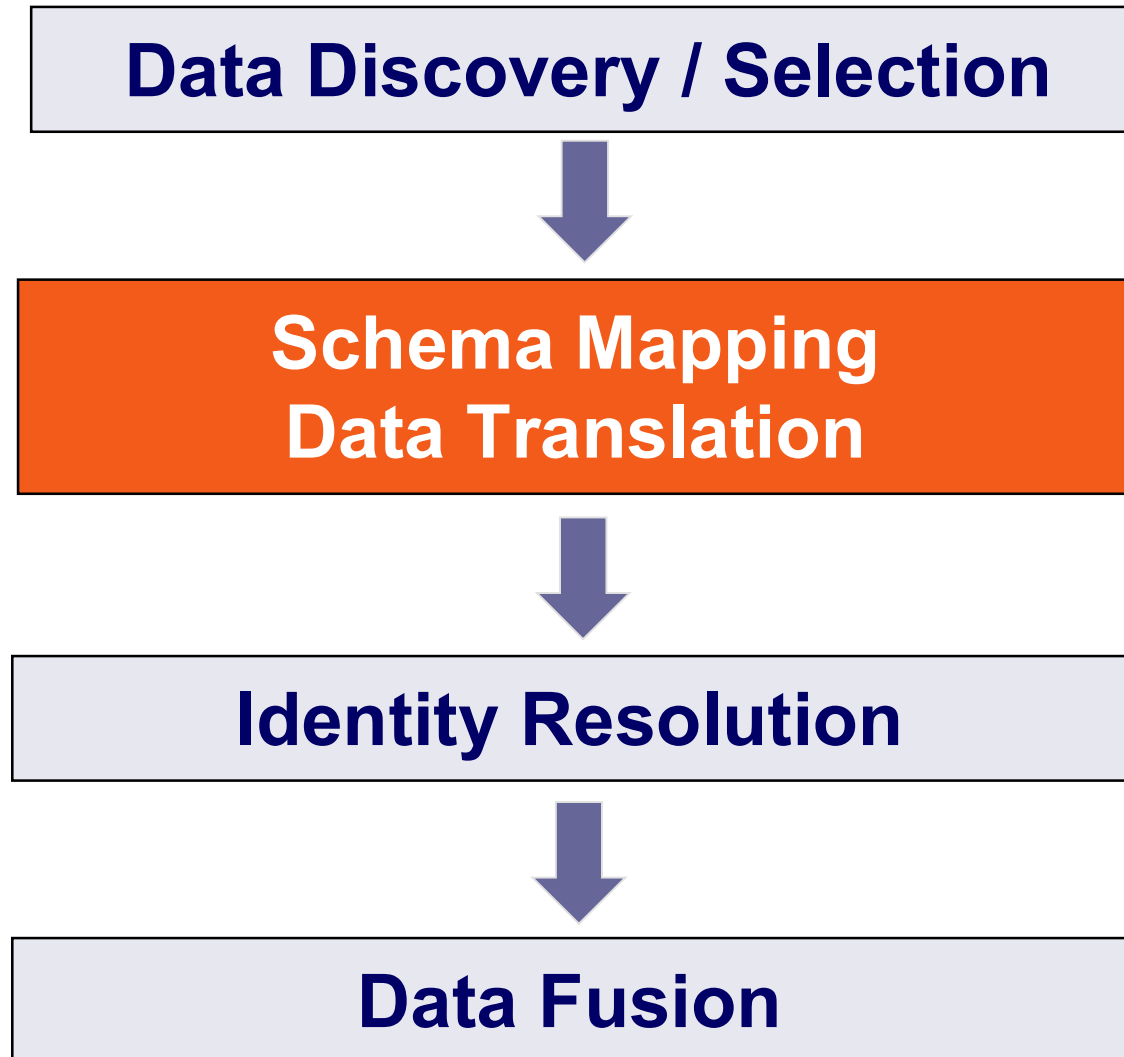


Web Data Integration

Schema Mapping and Data Translation



The Data Integration Process

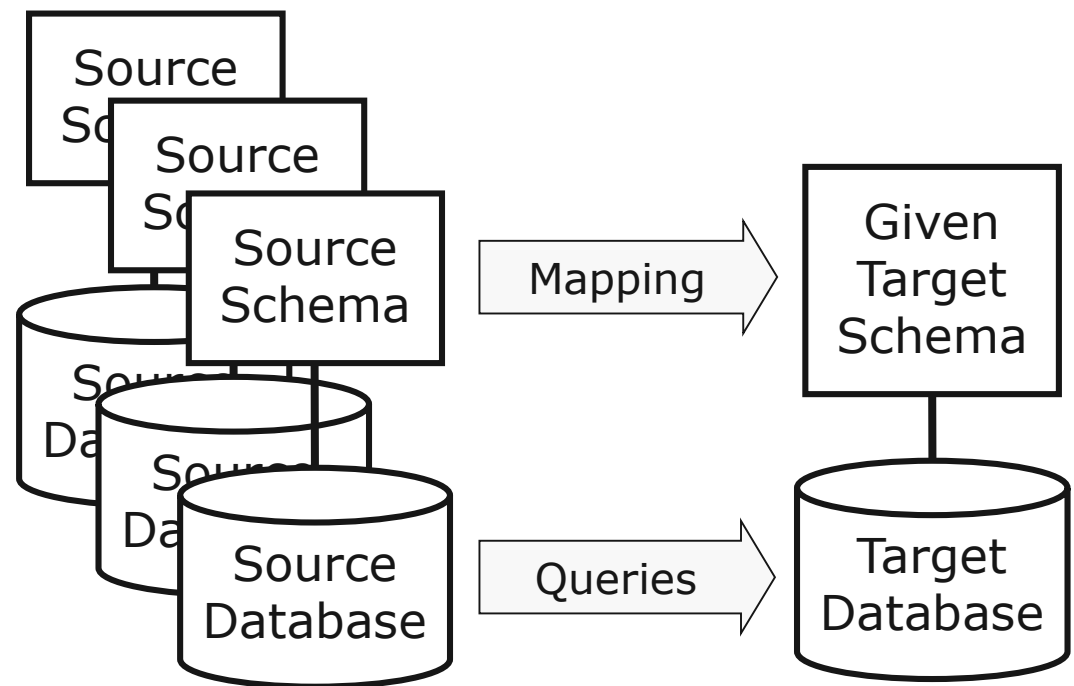


1. Two Basic Integration Situations
2. Types of Correspondences
3. Schema Integration
4. Data Translation
5. Schema Matching
6. Schema Heterogeneity on the Web

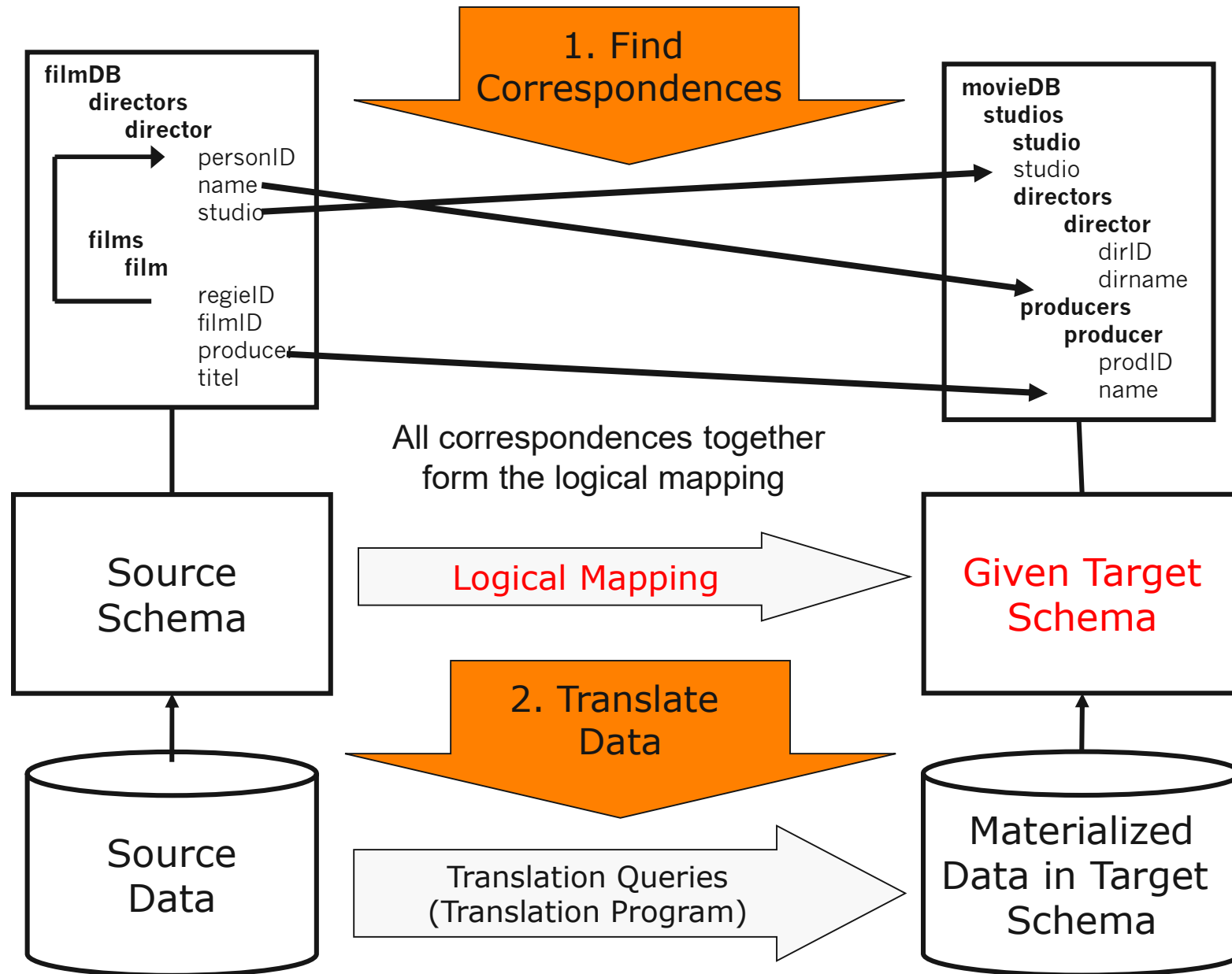
Basic Integration Situation 1: Schema Mapping

Goal: Translate data from a set of source schemata into a **given target schema.**

- Top-down integration situation
- Triggered by concrete information need (= target schema)



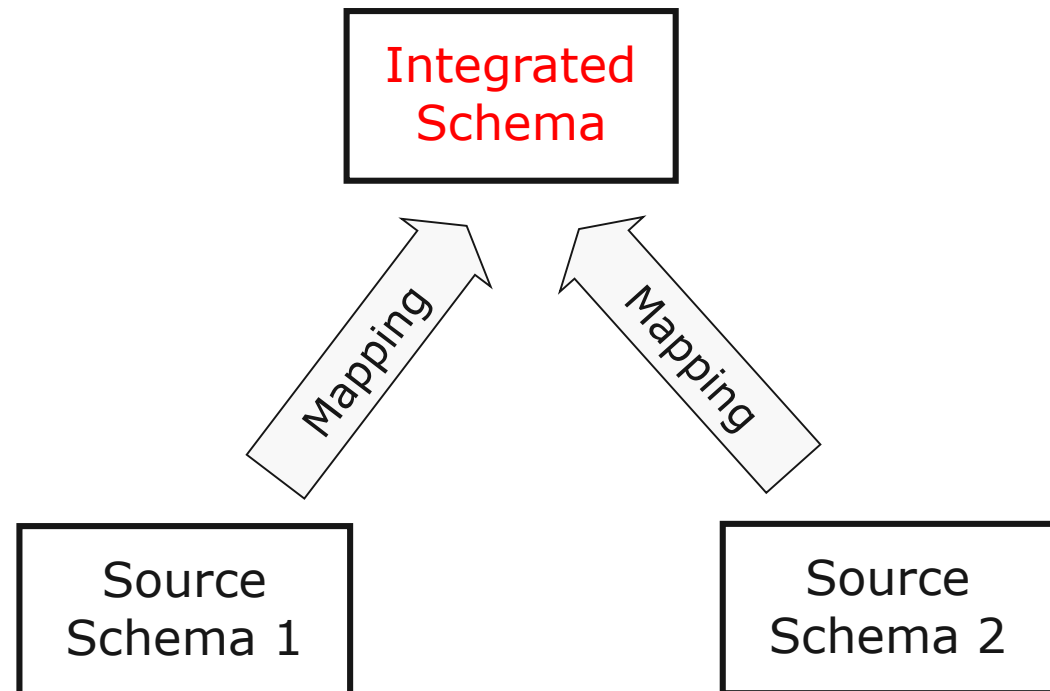
The Schema Mapping and Data Translation Process



Basic Integration Situation 2: Schema Integration

Goal: Create a new **integrated schema** that can represent **all data** from a given set of source schemata.

- Bottom-up integration situation
- Triggered by the goal to fulfill different information needs based on data from all sources.



2. Correspondences

A correspondence relates a set of elements in a schema S to a set of elements in schema T.

- **Mapping** = Set of all correspondences that relate S and T
- Correspondences are **easier to specify** than transformation queries
 - domain expert does not need technical knowledge about query language
 - specification can be supported by user interfaces (mapping editors)
 - step-by-step process with separate local decisions
- Correspondences can be annotated with **transformation functions**
 - normalize units of measurement (€ to US\$, cm and km to meters)
 - calculate or aggregate values (salary * 12 = yearly salary)
 - translate values using a translation table (area code to city name)
 - perform information extraction (product title to set of product attributes)

Types of Correspondences

– One-to-One Correspondences

- `Movie.title` $\rightarrow$ `Item.name`
- `Product.rating` $\rightarrow$ `Item.classification`
- `Movie` $\equiv$ `Film` (equivalence: Same semantic intention)
- `Athlete` $\subseteq$ `Person` (inclusion: All athletes are also persons)

– One-to-Many Correspondences

- `Person.Name` $\rightarrow$ `split()` $\rightarrow$ `FirstName` (Token 1)
 $\rightarrow$ `Surname` (Token 2)

– Many-to-One Correspondences

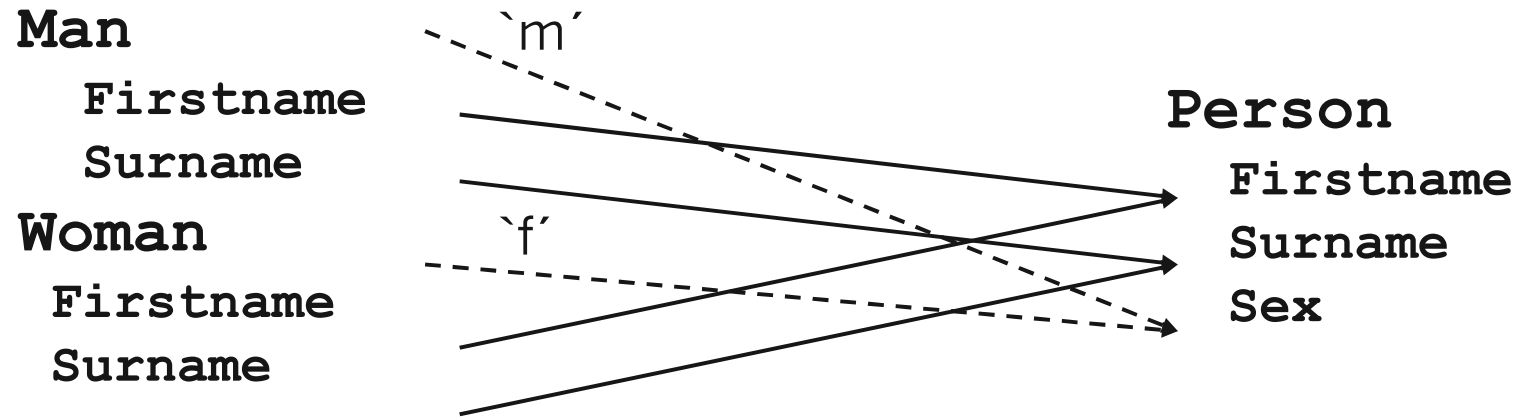
- `Product.basePrice` * (1 + `Location.taxRate`) $\rightarrow$ `Item.price`

