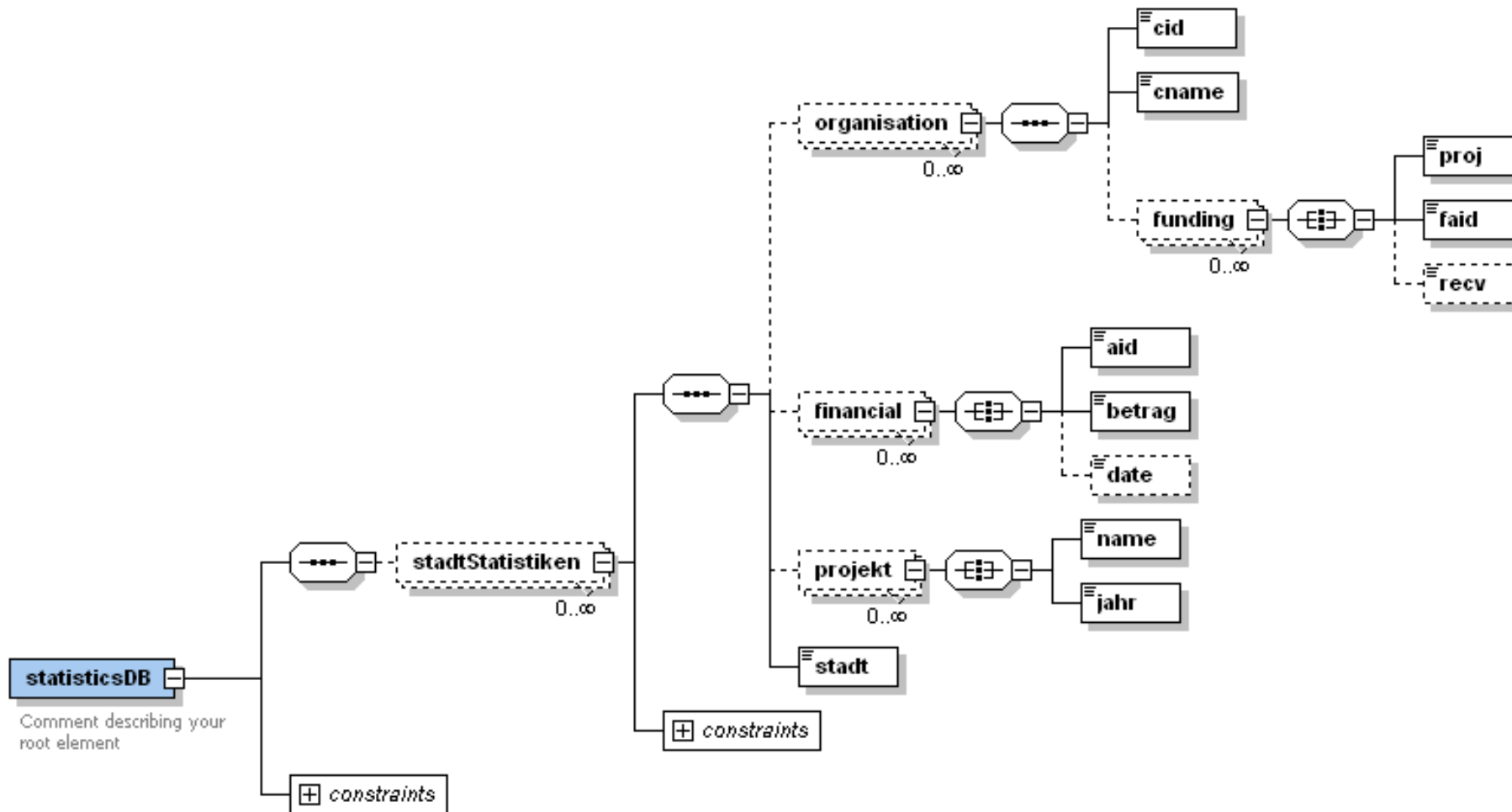


Felix Naumann  
Information Integration  
Winter 2019/20

# Szenario - ExpenseDB

expenseDB			
<b>xmlns:xsi</b>	http://www.w3.org/2001/XMLSchema-instance		
<b>xsi:nillamespac...</b>	workspace\Clio\schemas\expenseDB.xsd		
▲ <b>firma</b> (5)			
	<b>cid</b>	<b>cname</b>	<b>stadt</b>
1	GG	Garlic	Gilroy
2	CMPQ	Compaq	San Jose
3	NA	Network Associates	San Jose
4	IBM	International Business Machines	San Jose
5	MSFT	Microsoft	Redmond
▲ <b>spende</b> (6)			
	<b>cid</b>	<b>betrag</b>	<b>projekt</b>
1	MSFT	12350	SQLServer
2	IBM	3500	Clio
3	IBM	25000	DB2
4	NA	5000	FastNetwork
5	MSFT	500	Leo
6	IBM	10000	Clio
▲ <b>projekt</b> (5)			
	<b>name</b>	<b>jahr</b>	
1	SQLServer	1997	
2	FastNetwork	1996	
3	DB2	1987	
4	Clio	1999	
5	Leo	2000	

