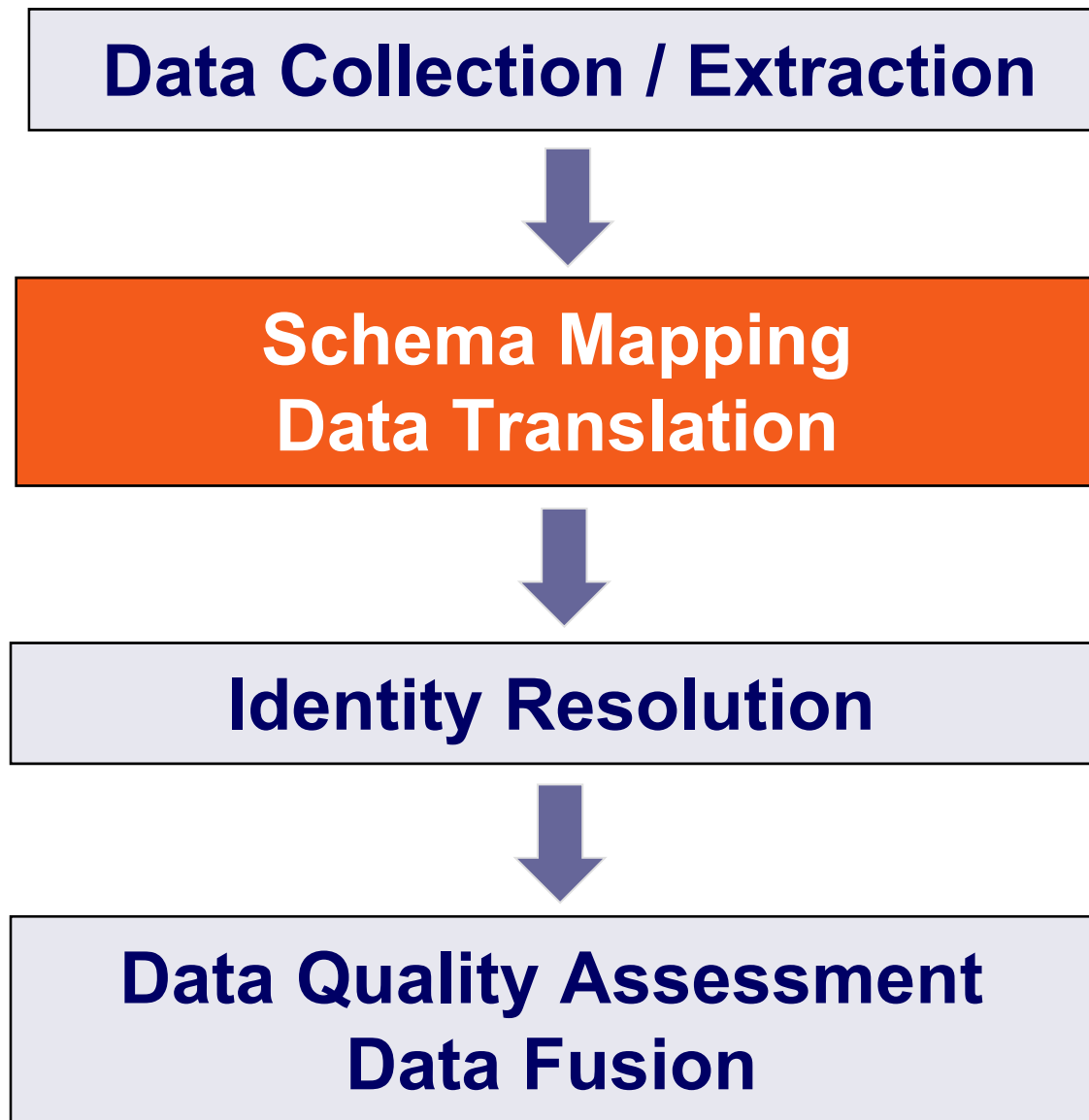


Web Data Integration

Schema Mapping and Data Translation



The Data Integration Process

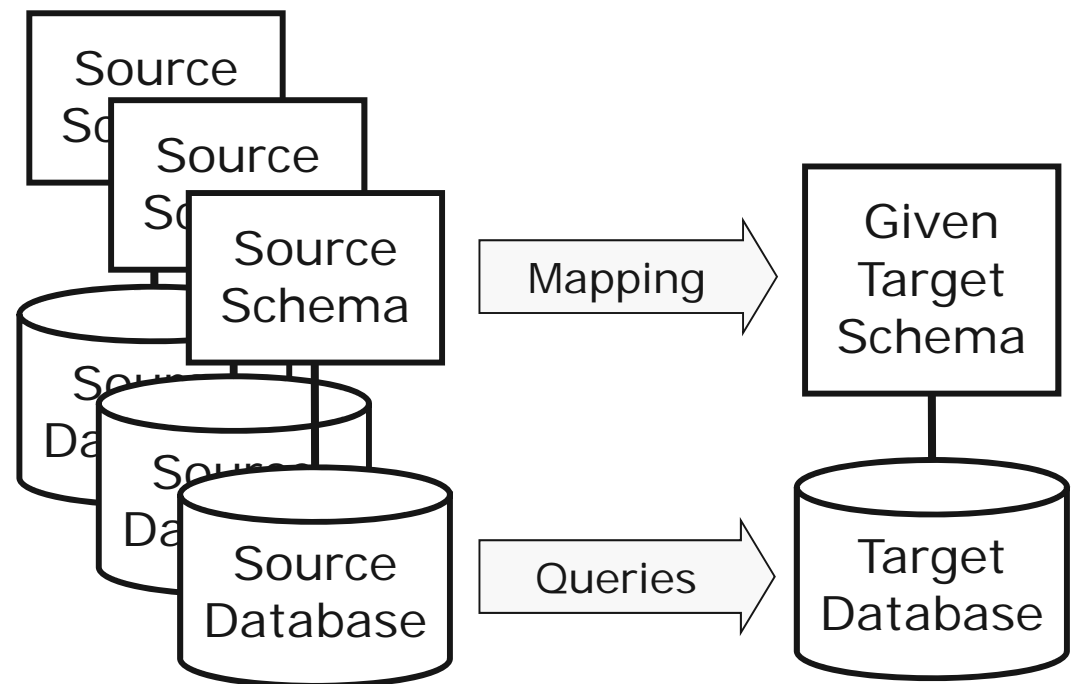


1. Two Basic Integration Situations
2. Types of Correspondences
3. Schema Integration
4. Data Translation
5. Schema Matching
6. Schema Heterogeneity on the Web

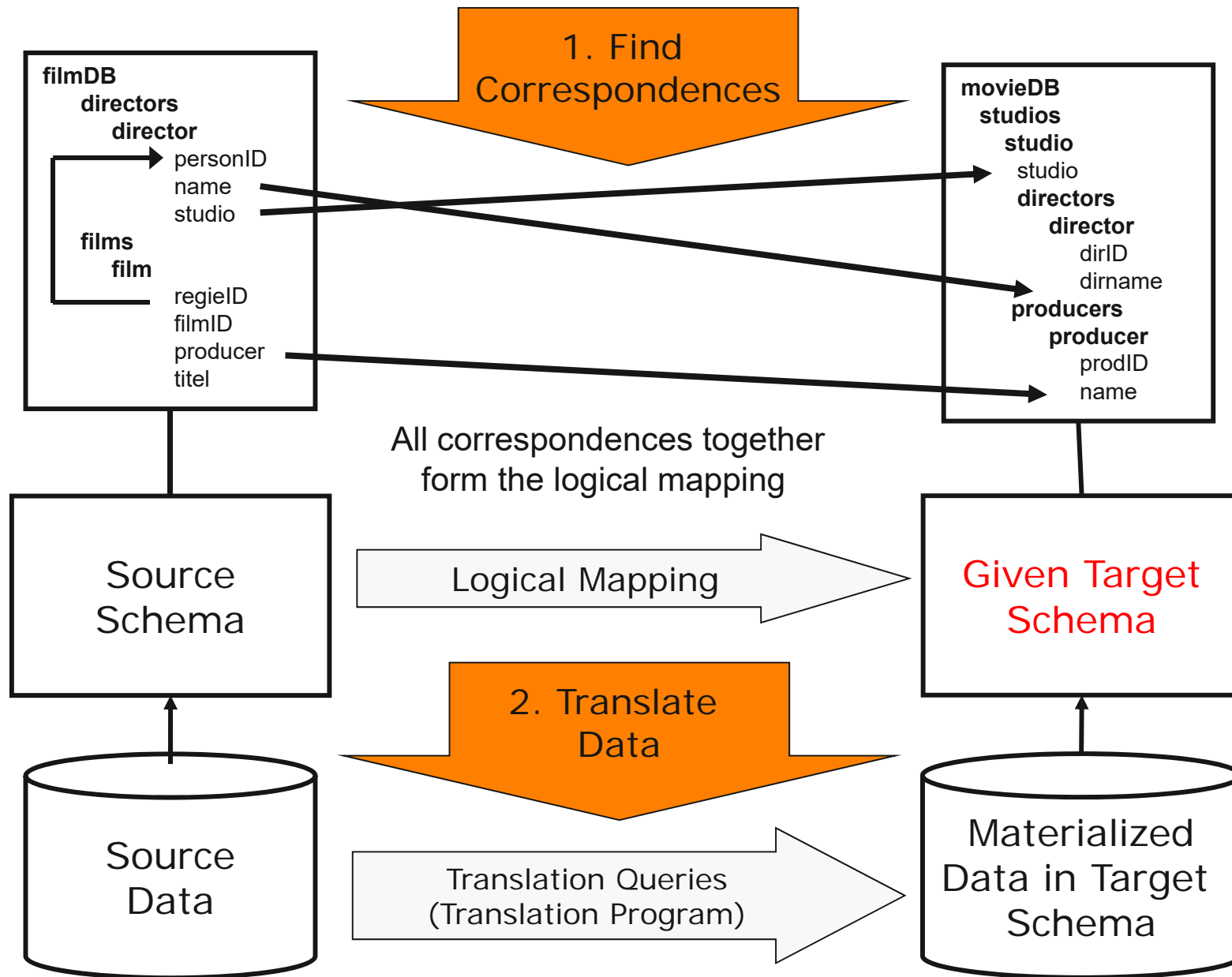
Basic Integration Situation 1: Schema Mapping