– Higher-Order Correspondences

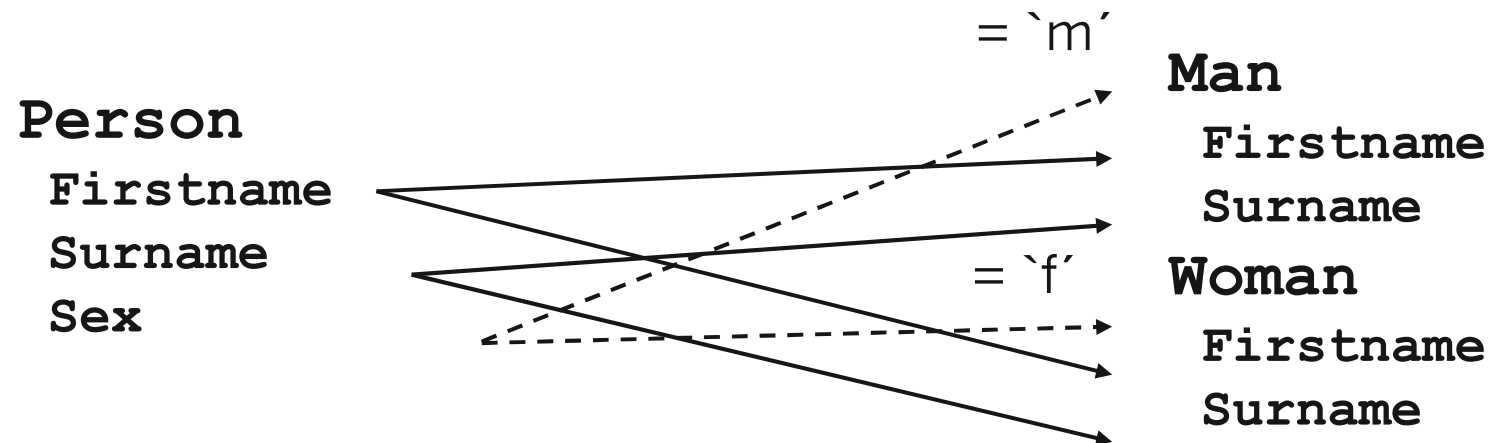
- relate different types of data model elements
- for example: Relations (classes) and attributes, see next slide

Examples of Higher-Order Correspondences

Relation-to-Value Correspondences



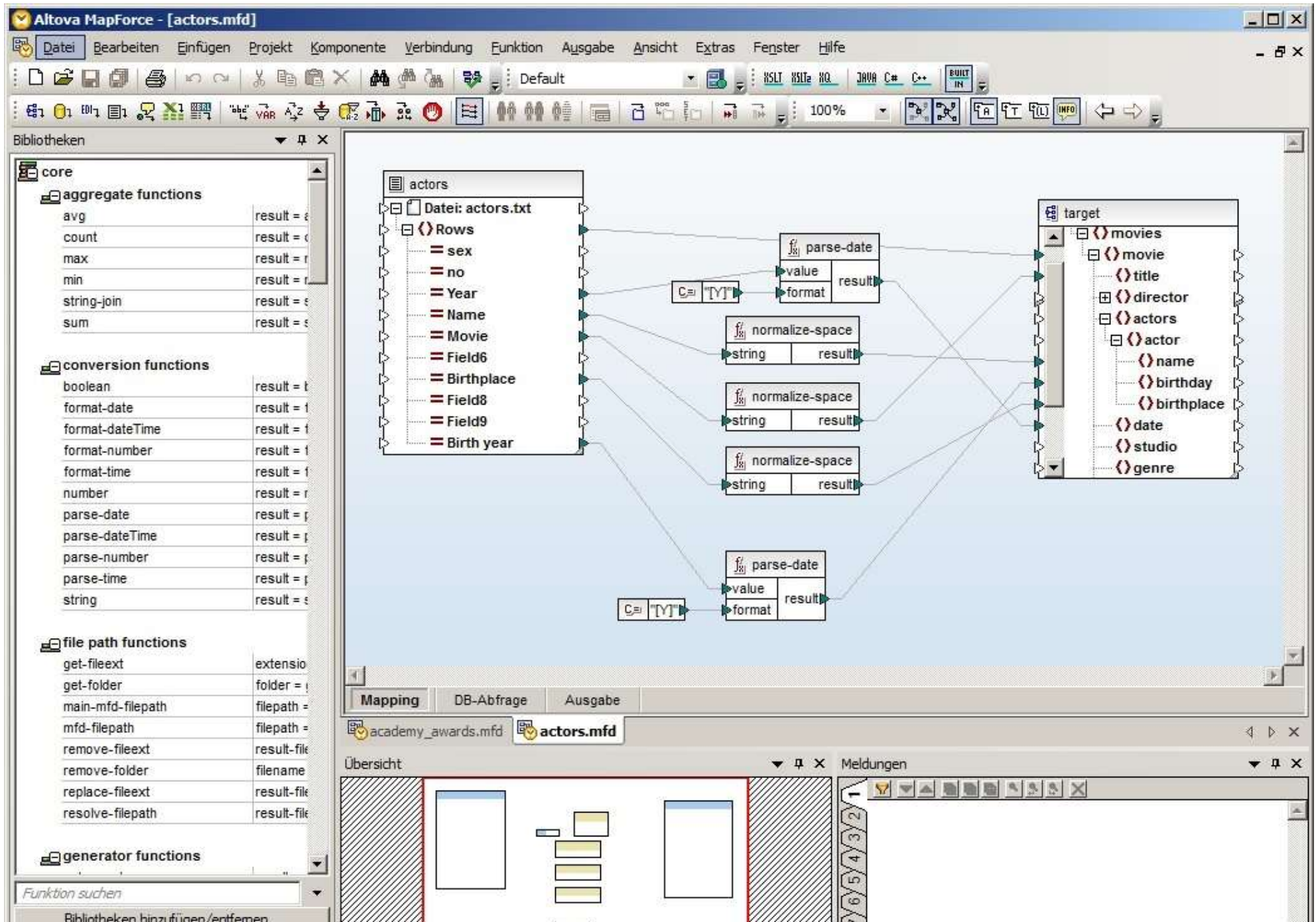
Value-to-Relation Correspondences



Types of Schema Heterogeneity that can be Captured

- Naming of
 - relations
 - attributes
 - Normalized vs. denormalized
 - Nesting vs. foreign keys
 - Below 1st normal form attributes
 - Alternative modelling
 - Relation vs. value
 - Relation vs. attribute
 - Attribute vs. value
- 1:1, 1:n, n:1
Correspondences
- Higher-order
Correspondences

Defining Correspondences



Discovering Correspondences

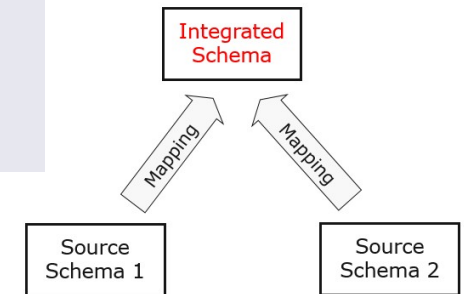
Schema Matching: Automatically or semi-automatically discover correspondences between schemata.



- Various schema matching methods exist (we will cover them later)
- Automatically finding a high-quality mapping works for simple tables within specific domains (e.g. persons, publications) but is error-prone for complex normalized schemata (e.g. databases behind ERP systems)
- In practice, schema matching is often used to create candidate correspondences that are verified by human experts afterwards

3. Schema Integration

Create a new **integrated schema** that can represent **all data** from a given set of source schemata.

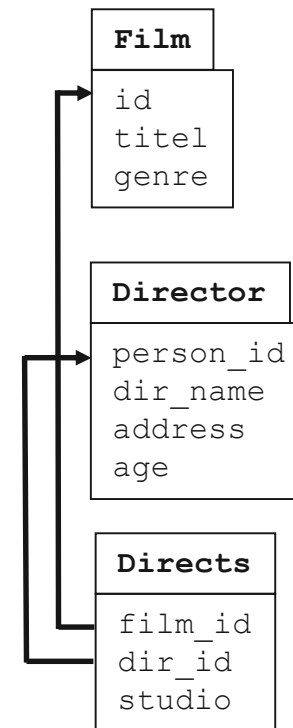


- Goals:
 - **Completeness:** All elements of the source schemata should be covered
 - **Correctness:** All data should be represented semantically correct
 - cardinalities, integrity constraints, ...
 - **Minimality:** The integrated schema should be minimal in respect to the number of relations and attributes
 - redundancy-free
 - **Understandability:** The schema should be easy to understand

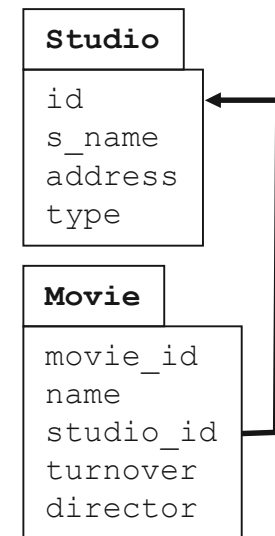
Example: Two Schemata about Films

Having a different focus and a different level of detail

- **Schema 1:** Who are the directors of a movie?
- **Schema 2:** What are the details about the studio in which the movie was shot?
- **Goals:**
 1. Completeness
 2. Correctness
 3. Minimality
 4. Understandability



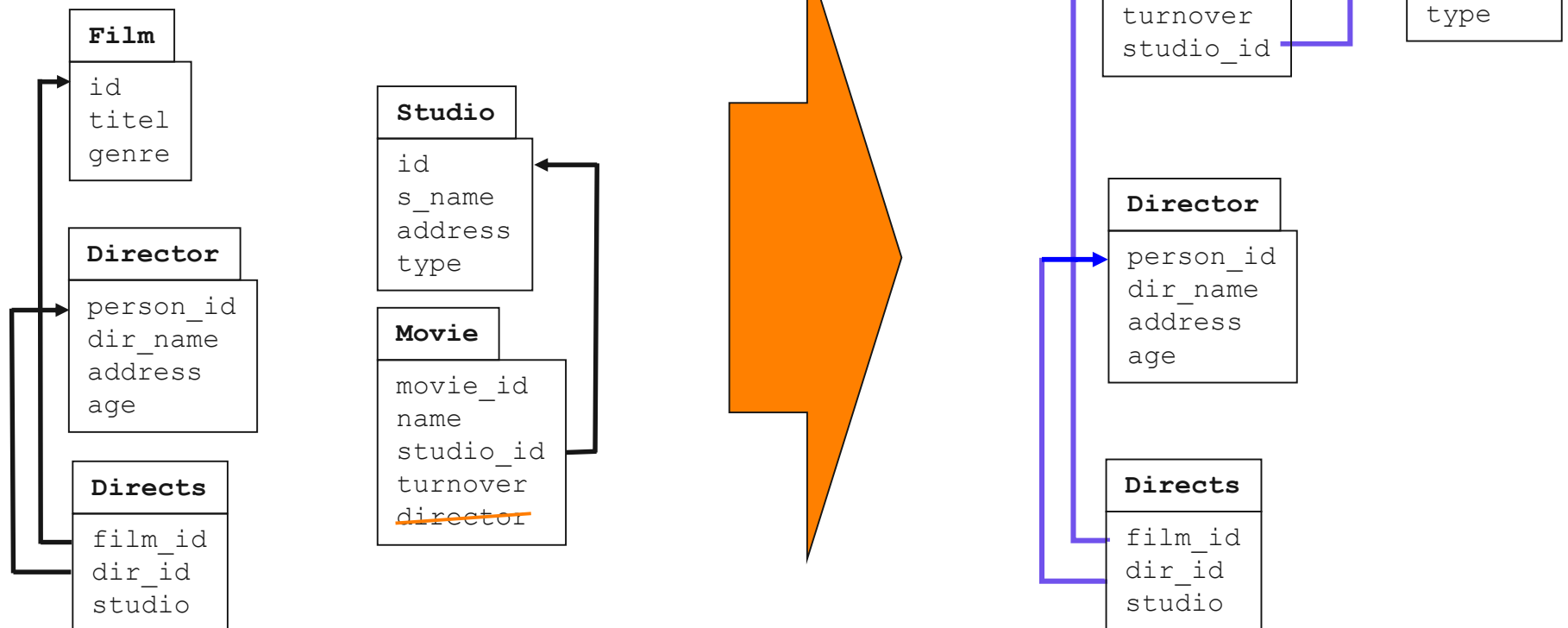
N:M
Relationship



Normalized
Schema

Schema Integration: Rules of Thumb

1. Merge all tables with their corresponding tables in other schema (Film, Movie)
2. Add all tables without corresponding tables (Director, Directs, Studio)
3. Add relationships with highest cardinality in order to keep expressivity (keep Directs)



Example of a Schema Integration Method

- Spaccapietra, et al.: Model Independent Assertions for Integration of Heterogeneous Schemas. VLDB 1992
- Input
 1. Two source schemata in **Generic Data Model**
 - classes, attributes, and relationships
 - similar to Entity Relationship Model
 2. **Correspondence Assertions**
 - correspondences between classes, attributes, and relationships
 - correspondences between paths of relationships
- Output: Integrated Schema

Integration Rules

Include into the target schema S:

1. **Equivalent classes** and merge their attribute sets

- Pick class / attribute names of your choice for equivalent classes / attributes

2. **Classes** with their attributes that are not part of any class-class correspondence (classes without direct equivalent)

3. **Direct relationships** between equivalent classes

- If $A \equiv A'$, $B \equiv B'$, $A-B \equiv A'-B'$ then include $A-B$

4. **Paths** between equivalent attributes and classes

a) If $A \equiv A'$, $B \equiv B'$, $A-B \equiv A'-A_1'-\dots-A_m'-B'$ then include the longer path

- as the length one path is subsumed by the longer path
- as the longer one is more expressive with respect to cardinality

b) If $A \equiv A'$, $B \equiv B'$, $A-A_1-\dots-A_n-B \equiv A'-A_1'-\dots-A_m'-B'$ then include both paths

- as they represent different relationships to B

5. **Equivalences between classes and attributes** are included as relationships

- again, prefer more expressive solution with respect to cardinality

Example: Class and Attribute Correspondences

- Class Correspondence

Film $\equiv$ **Movie**

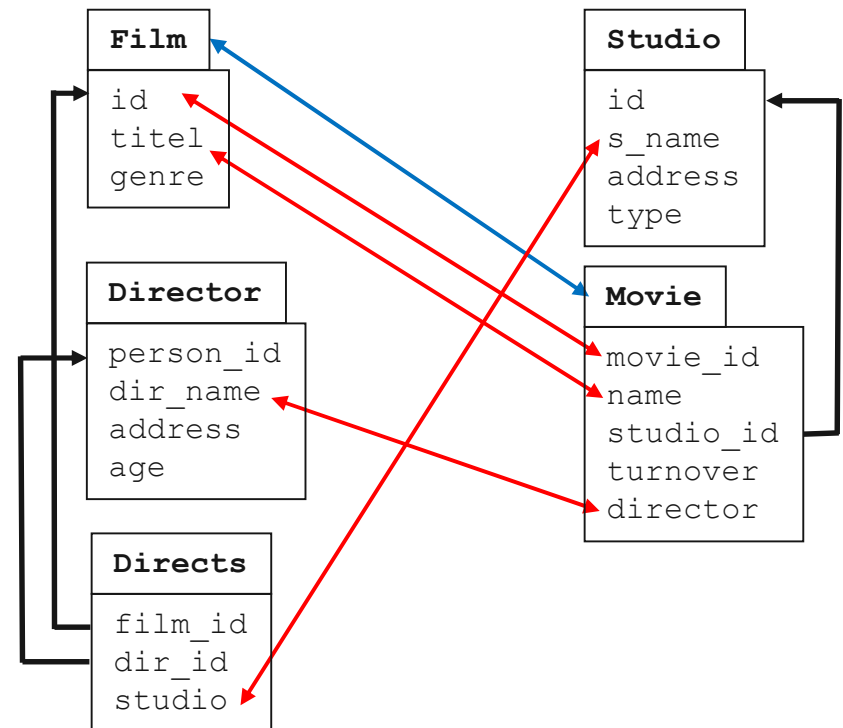
- Attribute Correspondences

id $\equiv$ **movie_id**

titel $\equiv$ **name**

dir_name $\equiv$ **director**

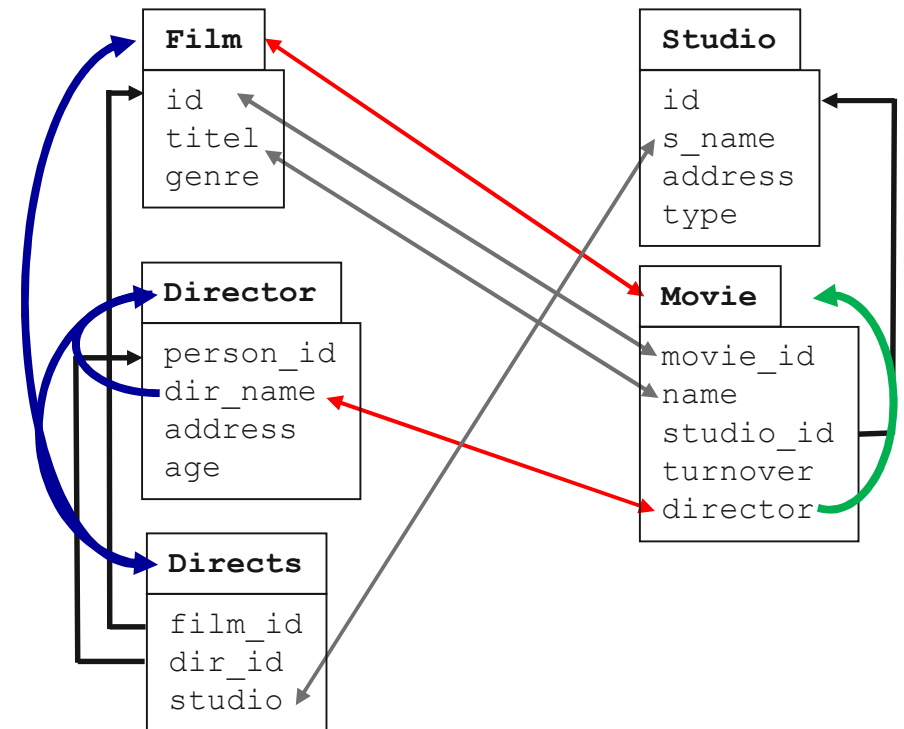
studio $\equiv$ **s_name**



Example: Relationship Path Correspondence 1

- Relationship Path Correspondence

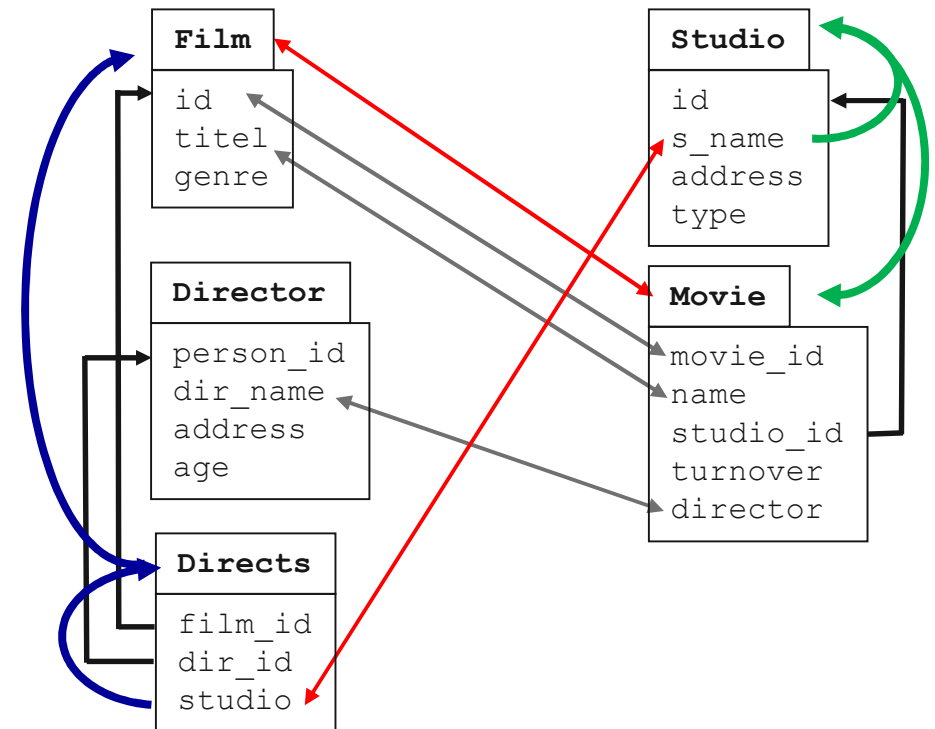
dir_name-Director-Directs-Film $\equiv$ **director-Movie**