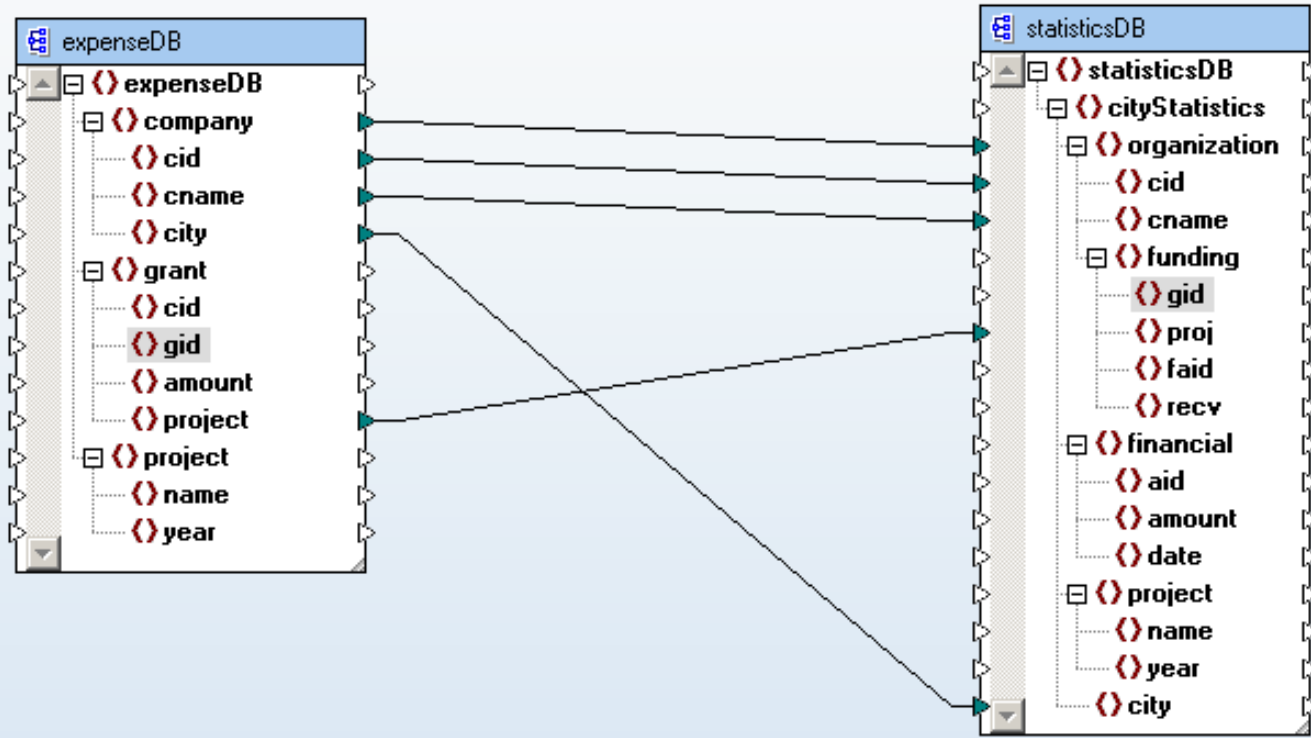
# Szenario StatisticsDB



Felix Naumann  
Information Integration  
Winter 2019/20



# Altovas MapForce



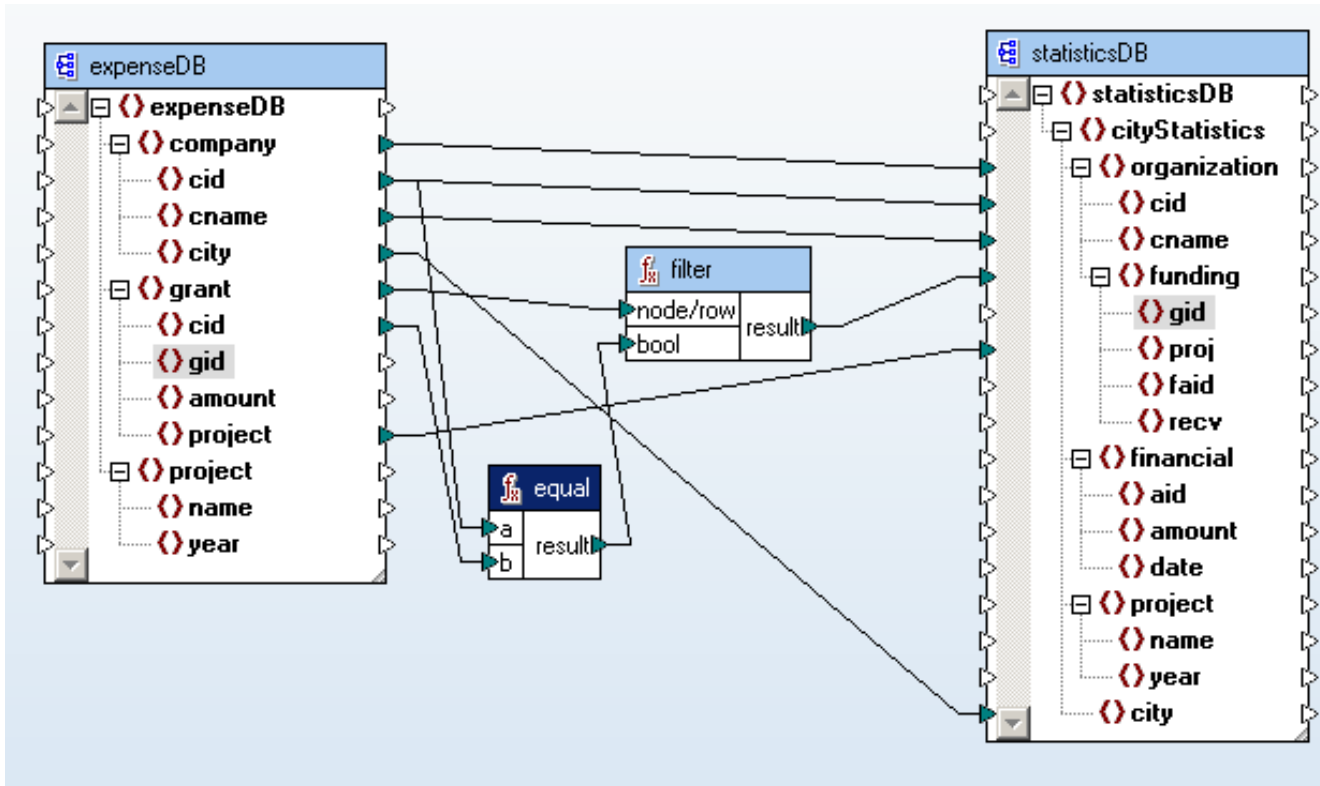
```

<?xml version="1.0" encoding="UTF-8"?>
<statisticsDB xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <cityStatistics>
    <organization>
      <cid>GG</cid>
      <cname>Garlic</cname>
      <funding>
        <proj>SQLServer</proj>
        <proj>Clio</proj>
        <proj>DB2</proj>
        <proj>FastNetwork</proj>
        <proj>Leo</proj>
        <proj>Clio</proj>
      </funding>
    </organization>
    <organization>
      <cid>CMPQ</cid>
      <cname>Compaq</cname>
      <funding>
        <proj>SQLServer</proj>
        <proj>Clio</proj>
        <proj>DB2</proj>
        <proj>FastNetwork</proj>
        <proj>Leo</proj>
        <proj>Clio</proj>
      </funding>
    </organization>
    <organization>
      <cid>Gilroy</cid>
      <cid>San Jose</cid>
      <cid>San Jose</cid>
      <cid>San Jose</cid>
      <cid>Redmond</cid>
    </organization>
  </cityStatistics>
  <organization>
    <cid>
    <cname>
    <funding>
      <gid>
      <proj>
      <faid>
      <recv>
    </funding>
    <financial>
      <aid>
      <amount>
      <date>
    </financial>
    <project>
      <name>
      <year>
      <city>
    </project>
  </organization>