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)



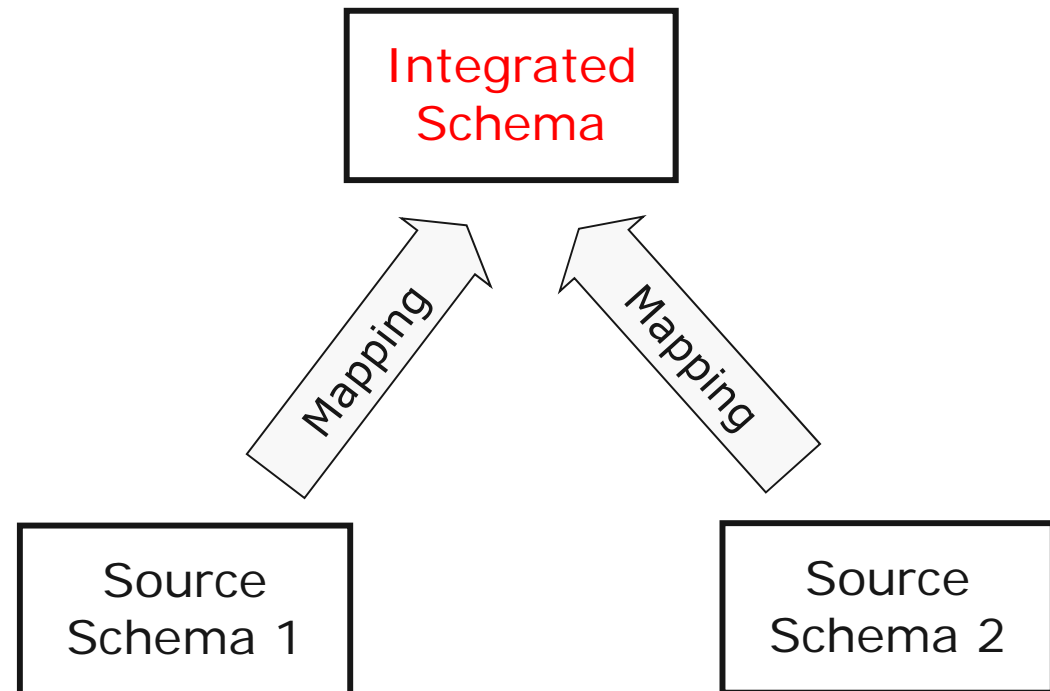
The Schema Mapping Process



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**.

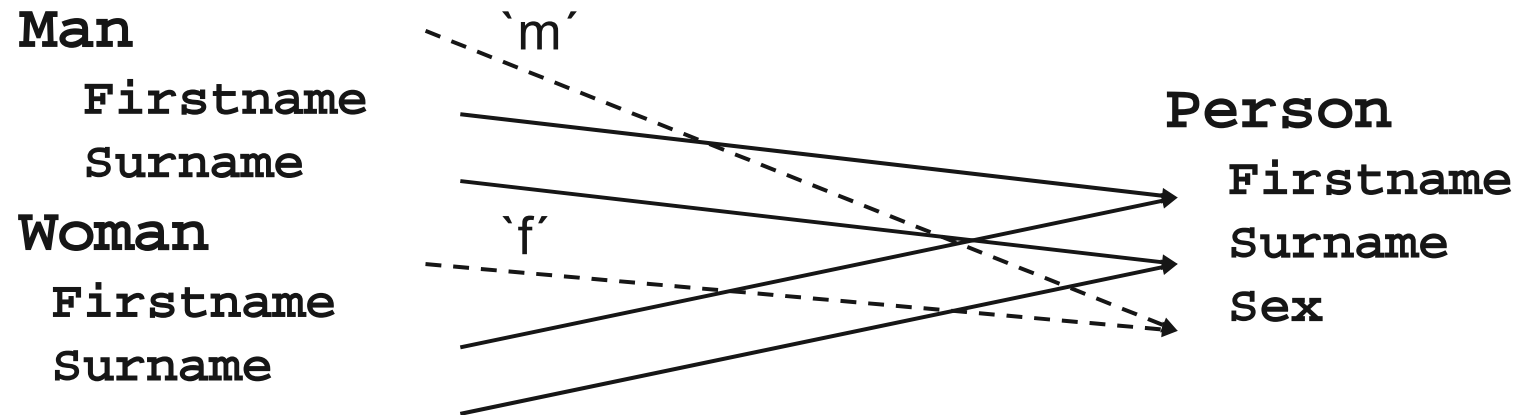
- **Mapping** = Set of all correspondences that relate S and T
- Correspondences are **easier to specify** than transformation queries
 - domain expert does not need technical knowledge about query language
 - specification can be supported by user interfaces (mapping editors)
 - step-by-step process 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)
 - cast attribute data types (integer to real)
 - translate values using a translation table (area code to city name)

Types of Correspondences

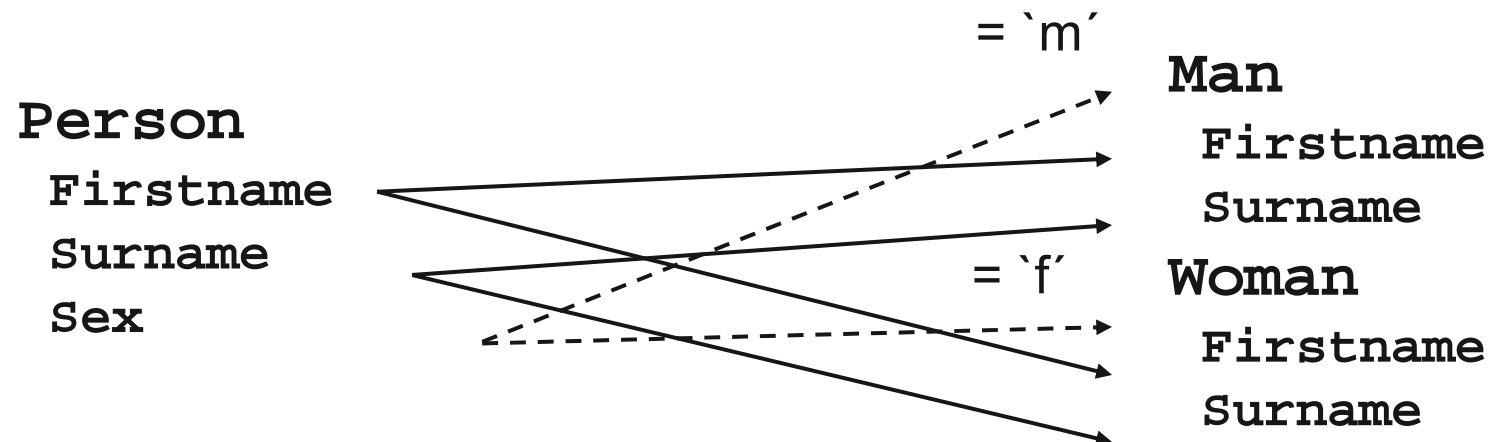
- **One-to-One Correspondences**
 - `Movie.title` \rightarrow `Item.name`
 - `Product.rating` \rightarrow `Item.classification`
 - `Movie` \equiv `Film` (equivalence: Same semantic intention)
 - `Athlete` \subseteq `Person` (inclusion: All athletes are also persons)
- **One-to-Many Correspondences**
 - `Person.Name` \rightarrow `split()` \rightarrow `FirstName` (Token 1)
 \rightarrow `Surname` (Token 2)
- **Many-to-One Correspondences**
 - `Product.basePrice` * $(1 + \text{Location.taxRate})$ \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