Example: Relationship Path Correspondence 2

- Relationship Path Correspondence

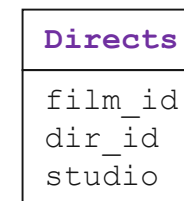
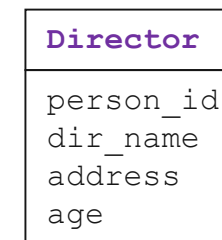
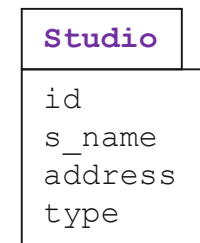
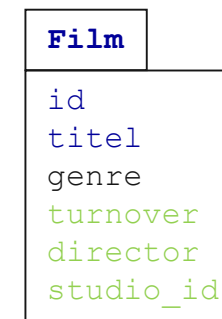
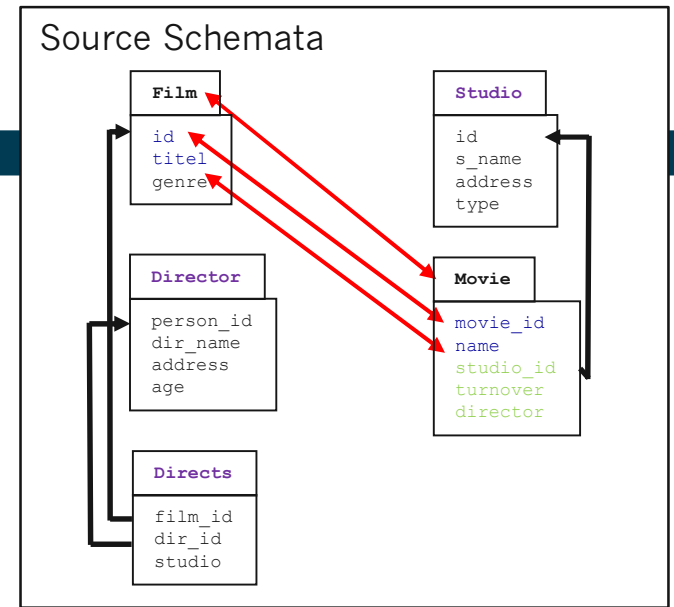
studio-Directs-Film $\equiv$ **s_name-Studio-Movie**



Creation of the Integrated Schema 1

– Integration Steps

1. Rule 1: Equivalent classes **Film** and **Movie** are merged to **Film**. Attributes are either merged (**id**, **title**) or simply copied (**turnover**, **director**, **studio_id**).
2. Rule 2: Classes without direct equivalent are included into the integrated schema (**Director**, **Directs**, **Studio**)



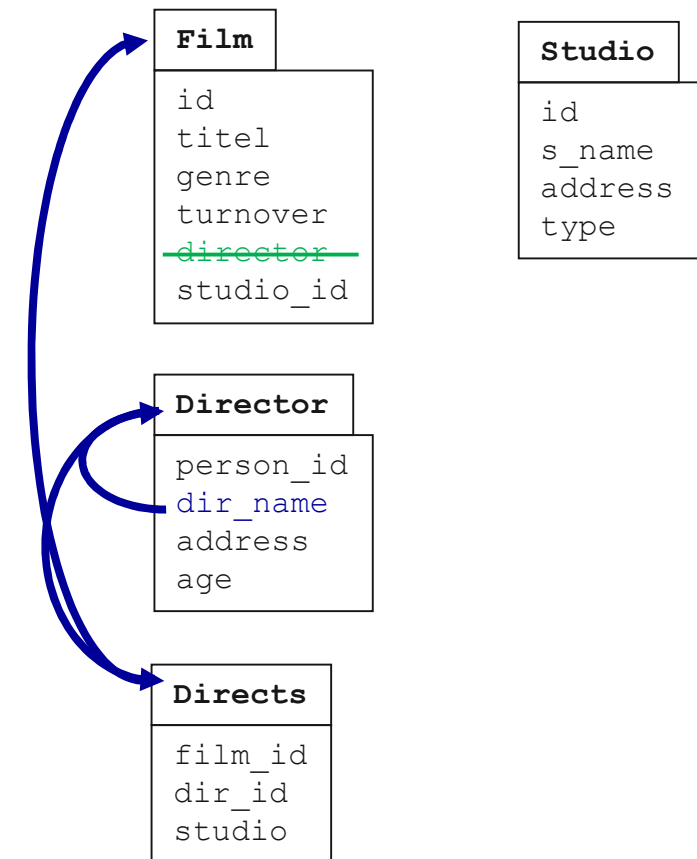
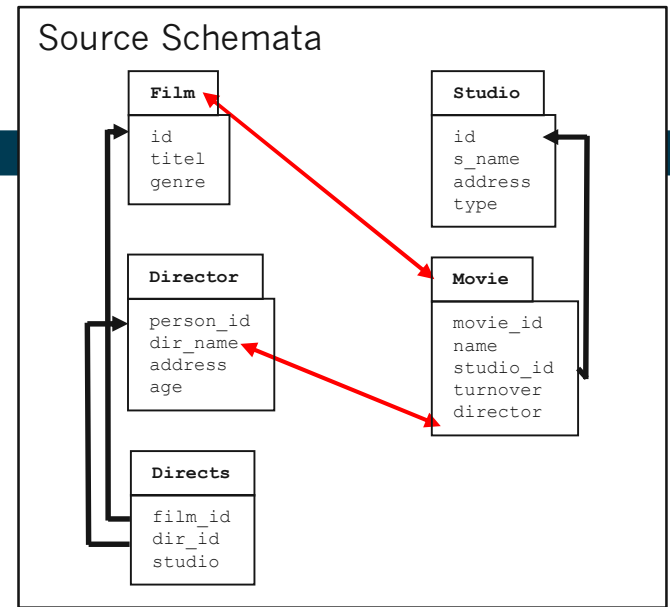
Creation of the Integrated Schema 2

– Correspondence

- `dir_name-Director-Directs-Film` $\equiv$ `director-Movie`

– Integration Steps

3. Rule 4a: The path `dir_name-Director-Directs-Film` is included. The path `director-Movie` is left out as it is less expressive (allows only one director per movie).
4. Thus, `dir_name` is kept and `director` removed.



Creation of the Integrated Schema 3

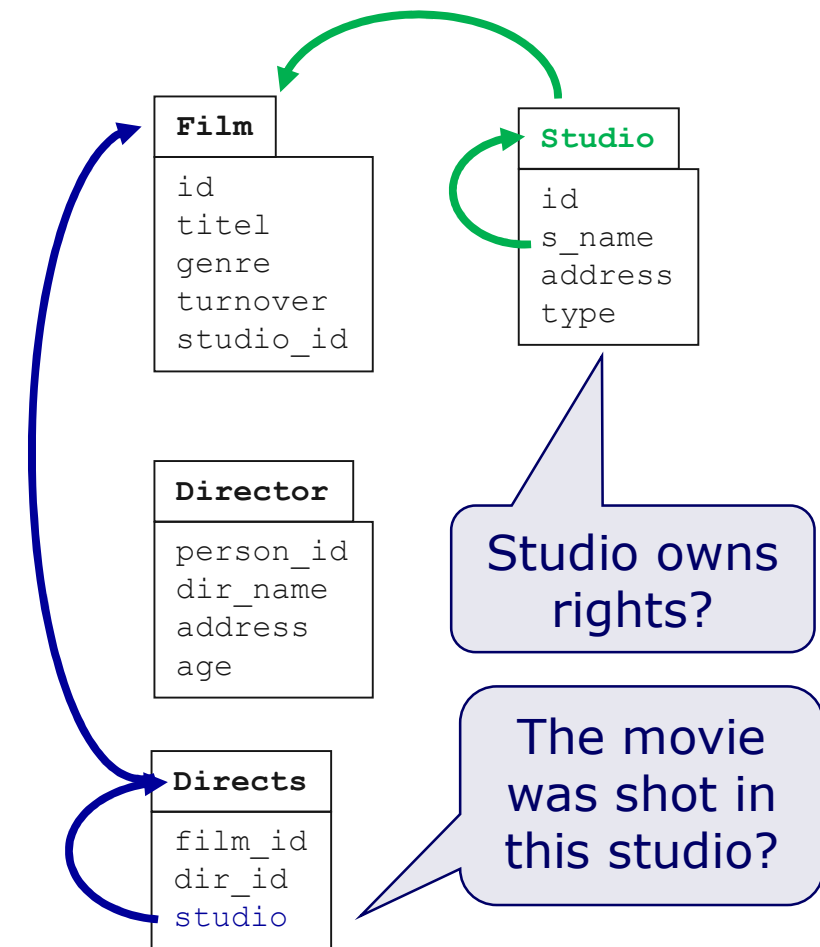
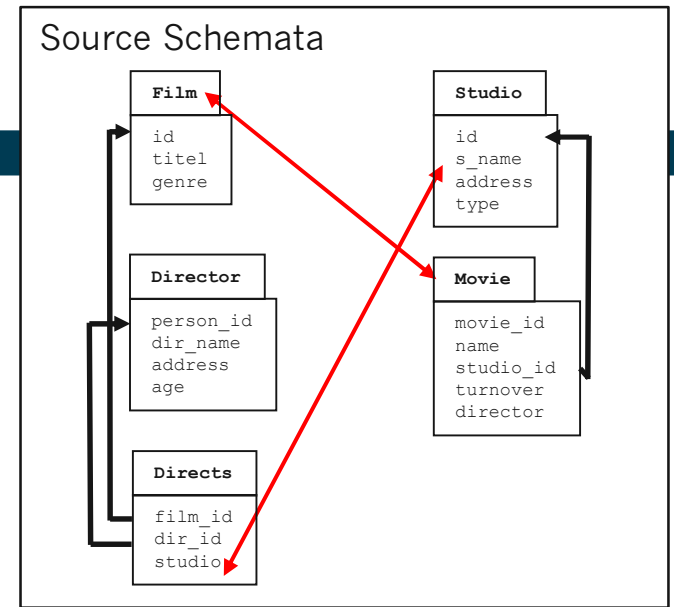
– Correspondence

studio-Directs-Film $\equiv$
s_name-Studio-Movie

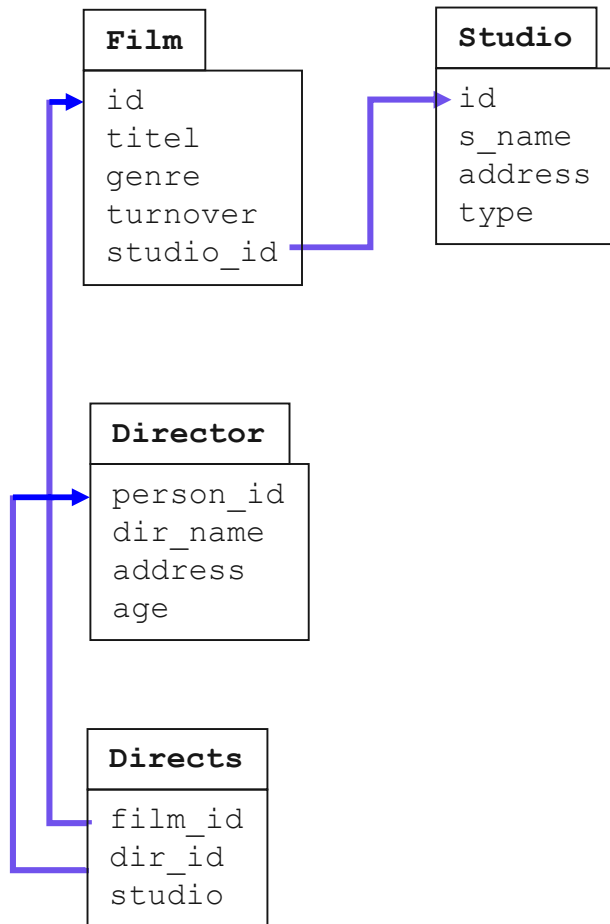
– Integration Step

5. Rule 4b: Both paths are included as both have a length > 1 .

- **Studio** and **studio** are not merged as they have a different relationship to the surrounding classes and might thus mean different things.



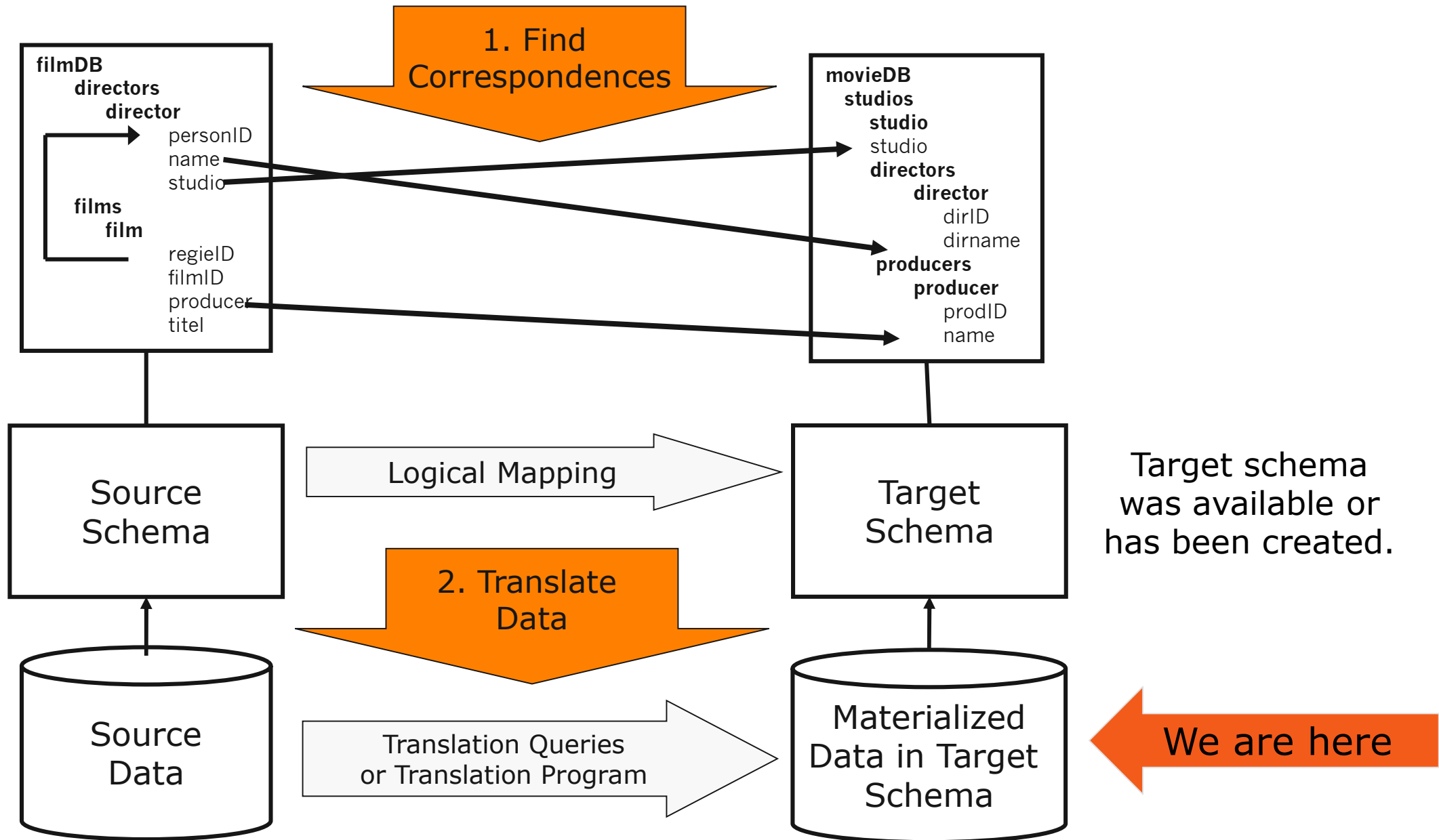
Final Integrated Schema



Fulfills the schema integration goals

- **Completeness**: All elements of the source schemata covered
- **Correctness**: All data can be represented semantically correct
- **Minimality**: The integrated schema is minimal in respect to the number of relations and attributes
- **Understandability**: The schema is easy to understand

4. Data Translation



Query Generation

Goal: Derive suitable data translation queries (or programs) from the correspondences.