```

Felix Naumann  
Information Integration  
Winter 2019/20

# Altovas MapForce



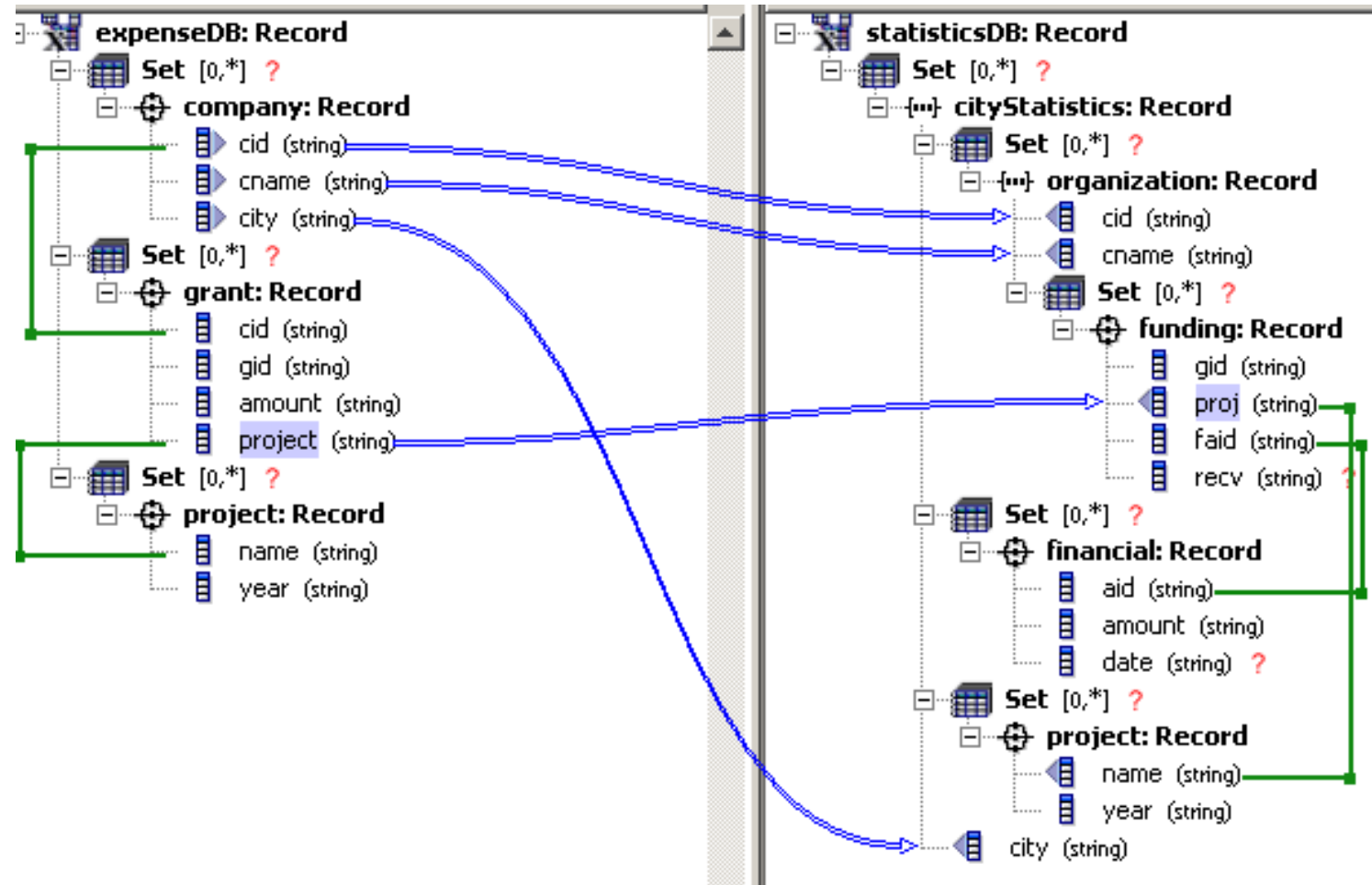
```

<organization>
  <cid>GG</cid>
  <cname>Garlic</cname>
</organization>
<organization>
<organization>
<organization>
  <cid>IBM</cid>
  <cname>International Business Machines</cname>
  <funding>
    <proj>Clio</proj>
  </funding>
  <funding>
    <proj>DB2</proj>
  </funding>
  <funding>
    <proj>Clio</proj>
  </funding>
</organization>
<organization>
  <cid>MSFT</cid>
  <cname>Microsoft</cname>
  <funding>
    <proj>SQLServer</proj>
  </funding>
  <funding>
    <proj>Leo</proj>
  </funding>
</organization>
<city>Gilroy</city>
<city>San Jose</city>
<city>San Jose</city>
<city>San Jose</city>
<city>Redmond</city>
</cityStatistics>
  