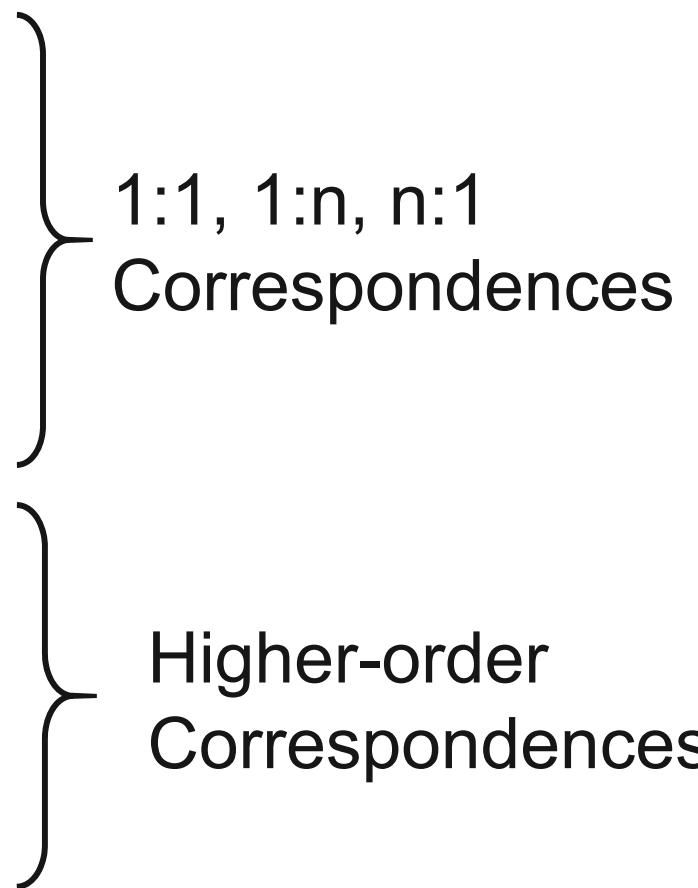
Relation-to-Value Correspondences



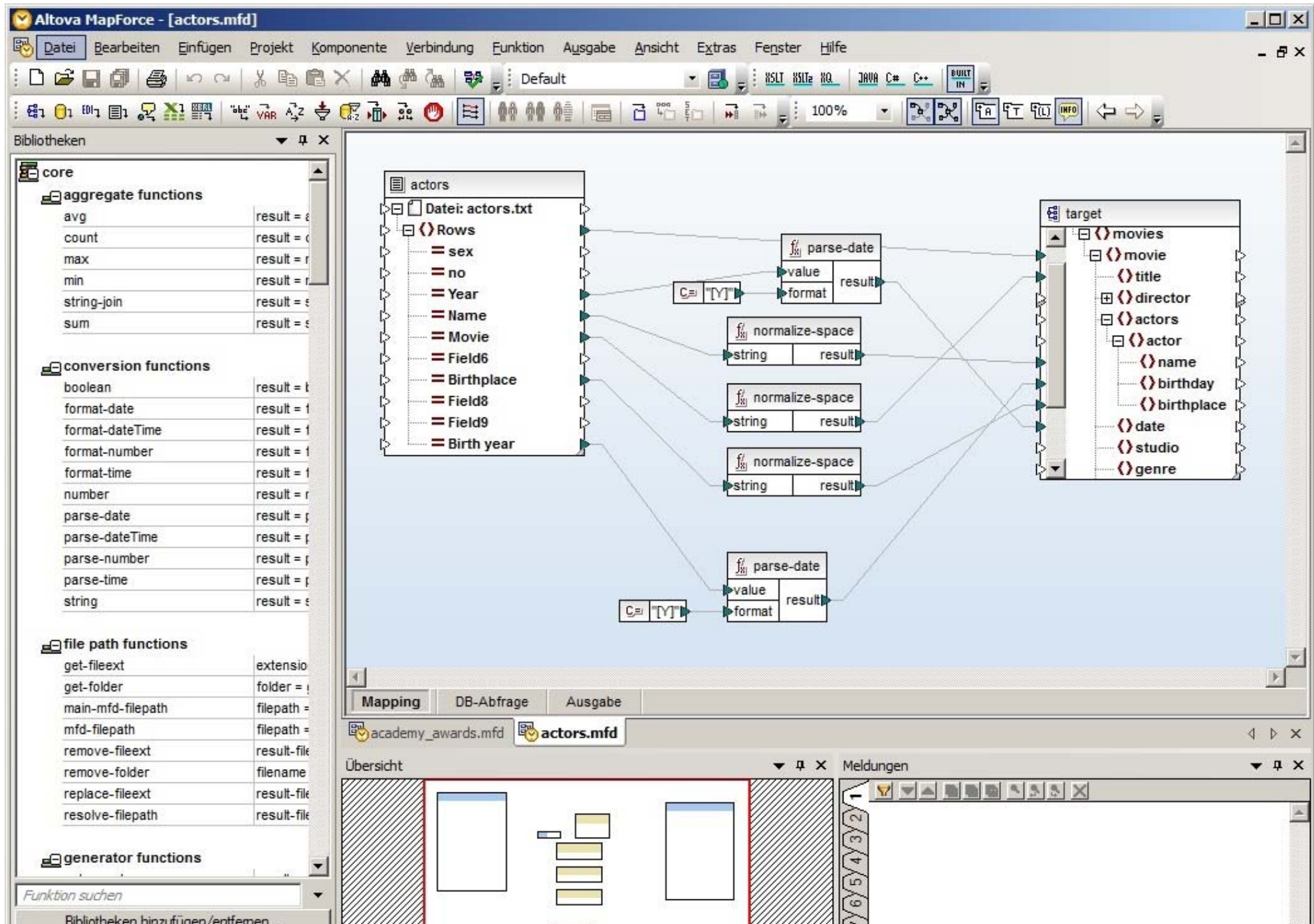
Value-to-Relation Correspondences



Types of Schema Heterogeneity that can be Captured

- Naming of
 - Relations
 - Attributes
 - Normalized vs. Denormalized
 - Nesting vs. Foreign Keys
 - 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

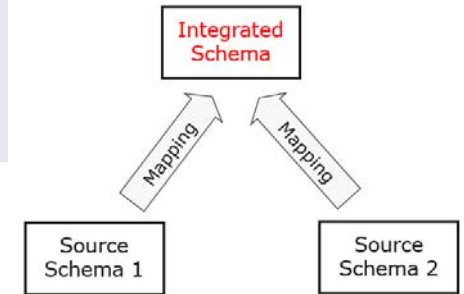
Schema Matching: Automatically or semi-automatically discover correspondences between schemata.



- Various schema matching methods exist (we will cover them later)
- Automatically finding a complete high-quality mapping is not possible in most real-world cases. Halevy: „It’s plain hard.“ :-(
- In practice, schema matching is often used to create candidate correspondences that are verified by human experts afterwards
- Realistic goals
 1. use matching to reduce the effort required from domain experts **or**
 2. be prepared to tolerate some noise in the generated mapping

3. Schema Integration

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

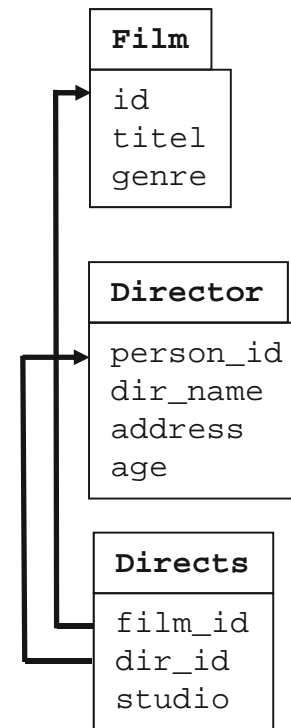


- Goals:
 - **Completeness:** All elements of the source schemata should be covered
 - **Correctness:** All data should be represented semantically correct
 - cardinalities, integrity constraints, ...
 - **Minimality:** The integrated schema should be minimal in respect to the number of relations and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