- Possible query types: SQL Select Into, SPARQL Construct, XSLT
- Example of a data translation query:

ARTICLE		PUBLICATION
• artPK	→	• pubID
• heading	→	• title
		• date

```
SELECT artPK AS pubID
       heading AS title
       null AS date
INTO PUBLICATION
FROM ARTICLE
```

- Challenges for more complex schemata
 - correspondences are not isolated but embedded into context (tables, relationships)
 - might require joining tables to overcome **different levels of normalization**
 - might require combining data from multiple source tables (**horizontal partitioning**)
 - will likely require **value normalization** and potentially **information extraction**

Normalized → Denormalized

ARTICLE

- artPK
- title
- pages

AUTHOR

- artFK
- name

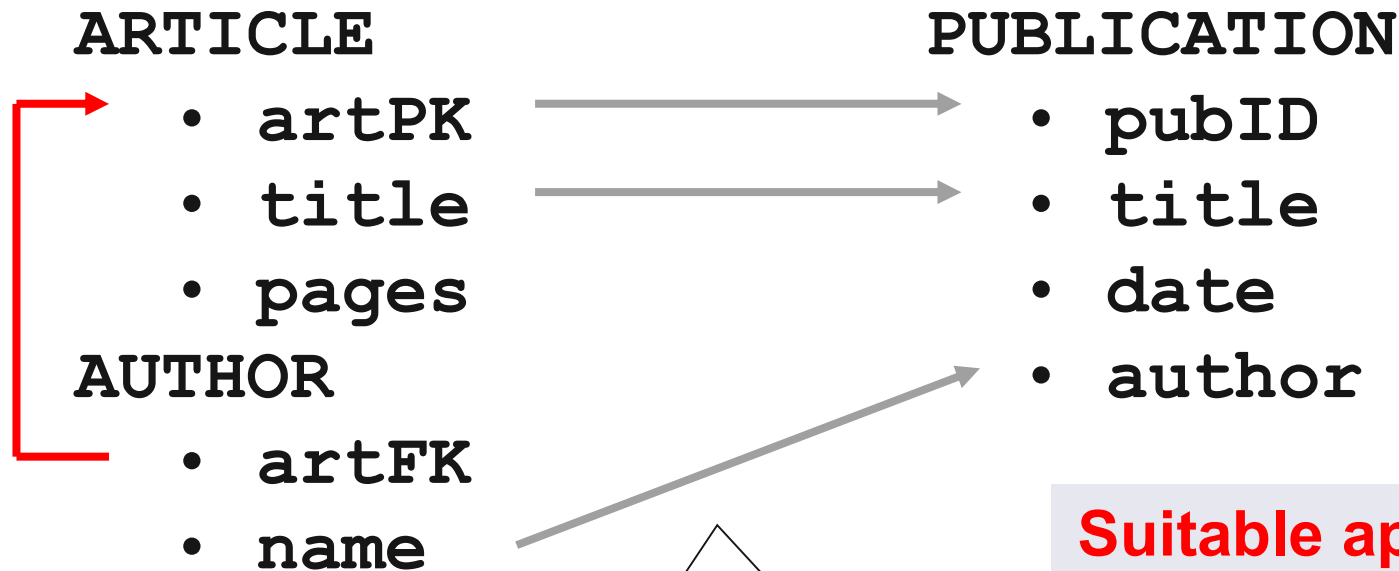
PUBLICATION

- pubID
- title
- date
- author

Naïve approach with one query per source table **does not work.**

```
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
INTO PUBLICATION          INTO PUBLICATION
FROM ARTICLE              FROM AUTHOR
```

Normalized → Denormalized



Suitable approach: Join tables using foreign key relationship.

```
SELECT      artPK AS pubID
            title AS title
            null AS date
            name AS author
INTO        PUBLICATION
FROM        ARTICLE, AUTHOR
WHERE       ARTICLE.artPK = AUTHOR.artFK
```

Denormalized → Normalized

PUBLICATION

- title
- date
- author

ARTICLE

- artPK
- title
- pages

AUTHOR

- artFK
- name

DISTINCT

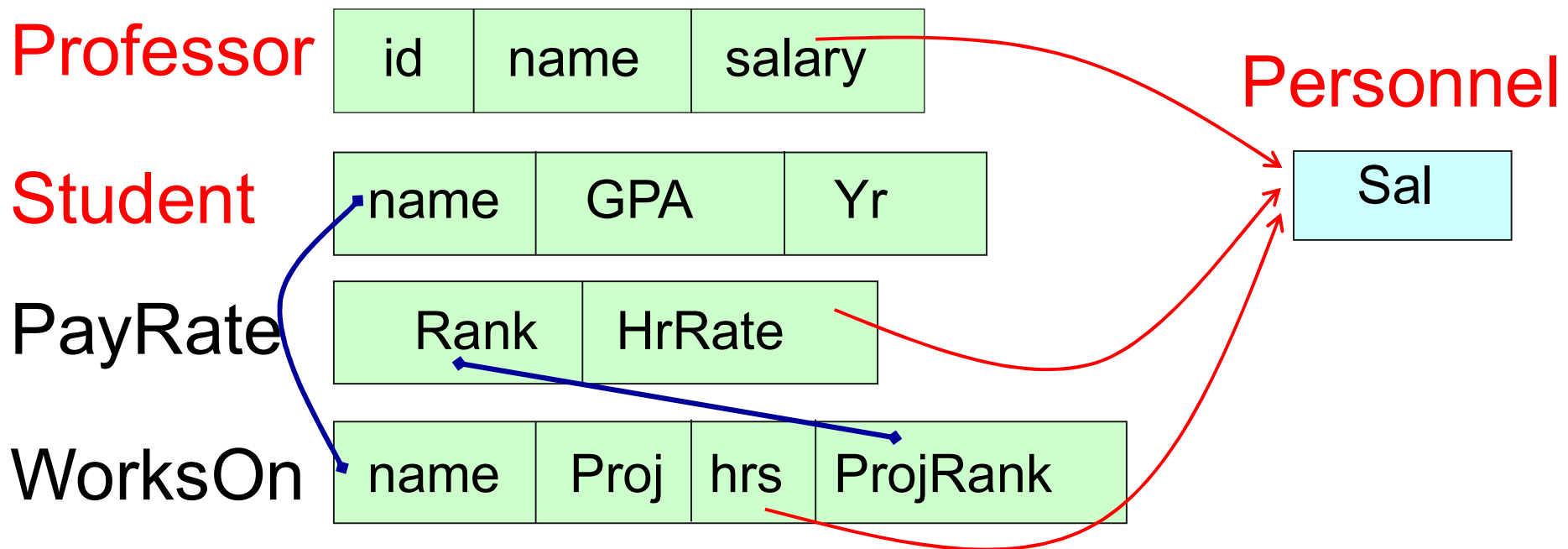
```
SELECT SK(title) AS artPK
       title AS title
       null AS pages
INTO ARTICLE
FROM PUBLICATION
```

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

SK(): Skolem function used to generate unique keys from distinct values, e.g. hash function.

Horizontal Partitioning / Many-to-One Correspondences

Data for target table might be horizontally distributed over multiple source tables.



1:1 Correspondence: Professor.salary → Personnel.Sal

N:1 Correspondence : PayRate.HrRate * WorksOn.Hrs → Personnel.Sal

UNION the Salaries of Professors and Students

1:1 Correspondence: Professor.salary → Personnel.Sal

N:1 Correspondence: PayRate.HrRate * WorksOn.Hrs → Personnel.Sal

```
INSERT INTO Personal (Sal)
```

```
SELECT salary  
FROM Professor
```

```
UNION
```

```
SELECT P.HrRate * W.hrs  
FROM PayRate P, WorksOn W  
WHERE P.Rank = W.ProjRank
```

Information Extraction / One-to-Many Correspondences

One-to-Many correspondences might require the application of **information extraction techniques to split string values.**

- Possible techniques
 - **Regular expressions:** restricted expressiveness, often rather cryptic
 - **Code:** flexible, but high maintenance effort
 - **PLMs** (e.g. BERT): require large amount of training data for fine-tuning
 - **LLMs** (e.g. GPT, Claude): offer decent zero-shot performance, using hosted models may result in high processing costs
- Consideration
 - Robustness vs. cost of training/applying

Input value

"Title": Dr. Brown's Infant-to-Toddler, Toothbrush Set, 1.4 Ounce, Blue



One-to-Many

Extracted values in target schema

"Brand": "Dr. Brown's",
"Age": "Infant-to-Toddler",
"Item Form": "Toothbrush Set",
"Color": "Blue",
"Material": "n/a"

Xu, et al.: Large Language Models for Generative Information Extraction: A Survey. Frontiers of Computer Science, 2024.
Brinkmann, et al.: ExtractGPT: Exploring the potential of Large Language Models for product attribute value extraction. 2024.

Value Normalization

Value normalization is the process of transforming data values into **a single, consistent value format** so they can be accurately compared, merged, or analyzed together.

Operation	Attributes	Examples
Name Expansion	Manufacturer, Generation, Capacity, Cache	"HP" → "Hewlett-Packard" "PII" → "Pentium II" "G1" → "Generation 1"
Generalization	Product Type, Color, Processor Type	"Oatmeal" → "Snacks and Breakfast" "Sparkling Juices" → "Beverages" "Neon Lime Green" → "Green"
Unit of Measurement Normalization	Dimensions, Paper Weight, Size/Weight, Rotational Speed, Pack Quantity	"7" → "17.8" "164 ft" → "4998.7" "20-lb." → "9.06" (kg) "0.31 oz" → "879" (g) "10k" → "10000"
String Wrangling	Identifiers, Ports Processor Core, Retail UPC, Brand	"CTW-4M(208)" → "CTW4M208" "Dual Port" → "2" "4-Core" → "4" "Quaker Foods" → "QUAKER FOODS"

Techniques used: regular expressions, code, classifiers, LLMs

Jaimovitch-López, et al.: Can language models automate data wrangling? Mach. Learn. 112(6), 2053–2082, 2023.
Brinkmann, et al.: Using LLMs for the extraction and normalization of product attribute values. ADBIS 2024.

Complete Algorithms for Generating Translation Queries

- Relational Case
 - Doan, Halevy, Ives: Principles of Data Integration. Pages 152-158.
- XML Case
 - Leser, Naumann: Informationsintegration. Pages 137-143.
- MapForce
 - implements another one which we will try out in the exercise

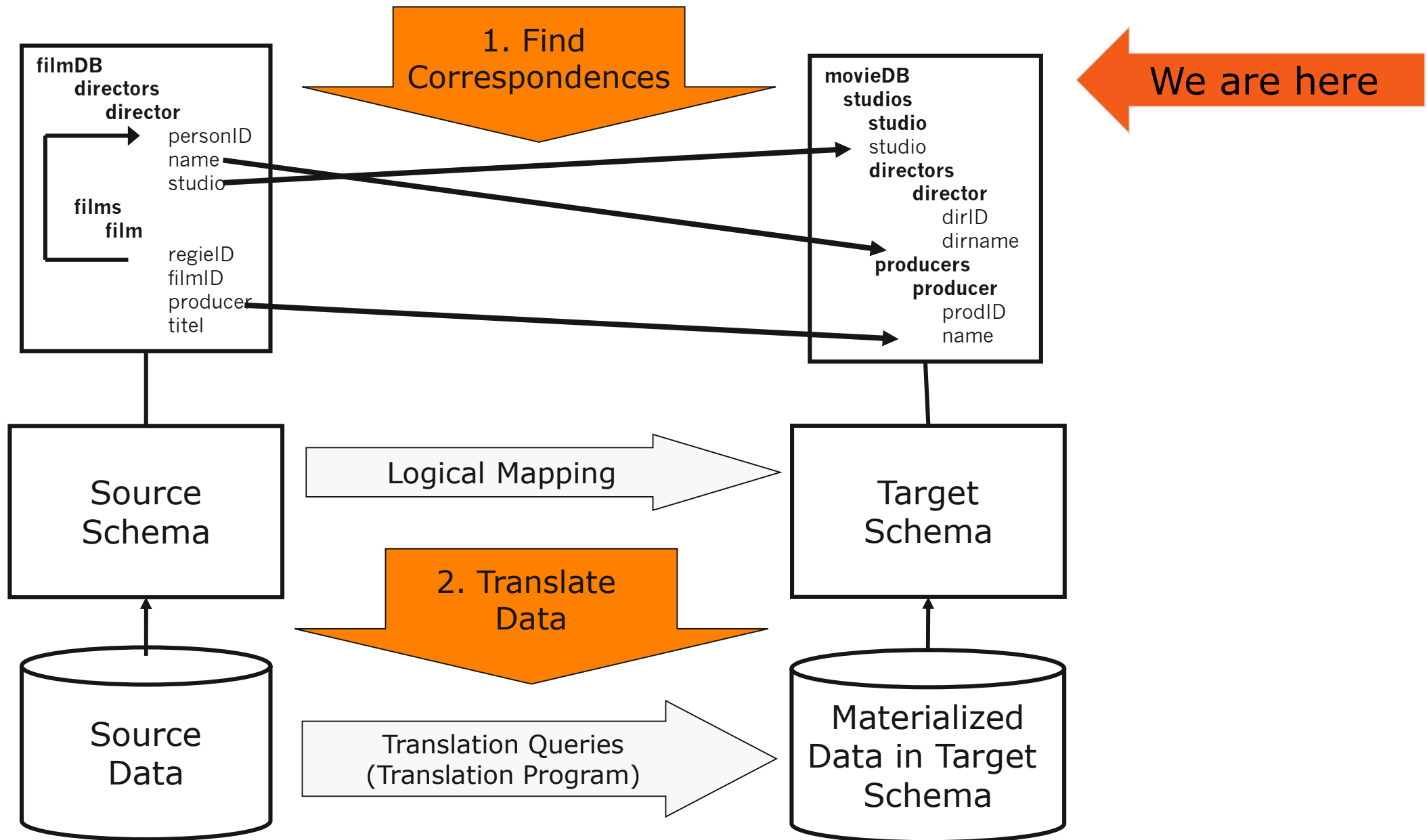
5. Schema Matching

Schema Matching: Automatically or semi-automatically discover correspondences between schemata.