```

## Altovas MapForce

---

- Zusammenfassung
  - Informationen gehen nicht verloren
  - Joins
    - werden nicht erkannt
    - können manuell in GUI erstellt werden
  - Gruppierung wird nicht erkannt



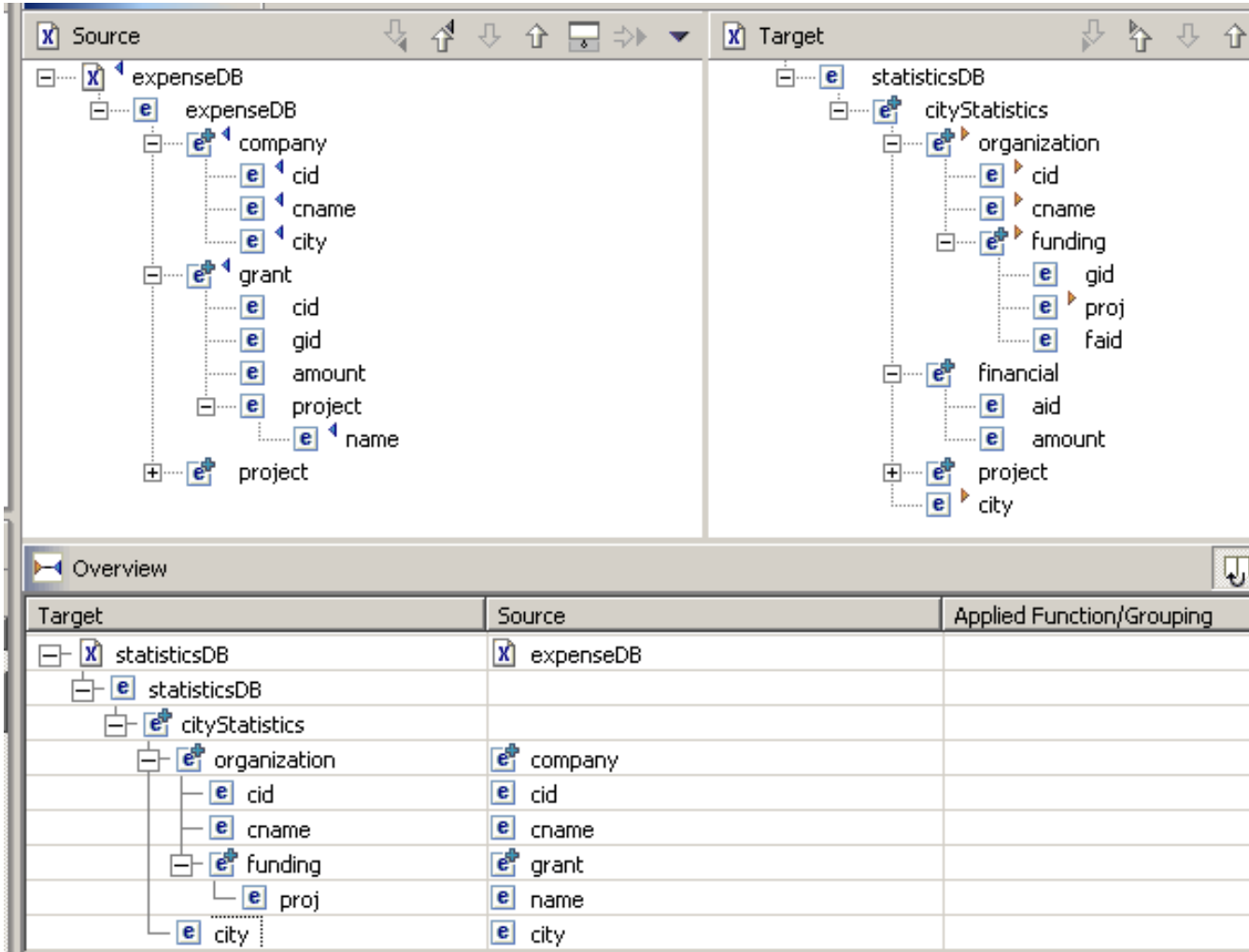
```
<?xml version="1.0" encoding="UTF-8" ?>
- <quip:result xmlns:quip="http://namespaces.softwareag.com/tamino/quip/">
- <statisticsDB>
+ <cityStatistics>
- <cityStatistics>
- <organization>
  <cid>IBM</cid>
  <cname>International Business Machines</cname>
- <funding>
  <gid>Sk72(Clio, International Business Machines, San Jose, IBM)</gid>
  <proj>Clio</proj>
  <faid>Sk67(Clio, International Business Machines, San Jose, IBM)</faid>
</funding>
- <funding>
  <gid>Sk72(DB2, International Business Machines, San Jose, IBM)</gid>
  <proj>DB2</proj>
  <faid>Sk67(DB2, International Business Machines, San Jose, IBM)</faid>
</funding>
</organization>
- <organization>
  <cid>NA</cid>
  <cname>Network Associates</cname>
- <funding>
  <gid>Sk72(FastNetwork, Network Associates, San Jose, NA)</gid>
  <proj>FastNetwork</proj>
  <faid>Sk67(FastNetwork, Network Associates, San Jose, NA)</faid>
</funding>
</organization>
+ <financial>
+ <financial>
+ <financial>
+ <project>
+ <project>
+ <project>
  <city>San Jose</city>
</cityStatistics>
</statisticsDB>
</quip:result>
```

## IBMs Clio

---

- Zusammenfassung
  - Joins werden erkannt
    - Aber nicht erzwungen: Interpretation