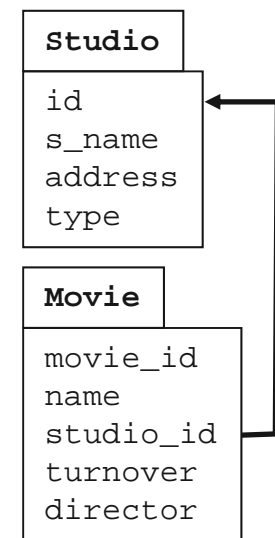
Example: Two Schemata about Films

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



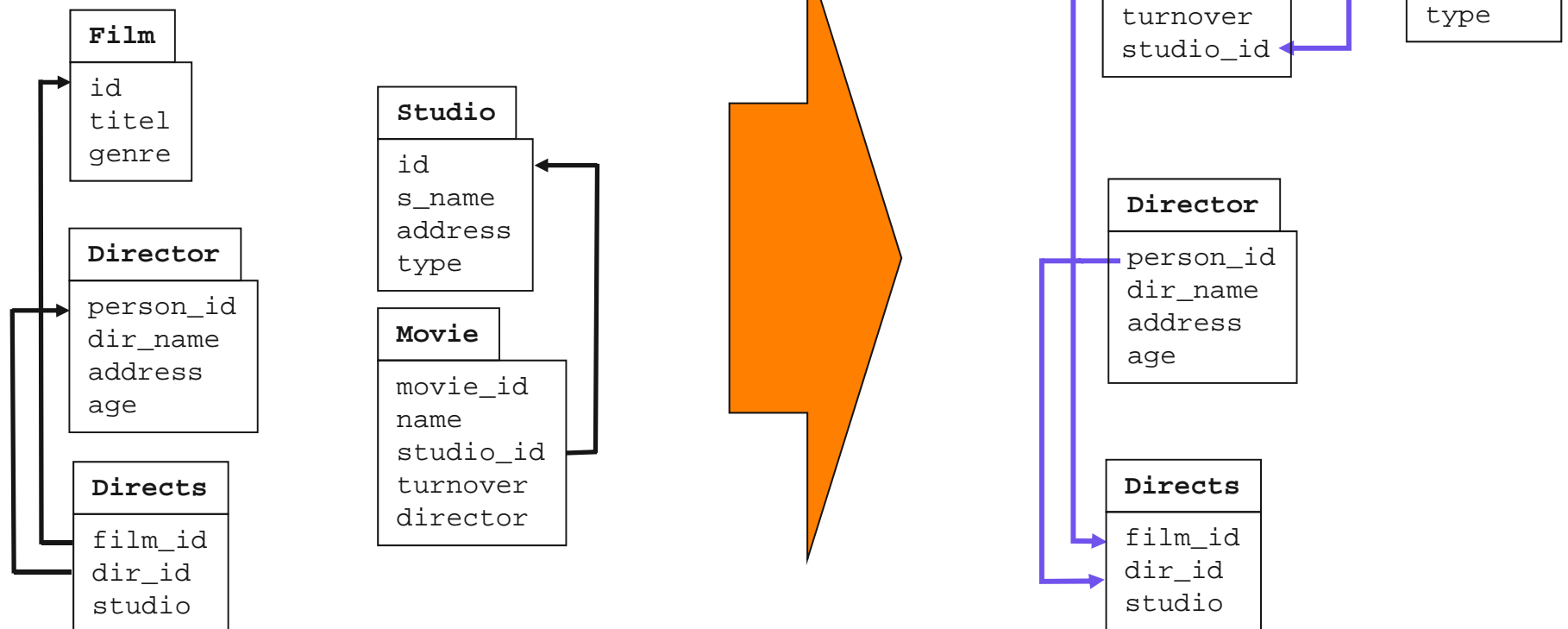
N:M
Relationship



Normalized
Schema

Schema Integration: Rules of Thumb

1. Merge all tables with equivalent tables in other schema (Film, Movie)
2. Add all tables without equivalent 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**