- Automatically finding a complete high-quality mapping (= set of all correspondences) is difficult in more complex use cases
 - ERP databases versus simple tables on the Web versus corporate data lakes
- In practice, schema matching is often used to create **candidate correspondences** that are verified by domain experts afterwards
- Most schema matching methods focus on **1:1 correspondences**
 - we also restrict ourselves to 1:1 correspondences

Schema Matching



Outline: Schema Matching

1. Challenges to Finding Correspondences
2. Schema Matching Methods
 1. Label-based Methods
 2. Instance-based Methods
 3. Structure-based Methods
 4. Combined Approaches
3. Generating Correspondences from the Similarity Matrix
4. PLM- and LLM-based Schema Matching
5. Table Annotation
6. Summary and Current Trends

5.1 Challenges to Finding Correspondences

1. Large schemata

- >100 tables and >1000 attributes

2. Esoteric naming conventions and different languages

- 4-character abbreviations: SPEY
- city vs. ciudad vs. مدينة

3. Generic, automatically generated names

- attribute1, attribute2, attribute3
(was used as names for product features in Amazon API)

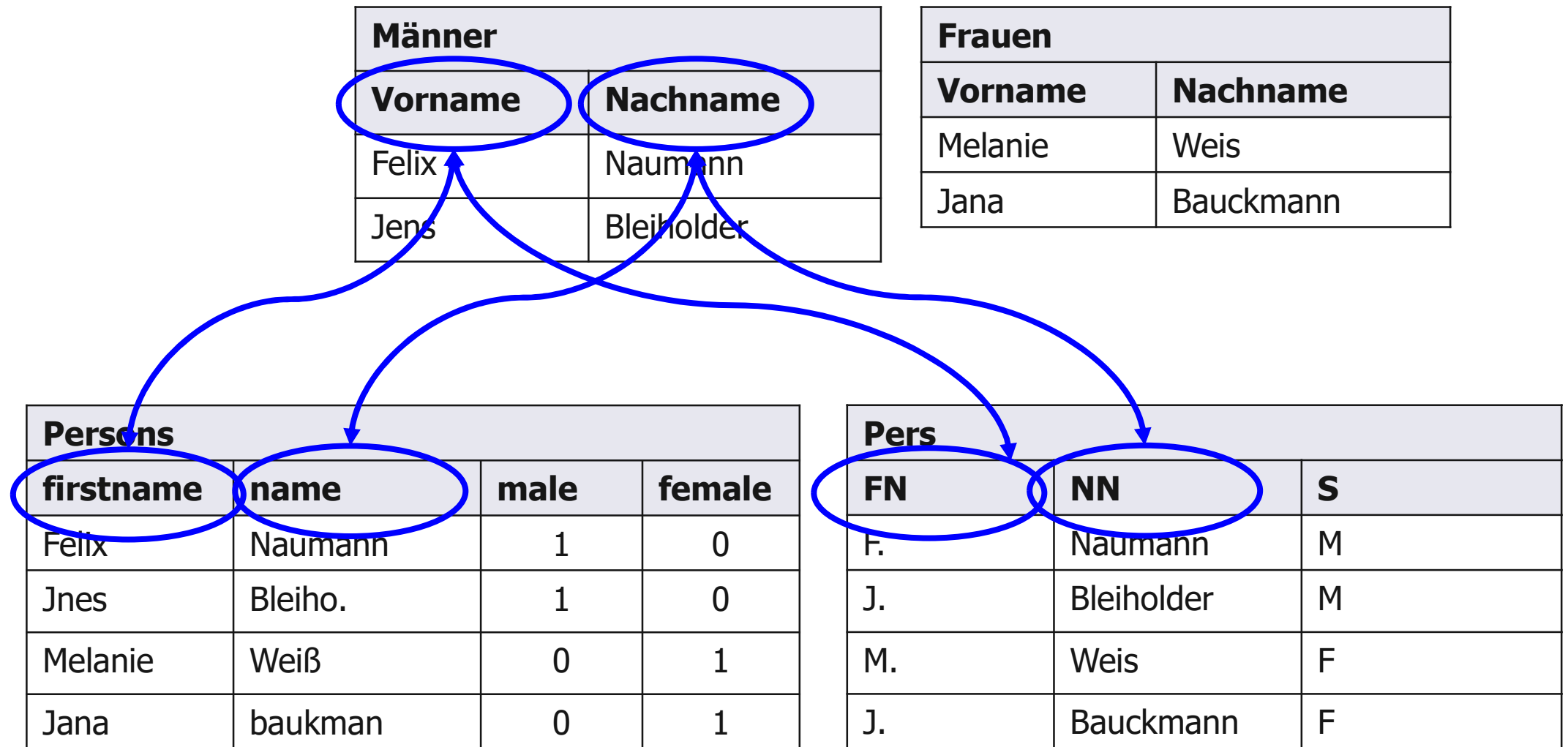
4. Semantic heterogeneity

- synonyms, homonyms, ...

5. Missing documentation

- no dataset metadata, data dictionary

Problem Space: Different Languages and Strange Names



How do humans know?

- We recognize **naming conventions** and different **languages**
 - use **table context**
 - **values** look like first names and surnames
 - values **look similar**
 - if there is a first name, there is usually also a surname
 - **persons** have first- and surnames
 - **man** are persons
- ➔ Recognizing these clues is hard for the computer (without additional resources or lots of pre-training)

Human

Männer	
Vorname	Nachname
Felix	Naumann
Jens	Bleiholder

Persons			
FirstNa	Name	male	femal
Felix	Naumann	1	0
Jnes	Bleiho.	1	0
Melanie	WeiB	0	1
Jana	baukman	0	1

☼*■□♣□	
†□□♣	♠♣■□♣
♣♣●✕□	♠□◆○□■
☺♣■♦	✕♣□●♣♣

Computer

♣♣□♦□♣■		
†□□♣	♠■□♣	■●♣
♣♣●✕	♠□◆○■	📁
☺♣■♦	♣●♣♣	📁
☼♣●□	♣♣✕	📁

5.2. Schema Matching Methods

1. **Label-based Methods:** Rely on the names of schema elements
2. **Instance-based Methods:** Compare the actual data values
3. **Structure-based Methods:** Exploit the structure of the schema
4. **Combined Approaches:** Use combinations of above methods

Source: Erhard Rahm and Philip Bernstein: A survey of approaches to automatic schema matching., VLDB Journal 10(4), 2001.

5.2.1 Label-based Schema Matching Methods

- Given two schemata with the attribute (class) sets A and B
 - **A**={ID, Name, Vorname, Alter}, **B**={No, Name, First_name, Age}
- Approach
 1. Generate **cross product** of all attributes (classes) from A and B
 2. For each pair calculate the **similarity of the attribute labels**
 - using some similarity metric: Levenshtein, Jaccard, Soundex, etc.
 3. The most similar pairs are the matches

	ID	Name	Vorname	Alter
No	0.8	0.6	0.4	0.4
Name	0.1	1.0	0.6	0.3
First_name	0.2	0.6	0.5	0.3
Age	0.4	0.3	0.2	0.7

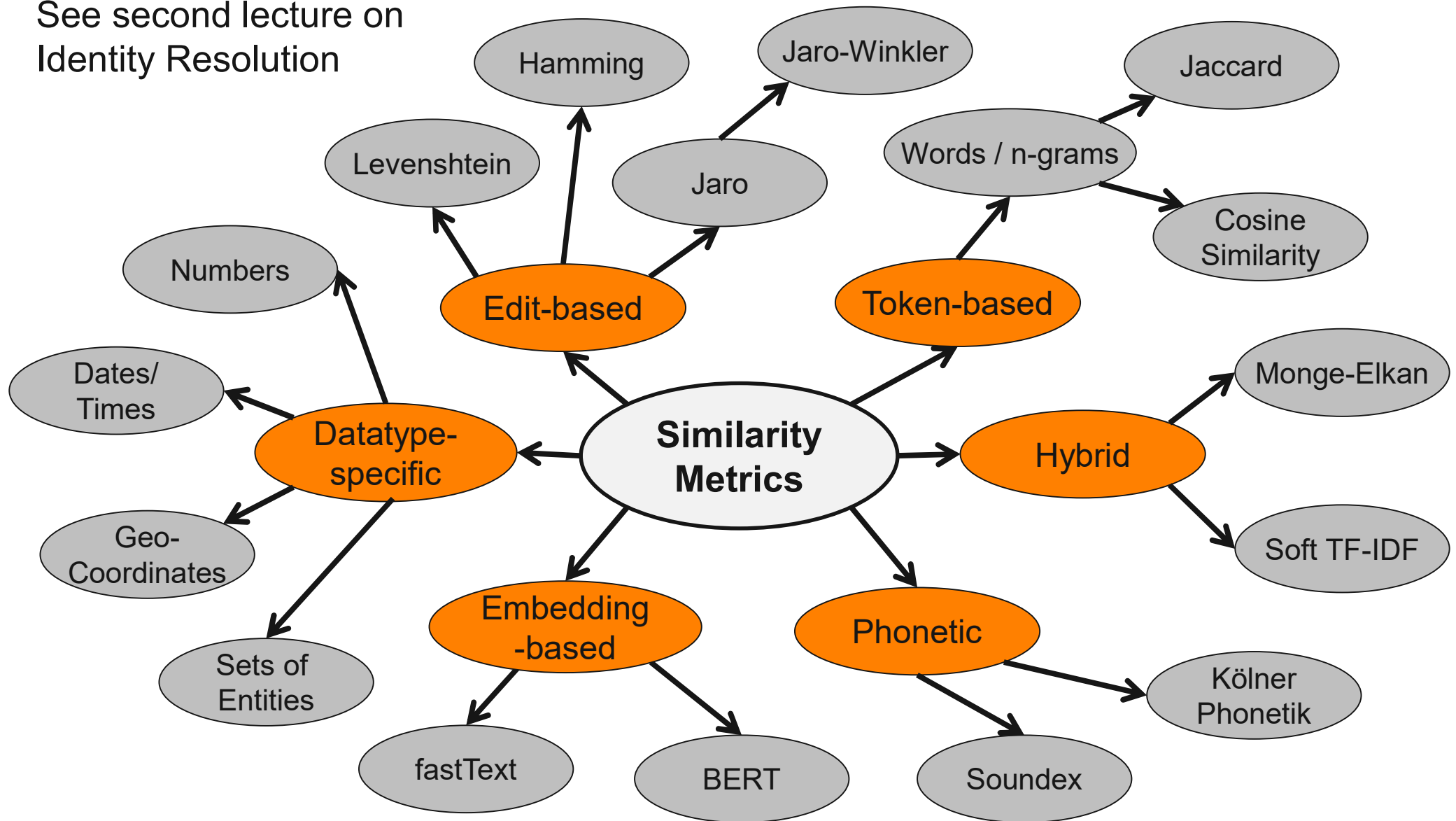
Example Metric: Levenshtein

- Measures the dissimilarity of two strings
- Measures the **minimum number of edits** needed to transform one string into the other
- Allowed edit operations
 - **insert** a character into the string
 - **delete** a character from the string
 - **replace** one character with a different character
- Examples
 - $\text{levensthein}(\text{'table'}, \text{'cable'}) = 1$ (1 substitution)
 - $\text{levensthein}(\text{'Chris Bizer'}, \text{'Bizer, Chris'}) = 11$ (10 substitution, 1 insertion)
- Converting Levenshtein distance into a similarity

$$\text{sim}_{\text{Levenshtein}} = 1 - \frac{\text{LevenshteinDist}}{\max(|s_1|, |s_2|)}$$

A Wide Range of Similarity Metrics Exists

See second lecture on
Identity Resolution



Problems of Label-based Schema Matching

1. Semantic heterogeneity is not recognized

- the labels of schema elements only partly capture their semantics
- synonyms und homonyms → embedding-based methods potentially better

2. Problems with different naming conventions

- Abbreviations: pers = person, dep = department
- Combined terms and ordering: id_pers_dep vs. DepartmentPersonNumber
- Different languages: city vs. ciudad vs. مدينة
- We need to apply **smart, domain-specific tweaks**:
 1. Preprocessing: Normalize labels in order to prepare them for matching
 2. Matching: Employ similarity metrics that fit the specifics of the schemata

Pre-Processing of Labels

- Case and Punctuation Normalization
 - ISBN, IsbN, and I.S.B.N → isbn
- Explanation Removal
 - GDP (as of 2014, US\$) → gdp
- Stop Word Removal
 - in, at, of, and, ...
 - ex1:locatedIn → ex1:located
- Stemming
 - ex1:located, ex2:location → both stemmed to ,locat‘
 - but: ex1:locationOf, ex2:locatedIn (Inverse Properties!)
- Tokenization
 - ex1:graduated_from_university → {graduated,from,university}
 - ex2:isGraduateFromUniversity → {is,Graduate,from,University}
 - tokens are then compared one-by-one using for instance Jaccard similarity

Use Linguistic Resources for Pre-Processing

- Translate labels into target language
 - ciudad and مدينة → city
 - using for instance Google Translate
- Expand known abbreviations or acronyms
 - loc → location, cust → customer
 - using a domain-specific list of abbreviations or acronyms
- Expand with synonyms
 - add cost to price, United States to USA
 - using a domain-specific dictionary of synonyms or WordNet
- Expand with hypernyms (is-a relationships)
 - generalize book, laptop into product
 - using a domain-specific taxonomy or cross-domain resource, e.g. WordNet, DBpedia, WebIsA
- Use an LLM and hope that it can do all these things

5.2.2 Instance-based Schema Matching Methods

- Given two schemata with the attribute sets A and B and
 - all instances (records) of A and B or
 - a sample of the instances of A and B
- Approach
 - determine correspondences between A and B by examining which attributes in A and B **contain similar values**
 - as values often better capture the semantics of an attribute than its label
- Types of instance-based methods
 1. Attribute Recognizers
 2. Value Overlap
 3. Feature-based Methods
 4. Duplicate-based Methods

Table A	
A1	A2
Felix	Naumann
Jens	Bleiholder

Table B	
VN	NN
Felix	Naumann
Jens	Bleiholder

Attribute Recognizers and Value Overlap

1. Attribute Recognizers

- employ dictionaries, regexes or rules to recognize values of a specific attribute
 - Dictionaries fit attributes that only contain a relatively small set of values (e.g. age classification of movies (G, PG, PG-13, R), country names, US states)
 - Regexes or rules fit attributes with regular values (e.g. area code – phone number).
- similarity = fraction of the values of attribute B that match dictionary/rule of attribute A

2. Value Overlap

- calculate the similarity of attribute A and B as the the overlap of their values using the Jaccard similarity measure (or Generalized Jaccard):

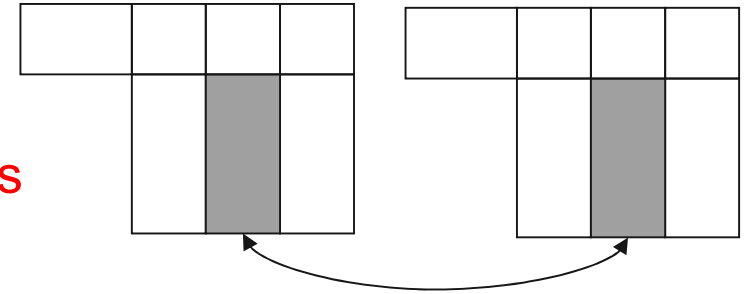
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Feature-based Methods

- Given two schemata with the attribute sets A and B and instances of A and B

- Approach

1. For each attribute calculate **interesting features** using the instance data, e.g.
 - attribute data type
 - average string length of attribute values
 - average maximal and minimal number of words
 - average, maximal and minimal value of numbers
 - standard derivation of numbers
 - does the attribute contain NULL values?
2. generate the cross product of all attributes from A and B
3. for each pair compare the similarity of the features



Example: Feature-based Matching

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

Nr	Adresse	Telefon
9	Seeweg, Berlin	030-3324566
3	Aalstr, Schwedt	0330-1247765
7	Rosenallee, Koche1	0884-334621

- Features: **Attribute data type, average string length**
 - Table1 = {(ID, NUM, 1), (Name, STR, 6), (Loc, STR, 18)}
 - Table2 = {(Nr, NUM, 1), (Adresse, STR, 16), (Telefon, STR, 11)}
- Similarity measure: **Euclidean Distance** (NUM=0, STR=1)

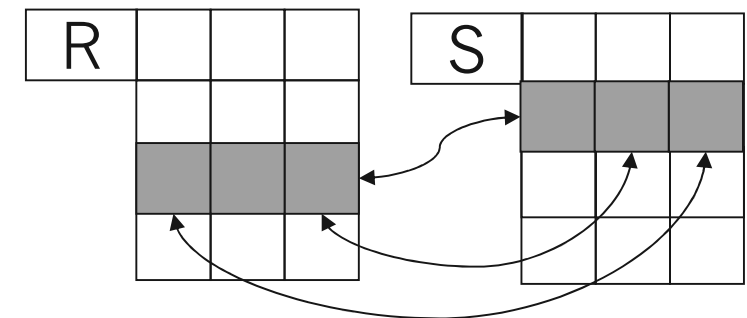
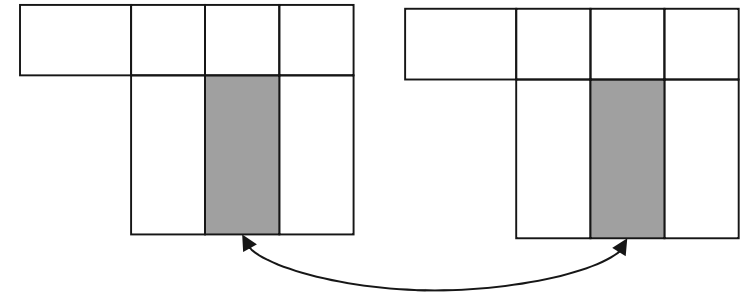
	ID	Name	Loc
Nr	$d(<0, 1>, <0, 1>)$	$d(<1, 6>, <0, 1>)$	$d(<1, 18>, <0, 1>)$
Adresse	$d(<0, 1>, <1, 16>)$	$d(<1, 6>, <1, 16>)$	$d(<1, 18>, <1, 16>)$
Telefon	$d(<0, 1>, <1, 11>)$	$d(<1, 6>, <1, 11>)$	$d(<1, 18>, <1, 11>)$