  - Gruppierung wird erkannt

# IBMs WSAD



The screenshot shows the IBM WSAD interface with two tree views: Source and Target. The Source tree shows a hierarchy starting with 'expenseDB', containing 'expenseDB', 'company' (with 'cid', 'cname', 'city'), 'grant' (with 'cid', 'gid', 'amount', 'project'), and 'project'. The Target tree shows a hierarchy starting with 'statisticsDB', containing 'cityStatistics', 'organization' (with 'cid', 'cname', 'funding' (with 'gid', 'proj', 'faid')), 'financial' (with 'aid', 'amount'), and 'project' (with 'city'). Below the trees is an 'Overview' table:

Target	Source	Applied Function/Grouping
statisticsDB	expenseDB	
statisticsDB		
cityStatistics		
organization	company	
cid	cid	
cname	cname	
funding	grant	
proj	name	
city	city	

```

<?xml version="1.0" encoding="UTF-8" ?>
- <statisticsDB>
- <cityStatistics>
  - <organization>
    <cid>GG</cid>
    <cname>Garlic</cname>
  + <funding>
  + <funding>
  + <funding>
  + <funding>
  + <funding>
  + <funding>
  - <funding>
    <gid />
    <proj>SQLServer</proj>
    <faid />
  </funding>
- <funding>
  <gid />
  <proj>FastNetwork</proj>
  </funding>
+ <funding>
+ <funding>
+ <funding>
</organization>
+ <organization>
+ <organization>
+ <organization>
+ <organization>
+ <financial>
- <project>
  <name />
  <year />
</project>
  <city>Gilroy</city>
</cityStatistics>
</statisticsDB>
  