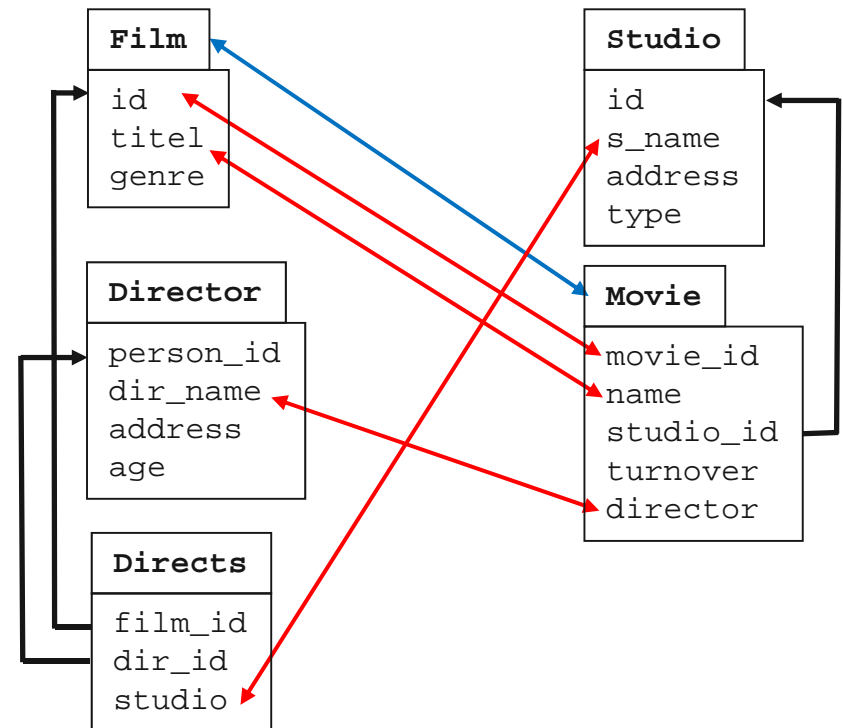
- Attribute Correspondences

id \equiv **movie_id**

titel \equiv **name**

dir_name \equiv **director**

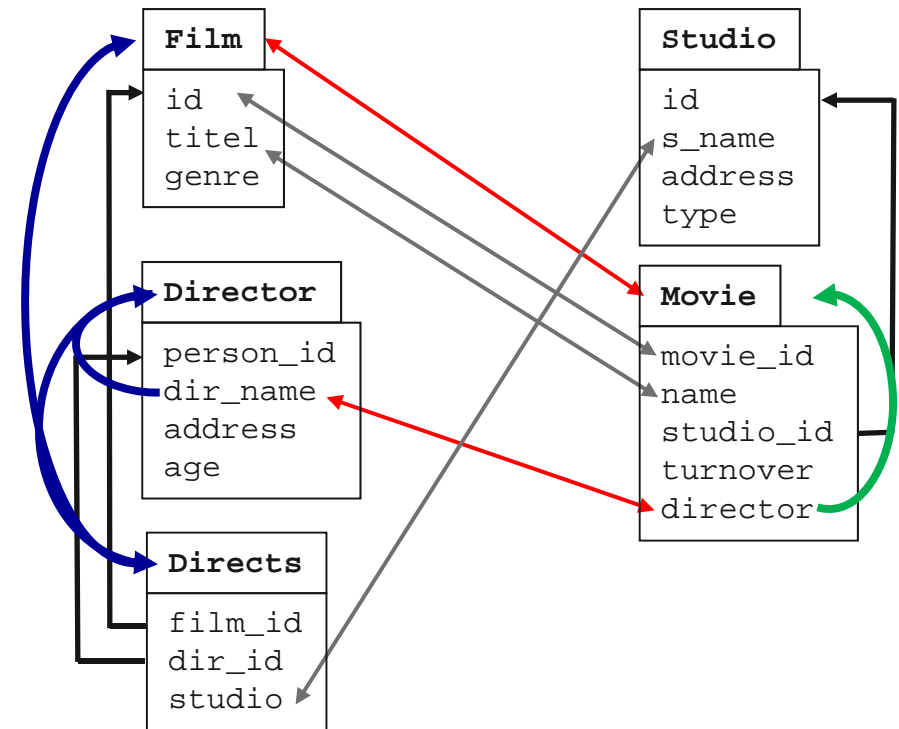
studio \equiv **s_name**



Example: Relationship Path Correspondence 1

- Relationship Path Correspondence

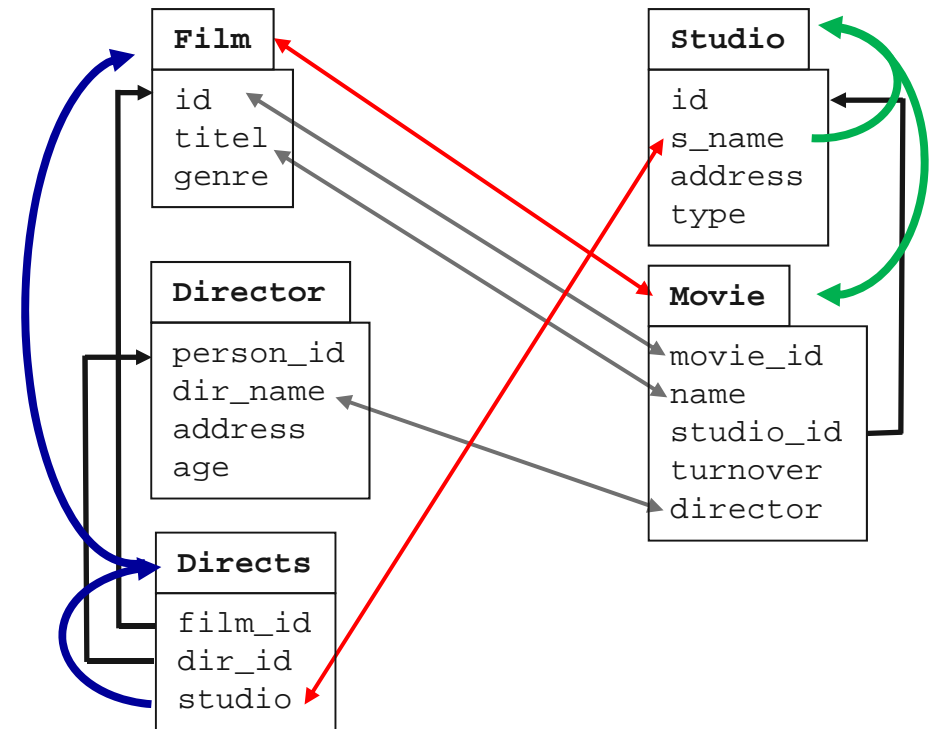
dir_name-Director-Directs-Film \equiv **director-Movie**



Example: Relationship Path Correspondence 2

- Relationship Path Correspondence

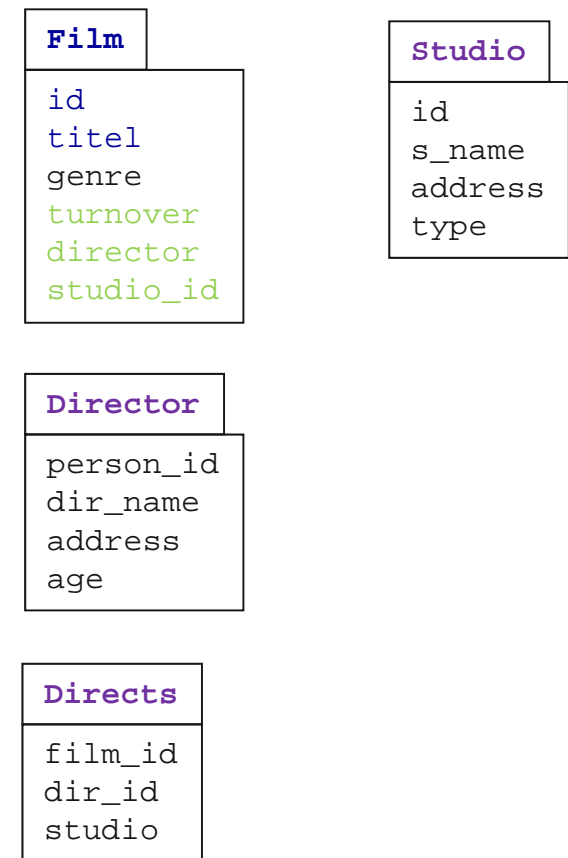
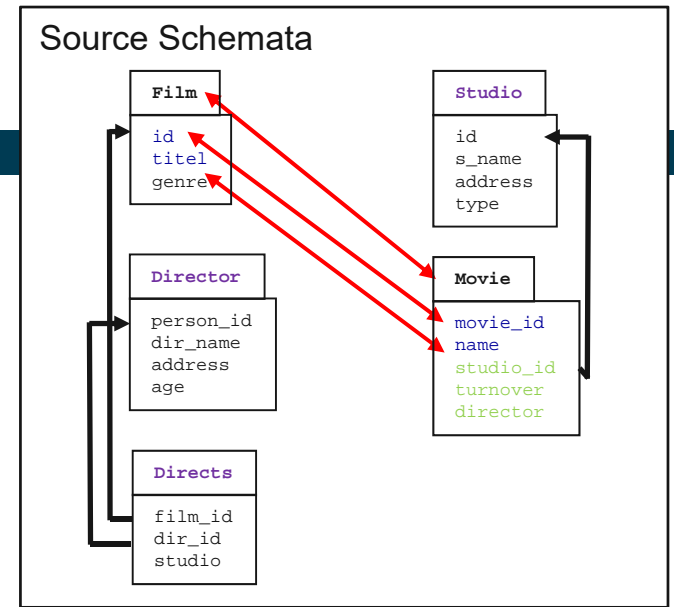
studio-Directs-Film \equiv **s_name-Studio-Movie**



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**)



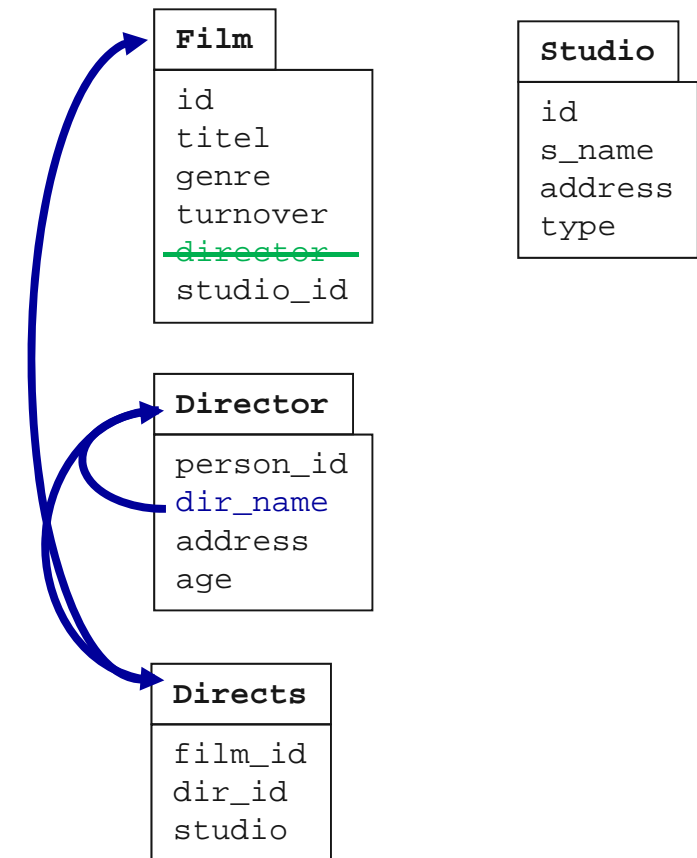
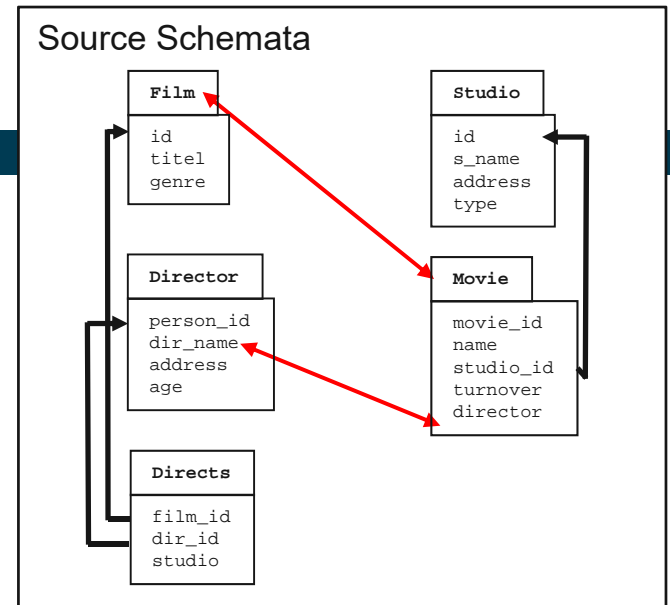
Creation of the Integrated Schema 2

– Correspondence

- `dir_name-Director-Directs-Film` = `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