Discussion: Feature-based Methods

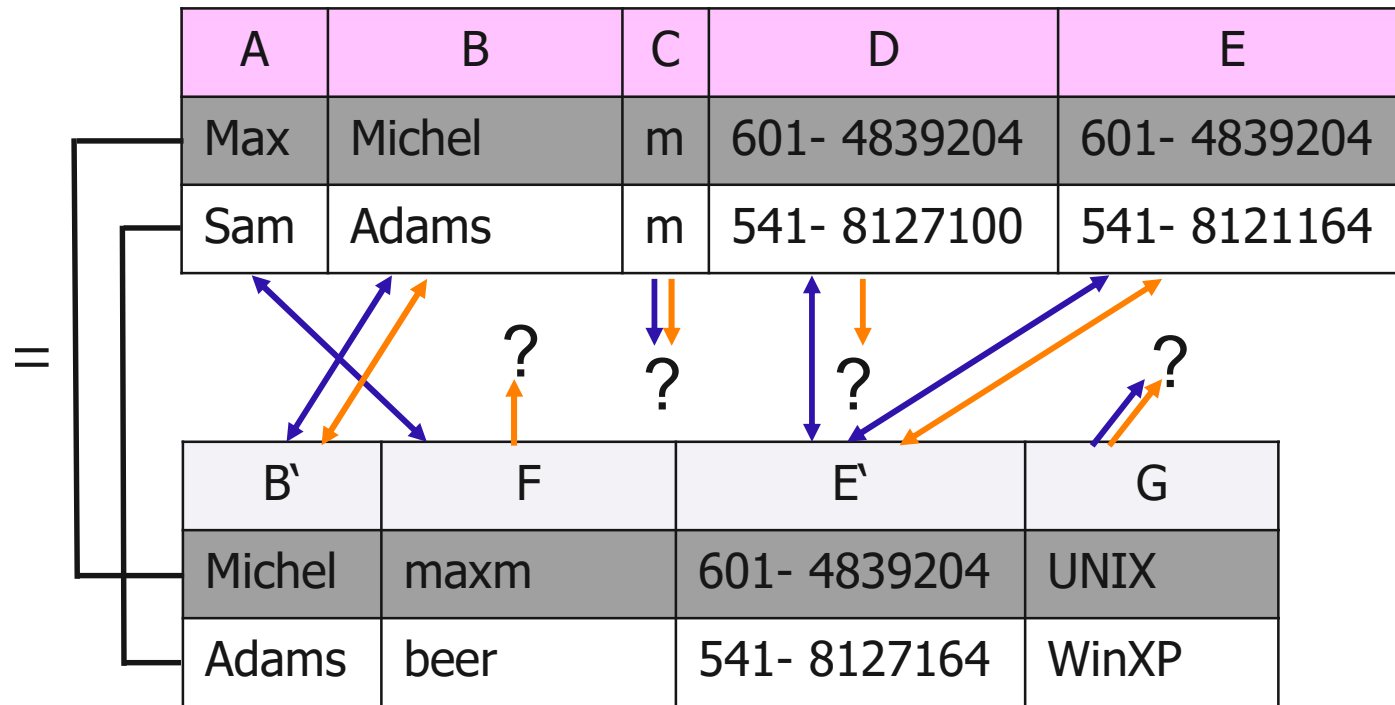
1. Require decision which features to use
 - good features depend on the attribute data type and application domain
2. Require decision how to compare and combine values
 - e.g. cosine similarity, Euclidian distance of normalized values, ...
 - different features likely require different weights
3. Similar attribute values do not always imply same semantics
 - phone number versus fax number
 - employee name versus customer name

Duplicate-based Methods

- Classical instance-based matching is **vertical**
 - Comparison of complete columns
 - ignores the relationships between instances
- Duplicate-based matching is **horizontal**
 1. Find (some) potential duplicates or use existing knowledge about duplicates
 2. Check which attribute values **closely match in each duplicate**
 3. Result: **Attribute correspondences per duplicate**
 4. Aggregate the attribute correspondences on duplicate-level into attribute correspondences on schema-level using **majority voting**.

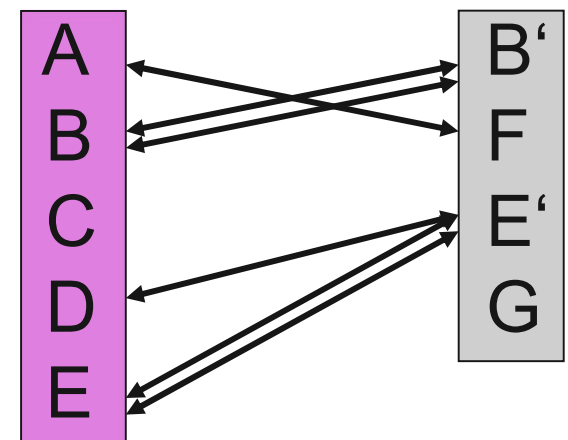


Example: Vote of Two Duplicates



Vote of the two duplicates:

Resulting schema-level correspondences:

$$B \equiv B', E \equiv E', A \equiv F$$


Using Duplicates for Cross-Language Infobox Matching

Coordinates:  52°30'2"N 13°23'56"E			
Country	Germany		
Government			
- Governing Mayor	Klaus Wowereit (SPD)		
- Governing parties	SPD / Die Linke		
- Votes in Bundesrat	4 (of 69)		
Area			
- City	891.85 km ² (344.3 sq mi)		
Elevation	34 - 115 m (-343 ft)		
Population (31 March 2010) ^[1]			
- City	3,440,441		
- Density	3,857.6/km ² (9,991.3/sq mi)		
- Metro	4,429,847		
Time zone	CET (UTC+1)		
- Summer (DST)	CEST (UTC+2)		
Postal code(s)	10001–14199		
Area code(s)	030		
ISO 3166 code	DE-BE		
Vehicle registration	B		
GDP/ Nominal	€ 90.1 ^[2] billion (2009) [citation needed]		
NUTS Region	DE3		
Website	berlin.de 		
		Basisdaten	
		Fläche:	891,85 km ² (14.)
		Einwohner:	3.456.264 ^[1] (8.) (31. Oktober 2010)
		Bevölkerungsdichte:	3.875 Einw. je km ² (1.) als Bundesland, (2.) als Gemeinde
		BIP:	90,1 Mrd. € (2009)
		Höhe:	34–115 m ü. NN
		Geografische Lage:	52° 31' N, 13° 24' O
		Zeitzone:	Mitteleuropäische Zeit (MEZ) UTC+1
		Postleitzahlen:	10115–14199
		Vorwahl:	030
		Kfz-Kennzeichen:	B
		Gemeindeschlüssel:	11 0 00 000
		ISO 3166-2:	DE-BE
		UN/LOCODE:	DE BER
		Website:	www.berlin.de 
		Politik	
		Reg. Bürgermeister:	Klaus Wowereit (SPD)
		Reg. Parteien:	SPD und Die Linke
		Sitzverteilung im Abgeordnetenhaus	SPD 54 CDU 36

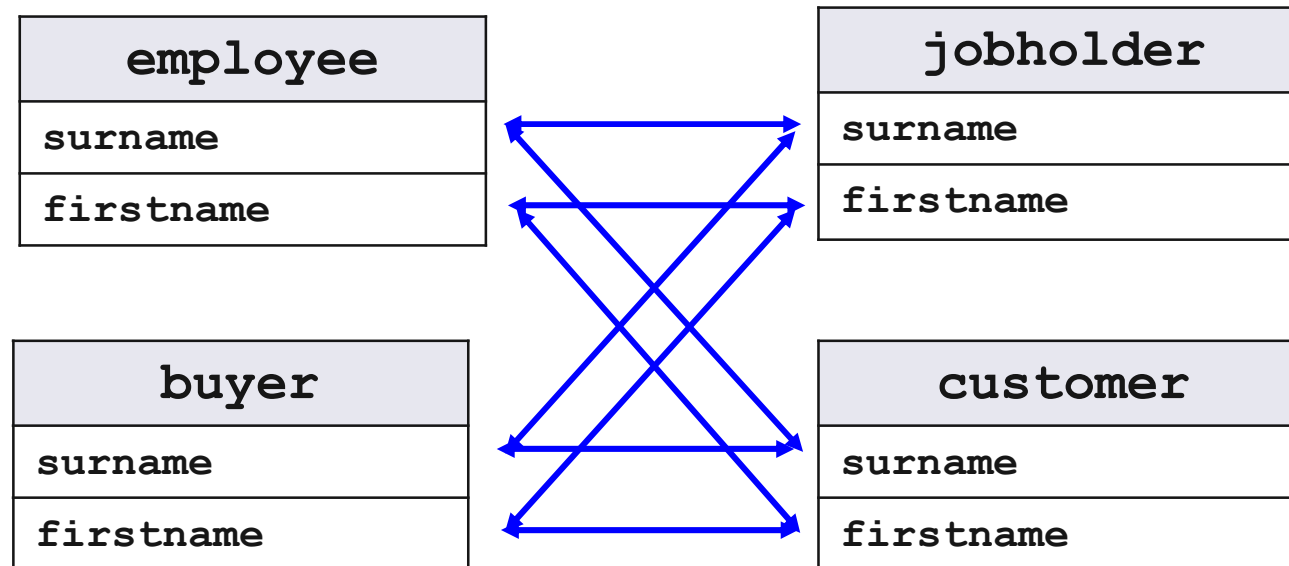
Source: Felix Naumann, ICIQ 2012 Talk

Discussion: Duplicate-based Methods

- Can correctly distinguish very similar attributes
 - Telephone number <> fax number, Surname<>Maiden name
- Work well if duplicates are known or easy to find
 - owl:sameAs statements in LOD cloud
 - shared IDs like GTINs, ISBNs, or GenIDs
- Does not work well if identity resolution is too noisy
 - e.g. products with very similar names

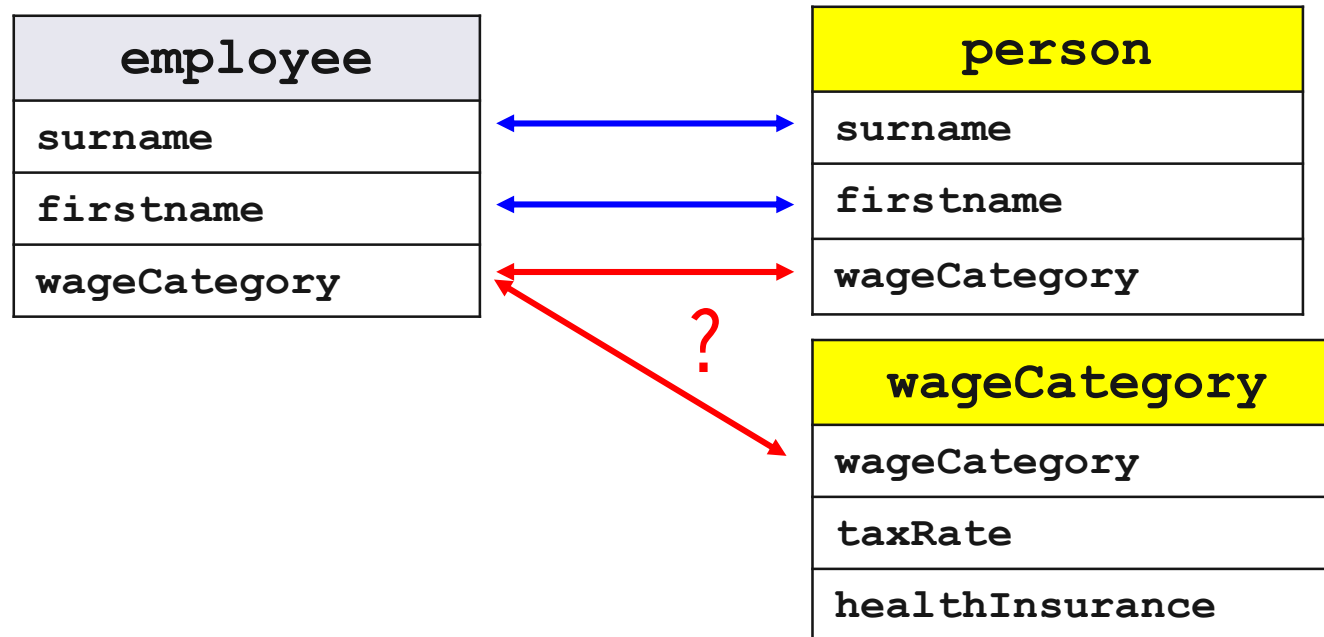
5.2.3 Structure-based Schema Matching Methods

- Addresses the following problem:



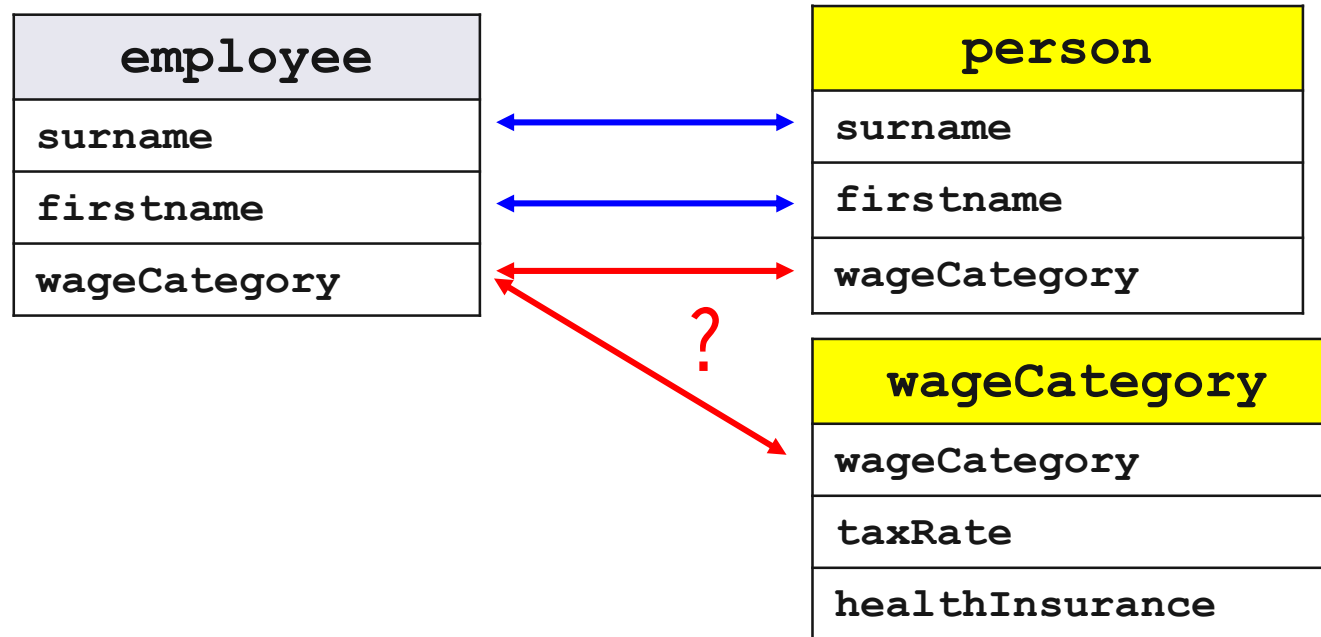
- Attribute-Attribute-Matching
 - Instance-based: Values of all attributes rather similar
 - Label-based: Labels of all attributes rather similar
 - All matchings are about equally good ☹️

Better approach: Exploit the Attribute Context



- Attributes that **co-occur** in one relation often (but not always) also co-occur in other relations.