```

## IBMs WSAD

---

- Zusammenfassung
  - Mögliche Joins werden nicht erkannt
  - Gruppierung wird nicht erkannt
  - Informationen werden verworfen
    - Nur erste Stadt
  - Information wird „erfunden“
    - Jede Organization erhält jedes Projekt



# Zusammenfassung

## ■ IBMs WSAD

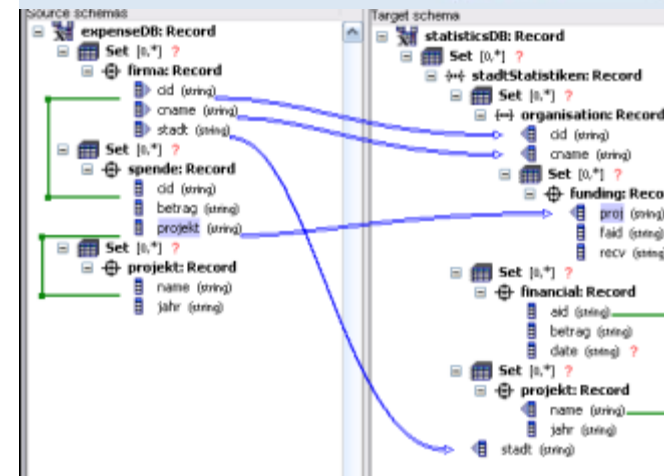
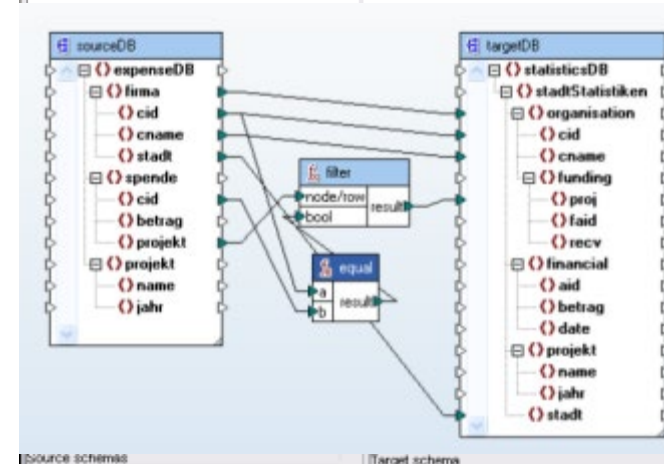
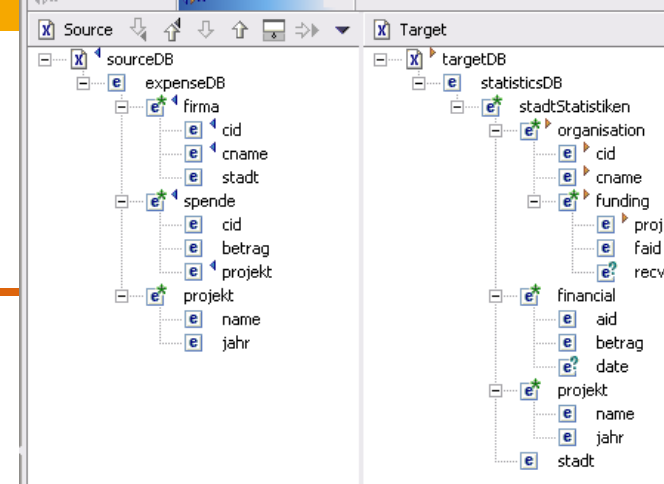
- Mögliche Joins werden nicht erkannt.
- Gruppierung wird nicht erkannt.
- Daten werden verworfen.
- Assoziationen zw. Daten werden „erfunden“.

## ■ Altovas MapForce

- Daten gehen nicht verloren.
- Joins werden nicht erkannt, aber können manuell in GUI erstellt werden.
- Gruppierung wird nicht erkannt.