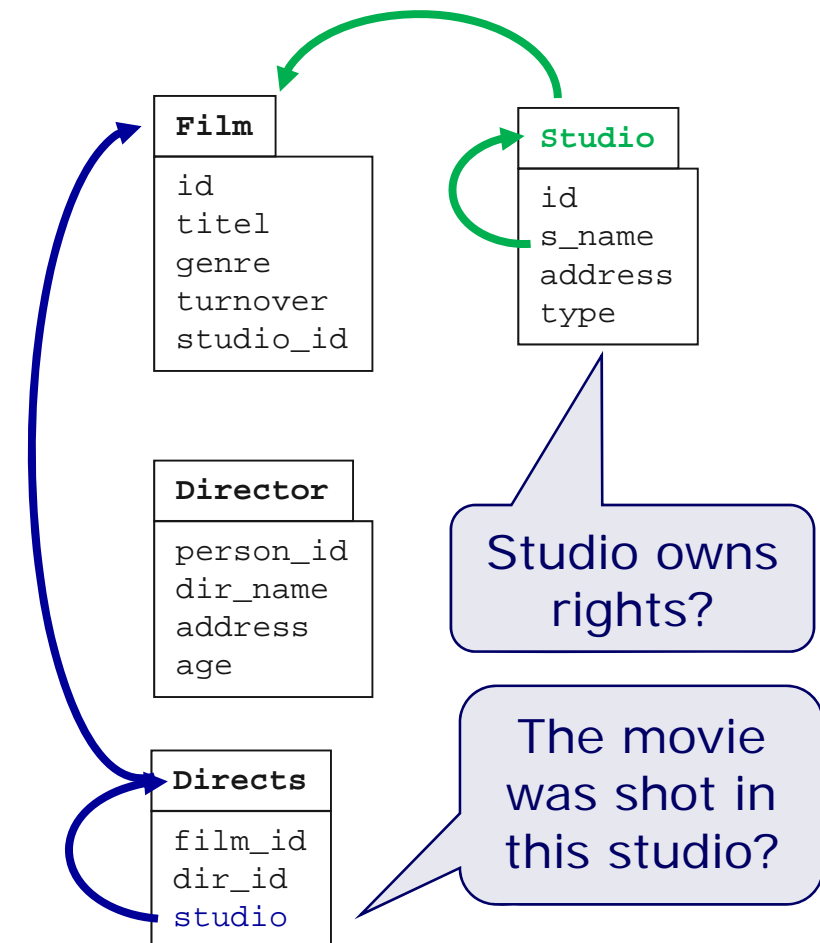
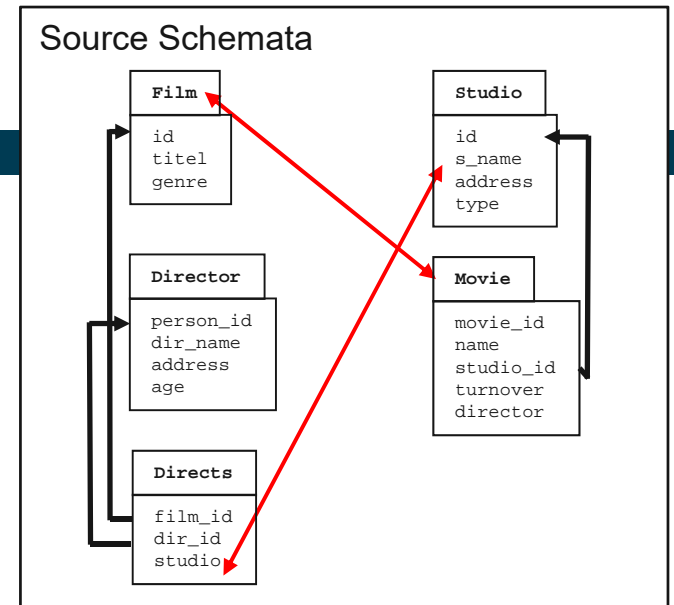
– Correspondence

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

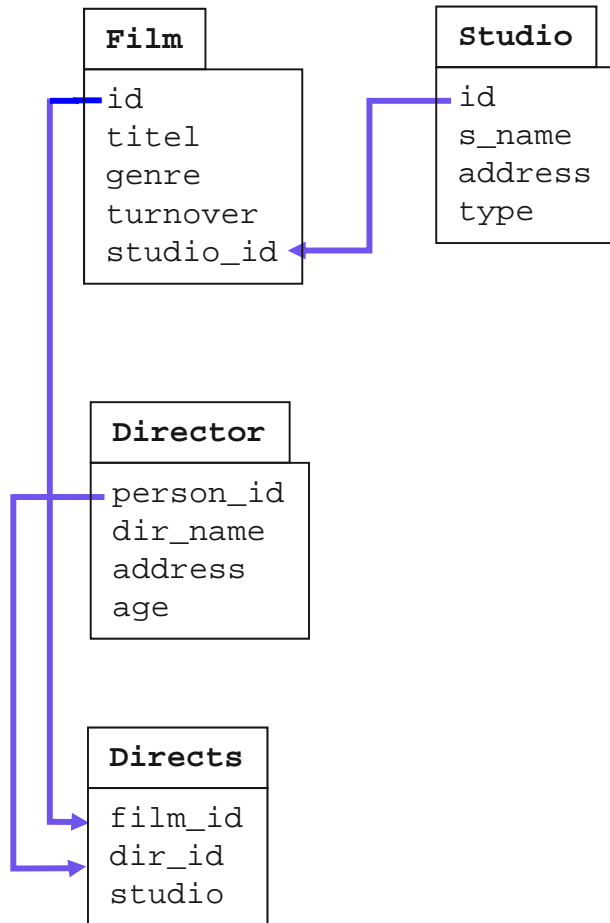
– Integration Step

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.



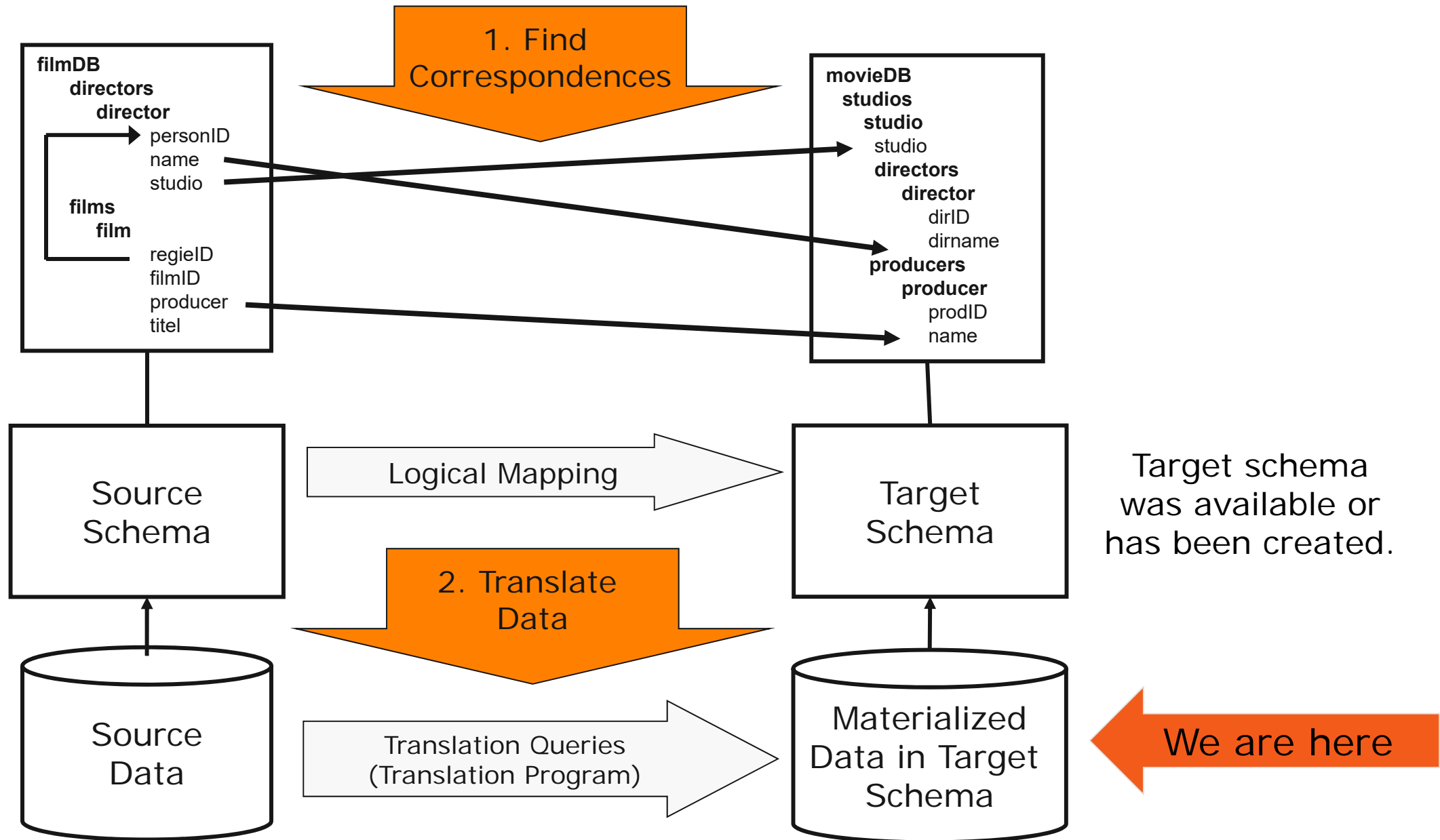
Final Integrated Schema



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 in order to overcome **different levels of normalization**
 - Might require combining data from multiple source tables (**horizontal partitioning**)