Approach: Spread Similarity to Neighbors



- Idea: High similarity of neighboring attributes and/or name of relation increases similarity of attribute pair
- Base similarities: Label-based and/or instance-based matching
- Simple calculation: Weight attribute similarity with average similarity of all other attributes in same relation and similarity of relation names
- Alternative calculation: Similarity Flooding algorithm (see references)

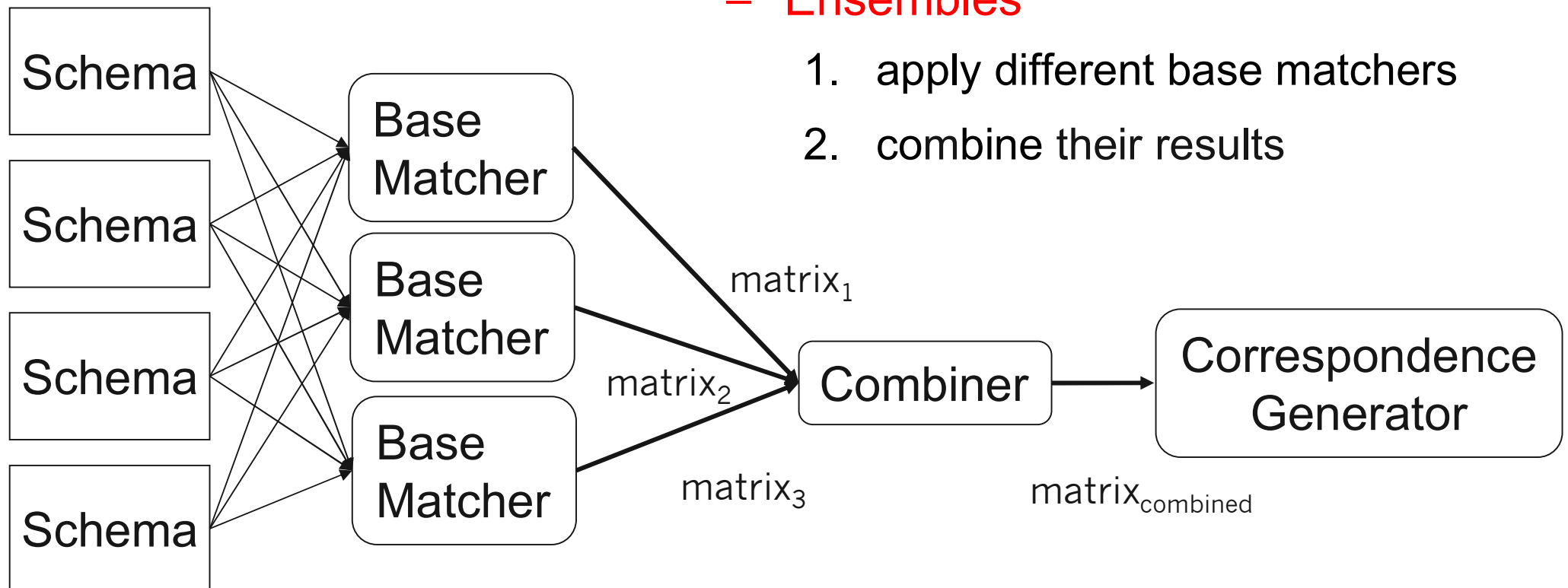
5.2.4 Combined Approaches

– Hybrid Approaches

- integrate different clues into single similarity function
- clues: labels, instance data, structure

– Ensembles

1. apply different base matchers
2. combine their results



Example of the Need to Exploit Multiple Types of Clues

realestate.com

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

- If we use only labels

- contact-agent matches either contact-name or contact-phone

homes.com

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

- If we use only data values

- contact-agent matches either contact-phone or office

- If we use both labels and data values

- contact-agent matches contact-phone

5.3 Generating Correspondences from the Similarity Matrix

Input: Matrix containing attribute similarities

Output: Set of correspondences

Correspondence
Generator

Local Single Attribute Strategies:

1. Thresholding

- all attribute pairs with sim above a threshold are returned as correspondences
- domain expert checks correspondences afterwards and selects the right ones

2. TopK

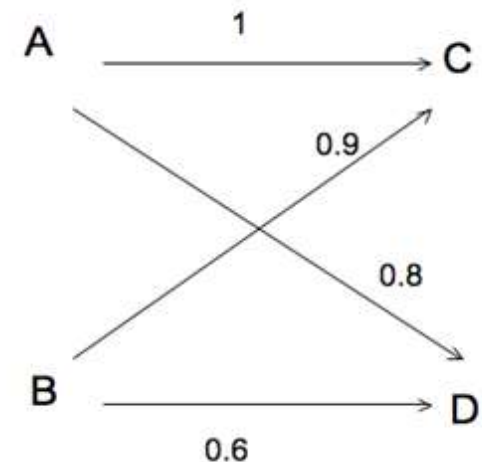
- give domain expert TopK correspondences for each attribute

3. Top1

- directly return the best match as correspondence (even if max similarity is low)
- very optimistic, errors might frustrate domain expert

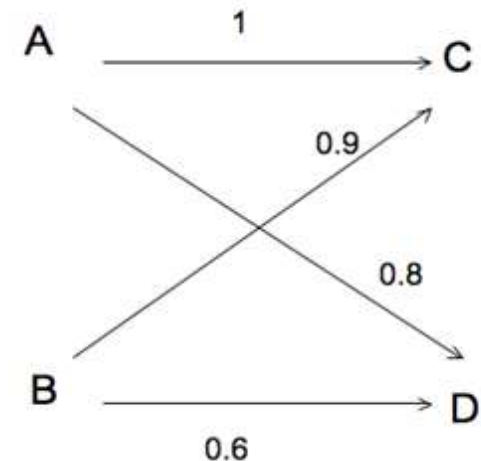
Alternative: Global Matching

- Looking at the complete mapping (all correct correspondences between A and B) gives us an additional restriction:
one attribute in A should only be matched to one attribute in B.
- Goal of **Global** Matching
 - Find optimal set of disjunct correspondences
 - avoid correspondence pairs of the form $A \equiv C$ and $B \equiv C$
- Approach:
 - find set of bipartite pairs with the maximal sum of their similarity values
- Example:
 - $A \equiv D$ and $B \equiv C$ have the maximal sum of their similarity values
 - Ignores that $\text{sim}(A, C) = 1$



Alternative: Stable Marriage

- Setting: Elements of A = women, elements of B = men
- $\text{Sim}(i,j)$ = degree to which A_i and B_j desire each other
- Goal: Find a stable match combination between men and women
- A match combination would be unstable if
 - there are two couples $A_i = B_j$ and $A_k = B_l$ such that A_i and B_l want to be with each other, i.e., $\text{sim}(i,l) > \text{sim}(i,j)$ and $\text{sim}(i,l) > \text{sim}(k,l)$
- Algorithm to find stable marriages
 - Let $\text{match} = \{\}$
 - Repeat
 - Let (i,j) be the highest value in sim such that A_i and B_j are not in match
 - Add $A_i = B_j$ to match
- Example: $A = C$ and $B = D$ form a stable marriage



PLM- and LLM-based Schema Matching

- use all three signals: column labels, data, table context
- combine similarity calculation, similarity aggregation, and correspondence generation into a single step
 - **PLM-based** (e.g. BERT): require a decent amount of task-specific training data, which is expensive to collect for schema matching
 - **LLM-based** (e.g. GPT): achieve good zero-shot performance ($F1 > 0.8$) using background knowledge instead of task-specific training data. Should be able to interpret dataset documentation such as data dictionaries
 - Example prompt (Table-GPT):

```
User: Please identify the matching columns between Table A and Table B.
For each column in Table A, specify the corresponding column in Table
B. If a column in A has no corresponding column in Table B, you can map
it to None. Represent each column mapping using a pair of column
headers in a list, i.e., [Table A Column, Table B column or None].
Return the final result as JSON in the format {"column_mappings": "<a
list of column pairs>"}
```

Zhang, et al.: Schema Matching using Pre-trained language models. ICDE, 2023 (PLM-based)

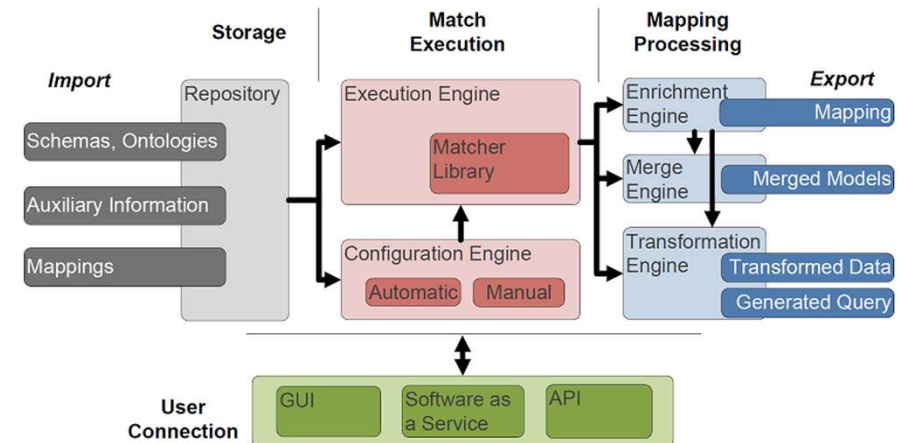
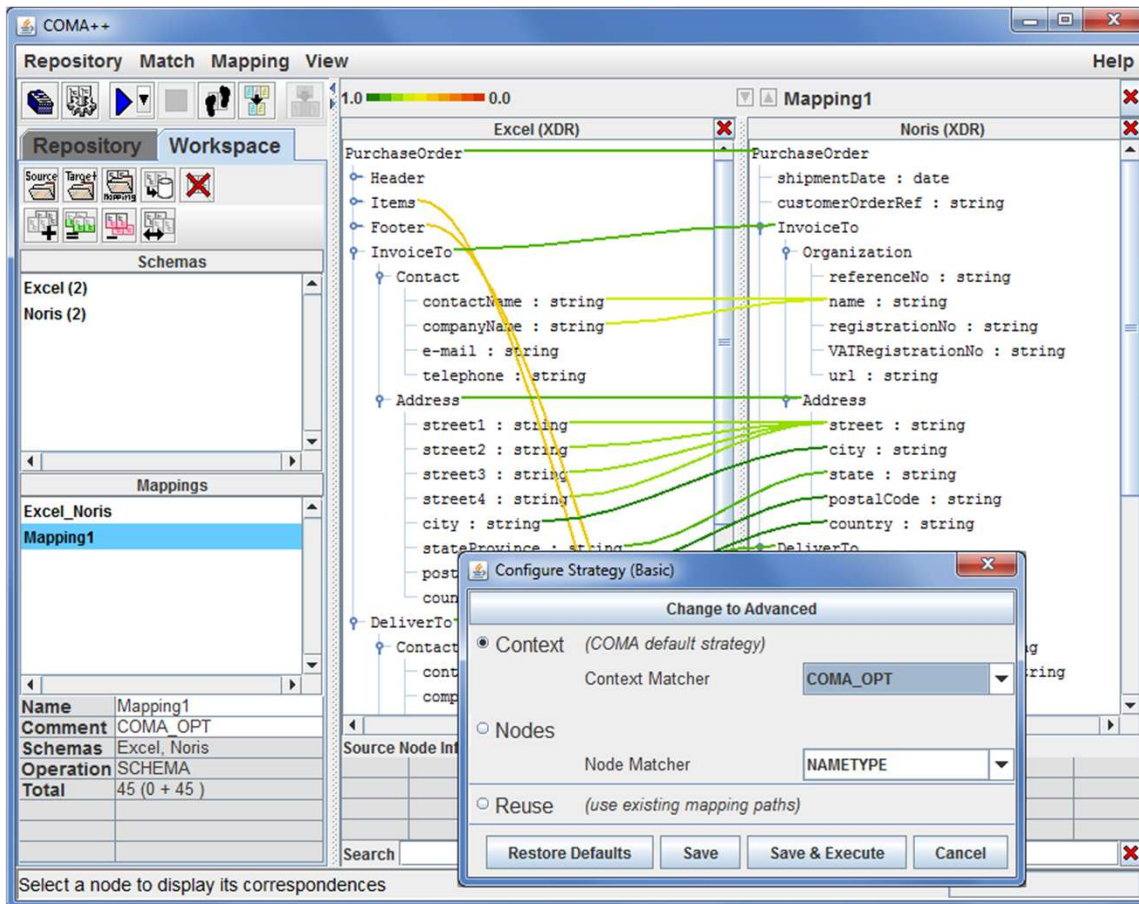
Li, et al.: Table-GPT: Table Fine-tuned GPT for Diverse Table Tasks. SIGMOD, 2024 (LLM-based)

Liu, et al.: Magneto: Combining Small and Large Language Models for Schema Matching. Arxiv, 2024 (PLM+LLM).

Example Matching System: COMA V3.0

Developed by the Database Group at the University of Leipzig

- provides wide variety of matchers (label, instance, structure, hybrid), no PLM/LLMs
- provides user interface for editing correspondences.
- provides data translation based on the correspondences.



<http://dbs.uni-leipzig.de/de/Research/coma.html>

5.5 Table Annotation

- Goal: Annotate the columns of tables in a **large table corpus** with concepts from a knowledge graph or shared vocabulary
 - use case: data lake indexing for data search (see slide set Data Discovery)
- Subtasks:
 - Column Type annotation: distance, weight, location, or person
 - Column Property annotation: proteinContent, fatContent, director, producer

 film	 director	 producer	 country
???	???	???	???
Happy Feet	George Miller, Warren Coleman, Judy Morris	Bill Miller, George Miller, Doug Mitchell	USA
Cars	John Lasseter, Joe Ranft	Darla K. Anderson	UK
Flushed Away	David Bowers, Sam Fell	Dick Clement, Ian La Frenais, Simon Nye	France

 **annotate**



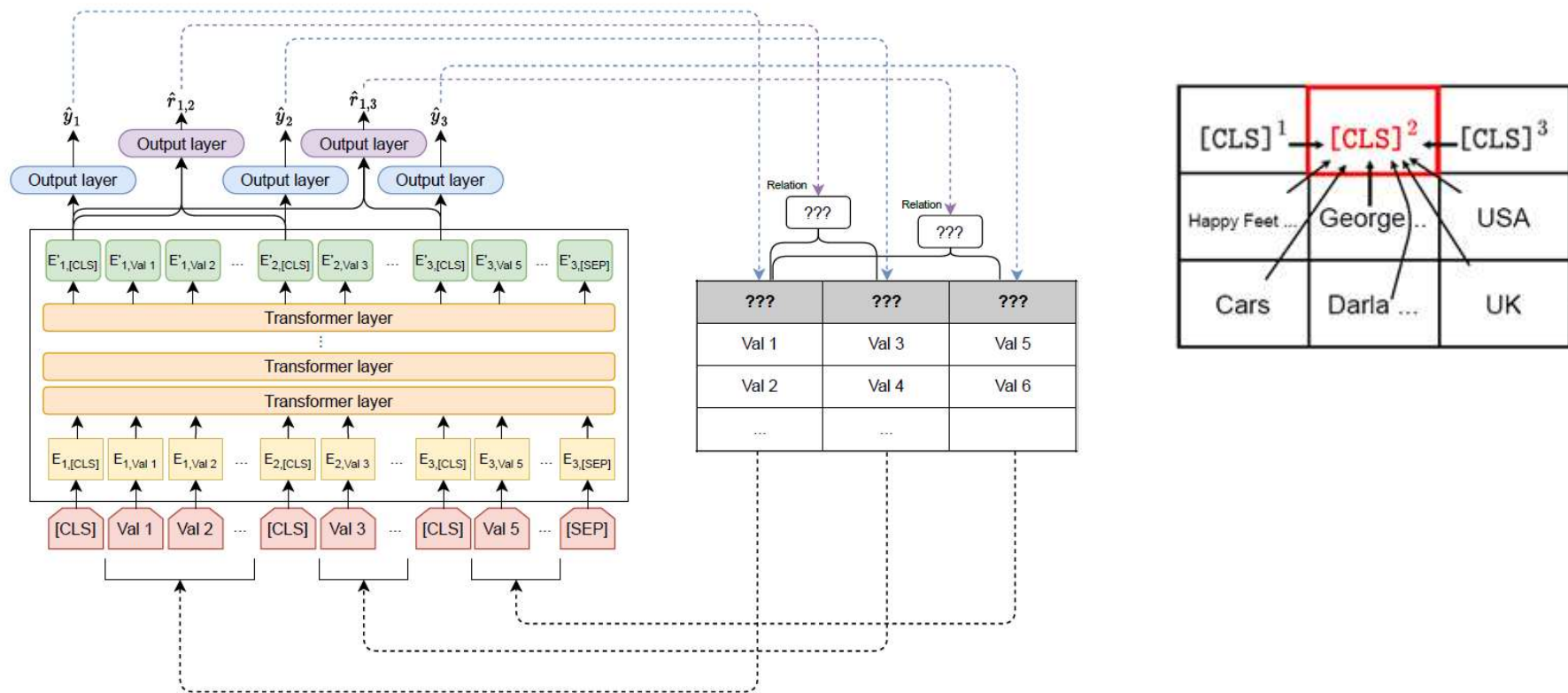
schema.org

SemTab evaluation campaign: <https://www.cs.ox.ac.uk/isg/challenges/sem-tab/>

Papers with Code: Table Annotation: <https://paperswithcode.com/task/table-annotation>

Example Table Annotation System: DoDuo

- directly fine-tunes BERT for column type and property annotation tasks using multi-task learning
- a table cell can pay attention to all neighboring cells



Suhara, et al.: Annotating Columns with Pre-trained Language Models. SIGMOD 2022.

Evaluation Results of Table Annotation Systems

- Column Type Annotation (~100 types)

Method	F1	P	R
TURL (TinyBERT)	88.86	90.54	87.23
DoDuo (BERT)	92.45	92.45	92.21

- Column Pair Annotation (Relation Extraction, ~100 relations)

Method	F1	P	R
TURL (TinyBERT)	90.94	91.18	90.69
DoDuo (BERT)	91.72	91.97	91.47

- The systems use lots of task-specific training data for fine-tuning
 - e.g. 10.000s of annotated tables

Suhara, et al.: Annotating columns with pre-trained language models. SIGMOD, 2022.
Deng, et al.: TURL: table understanding through representation learning. PVLDB 2020.

LLMs for Table Annotation

- use models like GPT4 or Llama3 **without fine-tuning**
- prompt triggers emergent capabilities and background knowledge
 - e.g. model knows in advance what restaurant names look like
- zero-shot performance of GPT4: **95%** F1 (32 types)

Prompt

Task: Classify the columns of a given table into one of the following classes: name of event, name of restaurant, postal code, region of address ... {32 semantic types are listed here}

Instructions: 1. Look at the input given to you and make a table out of it. 2. Look at the cell values in detail. 3. For each column, select a class that best represents the meaning of all cells in the column. 4. Answer with the selected class for each columns with the format Column1: class.

Table: Column 1 | Column 2 | Column 3 | Column 4 \n
Friends Pizza | 2525 | Cash Visa MasterCard | 7:30 AM \n ...

Korini and Bizer: Column Type Annotation using ChatGPT. TaDA Workshop@VLDB, 2023
<https://github.com/wbsg-uni-mannheim/TabAnnGPT>

5.7. Summary

- Schema Matching is an **active research area** with lots of approaches
 - yearly competitions: Ontology Alignment Evaluation Initiative (OAEI), SemTab
 - most recent approaches rely on PLMs and LLMs.
- Quality of discovered correspondences depends on difficulty of problem
 - many approaches work fine for single tables, but fail for larger schemas
- Thus, it is essential to **keep the domain expert in the loop**.
 - Active Learning
 - learn from user feedback while searching for correspondences
 - Crowd Sourcing
 - mechanical turk
 - DBpedia Mapping Wiki
 - Spread the manual integration effort over time
 - pay-as-you-go integration in data lakes

6. Schema Heterogeneity on the Web

1. Role of Standards

1. RDFa/Microdata/Microformats
2. Linked Data

2. Self-Descriptive Data on the Web

6.1 Role of Standards

For publishing data on the Web, various communities try to **avoid schema-level heterogeneity** by agreeing on standard schemata (also called vocabularies or ontologies).