## ■ IBMs Clío

- Joins werden erkannt, aber nicht erzwungen: Interpretationen möglich
- Gruppierung wird erkannt.



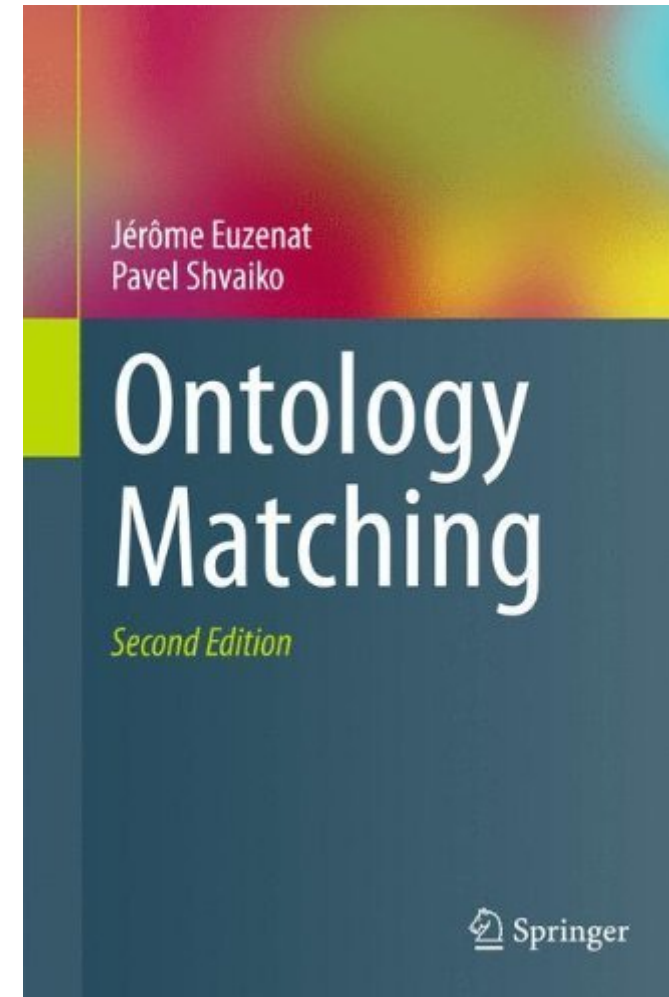
## Literatur – Schema Mapping

---

- [FHP+02] Ron Fagin, Mauricio Hernandez, Lucian Popa, Renee Miller, and Yannis Velegarakis, Translating Web Data, VLDB 2002, Hong Kong, China.
- Laura M. Haas, [Mauricio A. Hernández](#), [Howard Ho](#), [Lucian Popa](#), [Mary Roth](#): Clio grows up: from research prototype to industrial tool. [SIGMOD Conference 2005](#): 805-810
- Zu der Problematik, ob Duplikate Teil des Outputs sein sollten:
  - [Ronald Fagin](#), [Phokion G. Kolaitis](#), [Renée J. Miller](#), Lucian Popa: Data Exchange: Semantics and Query Answering. [ICDT 2003](#): 207-224

## Literatur – Schema Matching

- Artikel mit der Klassifikation:
  - [RB01] Erhard Rahm and Philip Bernstein, A survey of approaches to automatic schema matching, VLDB Journal 10(4), 2001.
- Online: <http://www.ontologymatching.org/> plus Buch
- Spezielle Algorithmen
  - [MGMR02] Sergey Melnik, [Hector Garcia-Molina](#), [Erhard Rahm](#): Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. [ICDE 2002](#): 117-128
  - [BN05] [Schema Matching using Duplicates](#) Alexander Bilke and Felix Naumann: *Proceedings of the International Conference on Data Engineering (ICDE 05)* Tokyo, Japan.
  - [Robin Dhamankar](#), [Yoonkyong Lee](#), AnHai Doan, [Alon Y. Halevy](#), [Pedro M. Domingos](#): iMAP: Discovering Complex Mappings between Database Schemas. [SIGMOD Conference 2004](#): 383-394
  - uvam.



Felix Naumann  
Information Integration  
Winter 2019/20