Normalized → Denormalized

ARTICLE

- artPK
- title
- pages

PUBLICATION

- pubID
- title
- date
- author

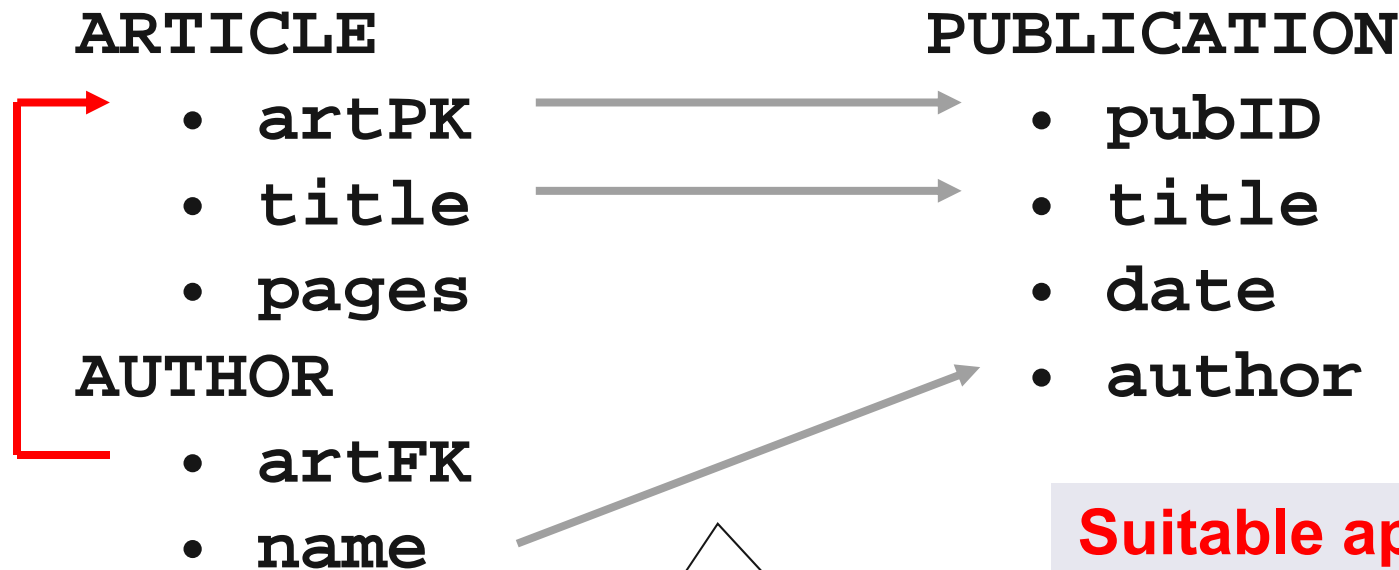
AUTHOR

- artFK
- name

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
```

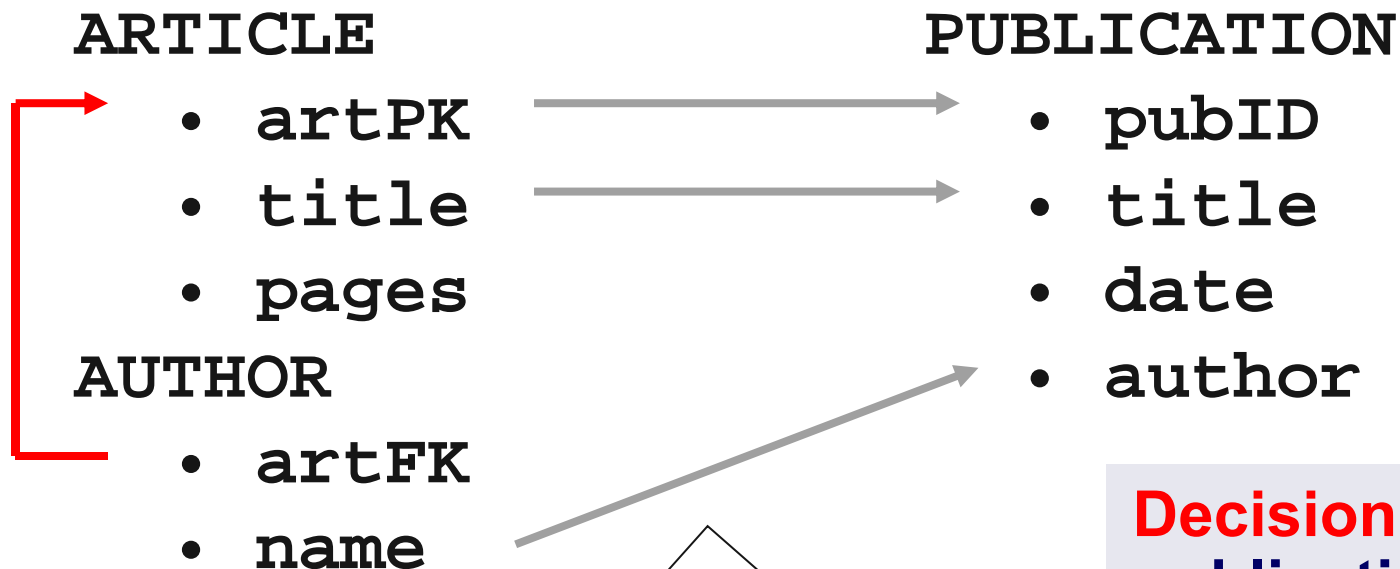
Normalized → Denormalized



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
```

INNER JOIN vs. OUTER JOIN



Decision: Do we want publications without author?

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


Denormalized → Normalized

PUBLICATION

- title
- date
- author

ARTICLE

- artPK
- title
- pages

AUTHOR

- artFK
- name

DISTINCT

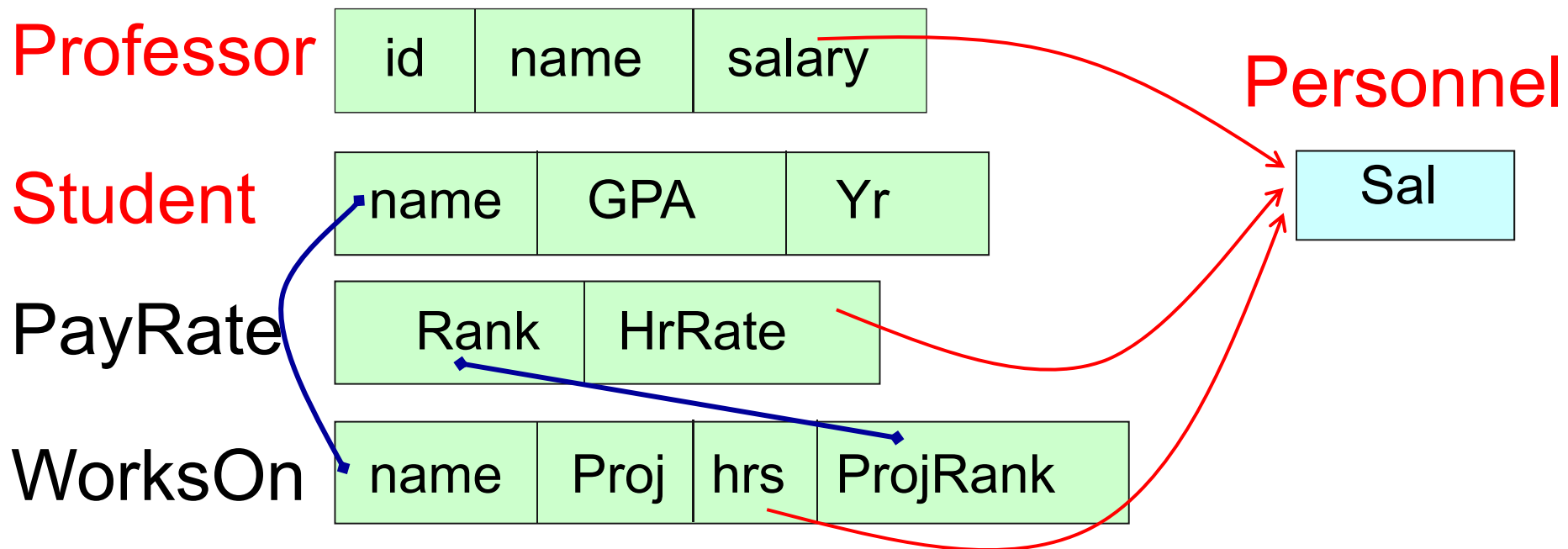
```
SELECT SK(title) AS artPK
       title AS title
       null AS pages
INTO   ARTICLE
FROM   PUBLICATION
```

```
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

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



Correspondence 1: Professor.salary → Personnel.Sal

Correspondence 2: PayRate.HrRate * WorksOn.Hrs → Personnel.Sal

UNION the Salaries of Professors and Students

Correspondence 1: Professor.salary \rightarrow Personnel.Sal

Correspondence 2: PayRate.HrRate * WorksOn.Hrs \rightarrow 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
```