- **schema.org**

- 600+ Types: Event, local business, product, review, person, place, ...



- **JSON Schema Store**

- 900+ schemas: locations, CVs, github, AWS, Minecraft
<https://www.schemastore.org/json/>

- **Linked Data**

- various widely used vocabularies
- FOAF, SKOS, Music Ontology, ...



Properties used to Describe Schema.org Products

Two million websites (PLDs) annotate product offers.

Attribute	% of PLDs
schema:Product/name	99 %
schema:Product/offers	94 %
schema:Offer/price	95 %
schema:Offer/priceCurrency	95 %
schema:Product/description	84 %
schema:Offer/availability	72 %
schema:Product/sku	56 %
schema:Product/brand	30 %
schema:Product/image	26 %
schema:Product/aggregateRating	17 %
schema:Product/mpn	6.3 %
schema:Product/productID	4.7 %
...	...

← New Samsung Galaxy S4 GT-19505 16GB 5.0 inches Android Smartphone with 2-Year Sprint Contract - White Frost

← 99.00 US\$

← The Galaxy S4 is among the earliest phones to feature a 1080p Full HD display. The various connectivity options on the Samsung include ...

← 000214632623

← Samsung

← GT-19505

← 000214632623



http://webdatacommons.org/structureddata/2020-12/stats/schema_org_subsets.html

Vocabularies in the LOD Cloud

Data sources **mix terms** from commonly used and proprietary vocabularies.

- Idea
 - Use common, easy-to-understand vocabularies wherever possible.
 - Define proprietary terms only if no common terms exist.
- LOD Cloud Statistics 2014
 - 271 (41.76%) are non-proprietary, 378 (58.24%) proprietary vocabularies
- Common Vocabularies

Vocabulary	Number of Datasets
foaf	701 (69.13%)
dcterms	568 (56.02%)
sioc	179 (17.65%)
skos	143 (14.10%)
void	137 (13.51%)
cube	114 (11.24%)

Source:
<http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/>

6.2 Self-Descriptive Data

Data sources in the LOD context try to increase the usefulness of their data and ease data integration by making it self-descriptive.

Aspects of self-descriptiveness

1. Reuse **terms from common vocabularies / ontologies**
2. Enable clients to **retrieve the schema**
3. Properly **document terms**
4. **Publish correspondences** on the Web
5. Provide provenance metadata
6. Provide licensing metadata

Specific realization of **FAIR data principles** (see slide set on Data Discovery)

Reuse Terms from Common Vocabularies

1. Common Vocabularies

- **Schema.org** for describing various types of entities
- **Friend-of-a-Friend** for describing people and their social network
- **Dublin Core** and **DCAT** provide terms for representing dataset metadata
- **SIOC** for describing forums and blogs
- **SKOS** for representing topic taxonomies
- **Organization Ontology** for describing the structure of organizations
- **Music Ontology** for describing artists, albums, and performances

2. Common sources of identifiers (URIs) for real world objects

- **XML Schema** for datatypes
- **LinkedGeoData** and **Geonames** locations
- **DBpedia** and **Wikidata** wide range of things
- **GeneID** and **UniProt** life science identifiers

Enable Clients to Retrieve the Schema

Clients can resolve the URIs that identify vocabulary terms in order to get their RDFS, OWL, XML schema definitions.

Some data on the Web

```
<http://richard.cyganiak.de/foaf.rdf#cygri>  
  foaf:name "Richard Cyganiak" ;  
  rdf:type <http://xmlns.com/foaf/0.1/Person> .
```

Resolve unknown term

<http://xmlns.com/foaf/0.1/Person>



RDFS or OWL definition

```
<http://xmlns.com/foaf/0.1/Person>  
  rdf:type owl:Class ;  
  rdfs:label "Person";  
  rdfs:subClassOf <http://xmlns.com/foaf/0.1/Agent> ;  
  rdfs:subClassOf <http://xmlns.com/wordnet/1.6/Agent> .
```

Documentation of Vocabulary Terms

The documentation of a vocabulary is published on the Web in machine-readable form and can be used as a clue for schema matching.

- Name of a vocabulary term
 - ex1:name **rdfs:label** "A person's name"@en .
 - ex1:name **rdfs:label** „Der Name einer Person"@de .
- Additional description of the term
 - ex1:name **rdfs:comment**
"Usual order: family name, given name"@en .
- Informal specifications originally intended for humans can increasingly be used by LLMs.

FOAF Vocabulary Specification 0.99

Namespace Document 14 January 2014 - *Paddington Edition*

This version:

<http://xmlns.com/foaf/spec/20140114.html> (rdf)

Latest version:

<http://xmlns.com/foaf/spec/> (rdf)

Previous version:

<http://xmlns.com/foaf/spec/20100809.html> (rdf)

Authors:

[Dan Brickley](#), [Libby Miller](#)

Contributors:

Members of the FOAF mailing list (foaf-dev@lists.foaf-project.org) and the wider [RDF and Semantic Web developer community](#). See [acknowledgements](#).

Copyright © 2000-2014 Dan Brickley and Libby Miller

This work is licensed under a [Creative Commons Attribution License](#).
This copyright applies to the *FOAF Vocabulary Specification* and accompanying documentation in RDF. Regarding underlying technology, FOAF uses W3C's [RDF](#) technology, an open Web standard that can be freely used by anyone.

Abstract

This specification describes the FOAF language, defined as a dictionary of named properties and classes using W3C's RDF technology.

FOAF is a project devoted to linking people and information using the Web. Regardless of whether information is in people's heads, in physical or digital documents, or in the form of factual data, it can be linked. FOAF integrates three kinds of network: *social networks* of human collaboration, friendship and association; *representational networks* that describe a simplified view of a cartoon universe in factual terms, and *information networks* that use Web-based linking to share independently published descriptions of this inter-connected world. FOAF does not compete with socially-oriented Web sites; rather it provides an approach in which different sites can tell different parts of the larger story, and by which users can retain some control over their information in a non-proprietary format.

<http://xmlns.com/foaf/spec/>

Publish Correspondences on the Web

Vocabularies are (partly) connected via vocabulary links.

Vocabulary Link

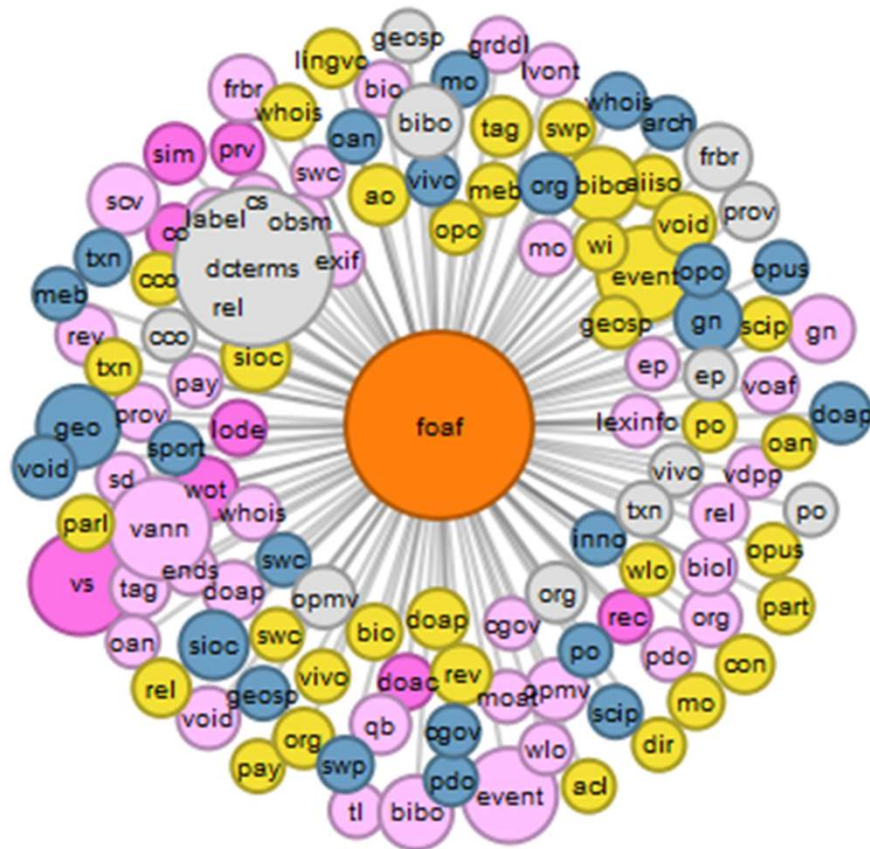
```
<http://dbpedia.org/ontology/Person>  
owl:equivalentClass  
<http://xmlns.com/foaf/0.1/Person> .
```

- Terms for representing correspondences
 - owl:equivalentClass, owl:equivalentProperty,
 - rdfs:subClassOf, rdfs:subPropertyOf
 - skos:broadMatch, skos:narrowMatch

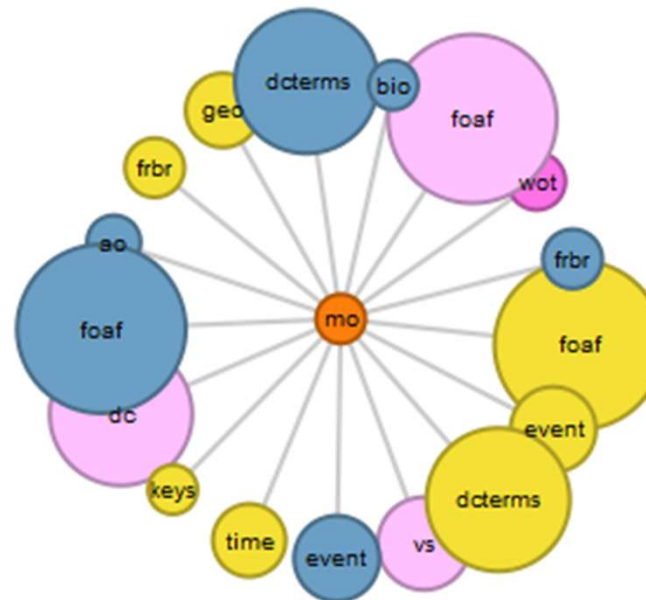
Deployment of Vocabulary Links

Vocabulary links:

Vocabularies referencing "foaf" (119)



Vocabularies referenced by "mo" (17)

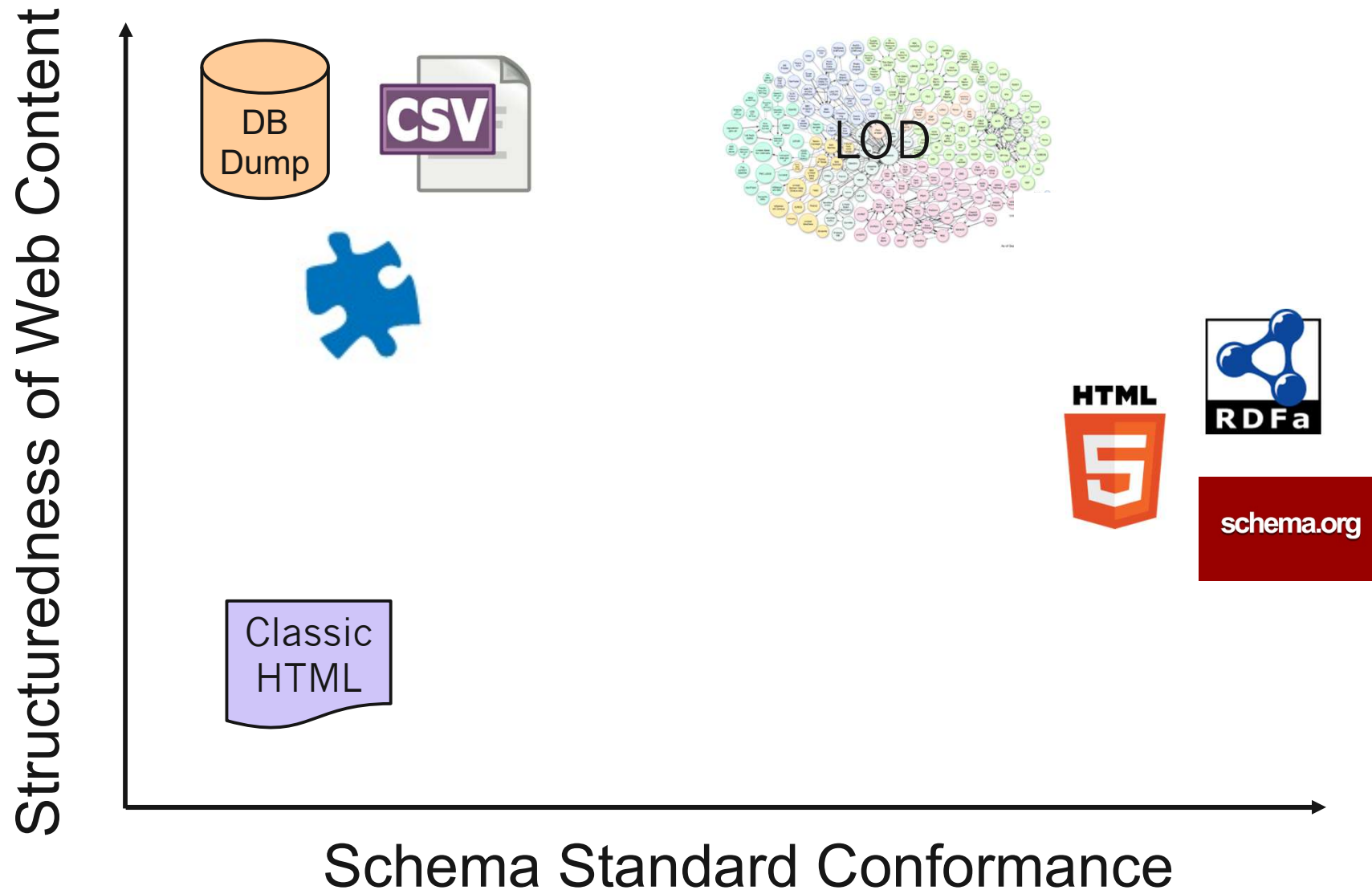


Source: Linked Open Vocabularies,
<https://lov.linkeddata.es/dataset/lov/>

Link types

- Similar to
- Used by
- Relies on
- Metadata vocabulary
- Extends
- Specializes
- Generalizes
- Has equivalences with
- Has disjunctions with

Summary: Structuredness and Standard Conformance



7. References

- Schema Integration
 - Leser, Naumann: Informationsintegration. Chapter 5.1, dpunkt Verlag, 2007.
 - Spaccapietra, et al.: Model Independent Assertions for Integration of Heterogeneous Schemas. VLDB, 1992.
- Data Translation
 - Doan, Halevy, Ives: Principles of Data Integration. Chapter 5.10, Morgan Kaufmann, 2012.
 - Leser, Naumann: Informationsintegration. Chapter 5.2, DBunkt Verlag, 2007.
- Schema Matching
 - Doan, Halevy, Ives: Principles of Data Integration. Chapter 5, Morgan Kaufmann, 2012.
 - Dong, Srivastava: Big Data Integration. Chapter 2. Morgan & Claypool Publishers, 2015.
 - Euzenat, Shvaiko: Ontology Matching. Springer, 2013.
 - Doan: iMAP: Discovering complex semantic Matches between Database Schemas. SIGMOD, 2004.
 - Rahm, Madhavan, Bernstein: Generic Schema Matching, Ten Years Later. VLDB, 2011.
 - Rinser, Lange, Naumann: Cross-lingual Entity Matching and Infobox Alignment in Wikipedia. Information Systems (IS) 38(6):887–907, 2013.
 - Zhang et al.: Schema Matching using Pre-Trained Language Models. ICDE 2023.

References

- Table Annotation
 - Suhara, et al.: Annotating columns with pre-trained language models. SIGMOD, 2022.
 - Deng, et al.: TURL: table understanding through representation learning. PVLDB 2020.
 - Korini and Bizer: Column Type Annotation using ChatGPT. TaDA @VLDB, 2023.
- Data Spaces and Data Lakes
 - Hai, Rihan, Christoph Quix, and Matthias Jarke: Data lake concept and systems: a survey. arXiv:2106.09592 (2021).
 - Sawadogo, Pegdwendé, and Jérôme Darmont: On data lake architectures and metadata management. Journal of Intelligent Information Systems, 2021.
 - Franklin, M., Halevy, A., and Maier, D.: From Databases to Dataspaces A new Abstraction for Information Management. SIGMOD Rec., 2005.
- Schema Standardization on the Web
 - Bizer, et al.: Deployment of RDFa, Microdata, and Microformats on the Web – A Quantitative Analysis. 12th International Semantic Web Conference, 2013.
 - Schmachtenberg, et al: Adoption of the linked data best practices in different topical domains. 13th International Semantic Web Conference, 2014.