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

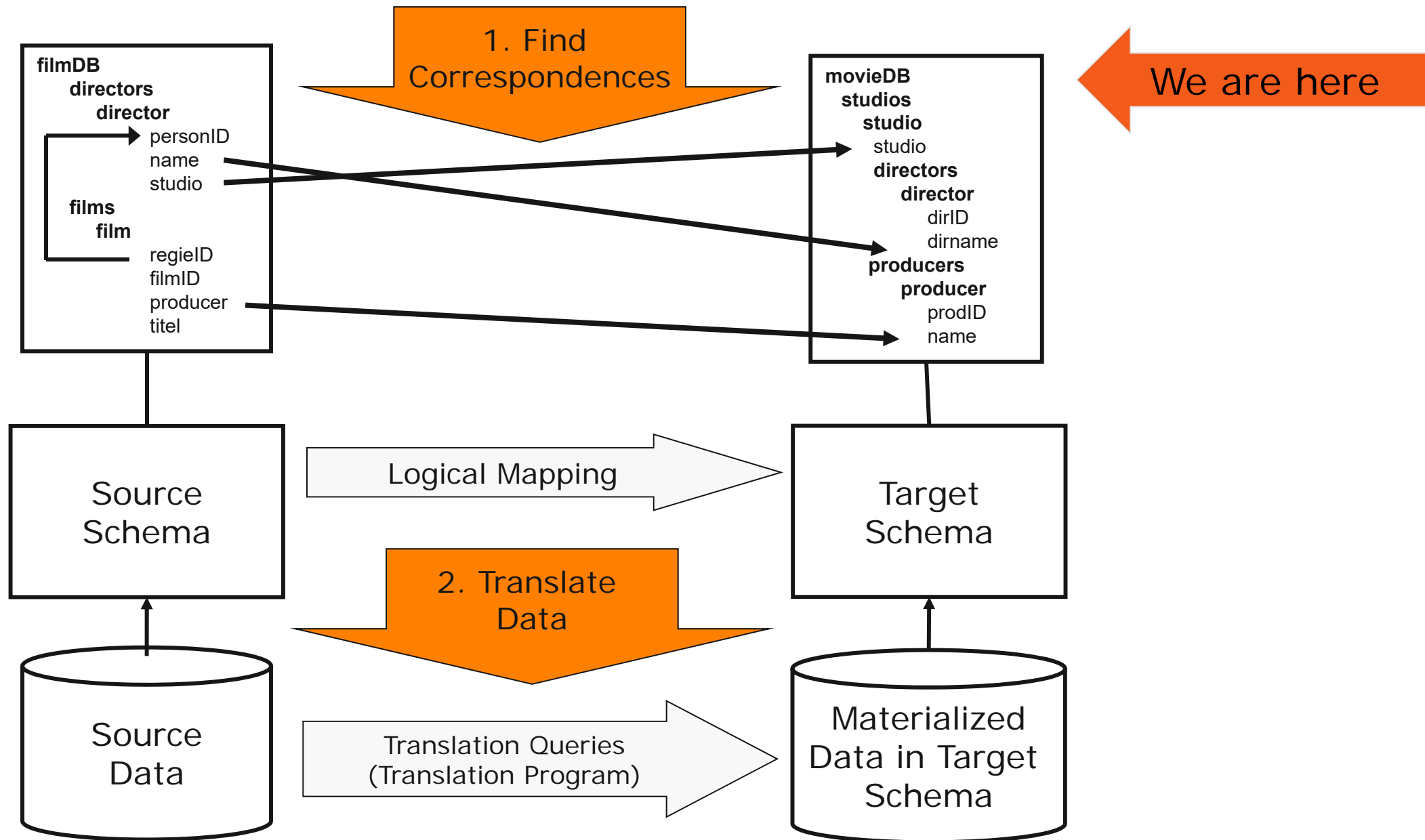
5. Schema Matching

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



- Automatically finding a complete high-quality mapping (= set of all correspondences) is difficult in many real-world cases
- In practice, schema matching is used to create **candidate correspondences** that are verified by domain experts afterwards
- Most schema matching methods focus on **1:1 correspondences**
 - we restrict ourselves to 1:1 for now and speak about 1:n and n:1 later.

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. Finding One-to-Many and Many-to-One Correspondences
5. Example Schema Matching System
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

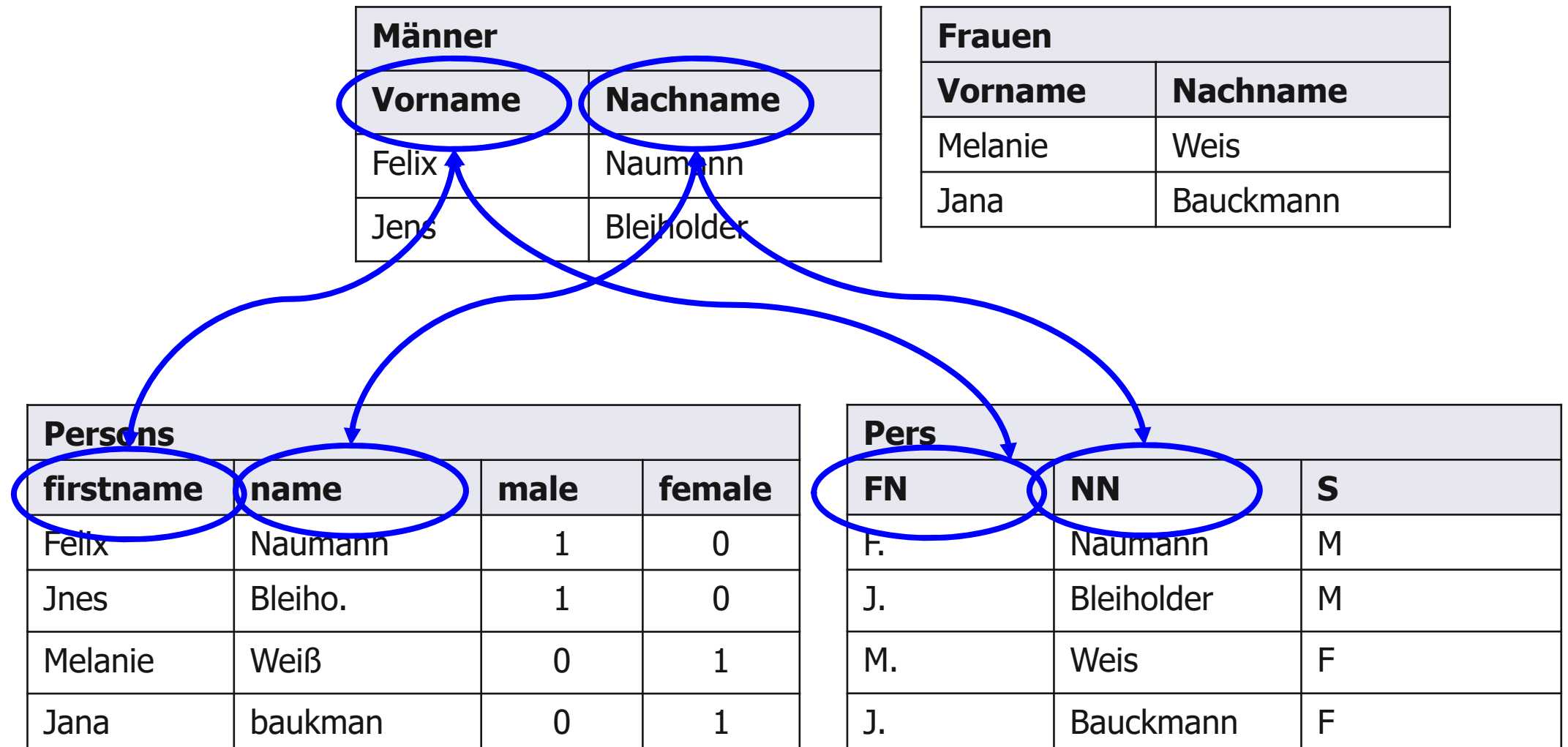
- attribute1, attribute2, attribute3
(was used as names for product features in Amazon API)

4. Semantic heterogeneity

- synonyms, homonyms, ...

5. Missing documentation

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

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

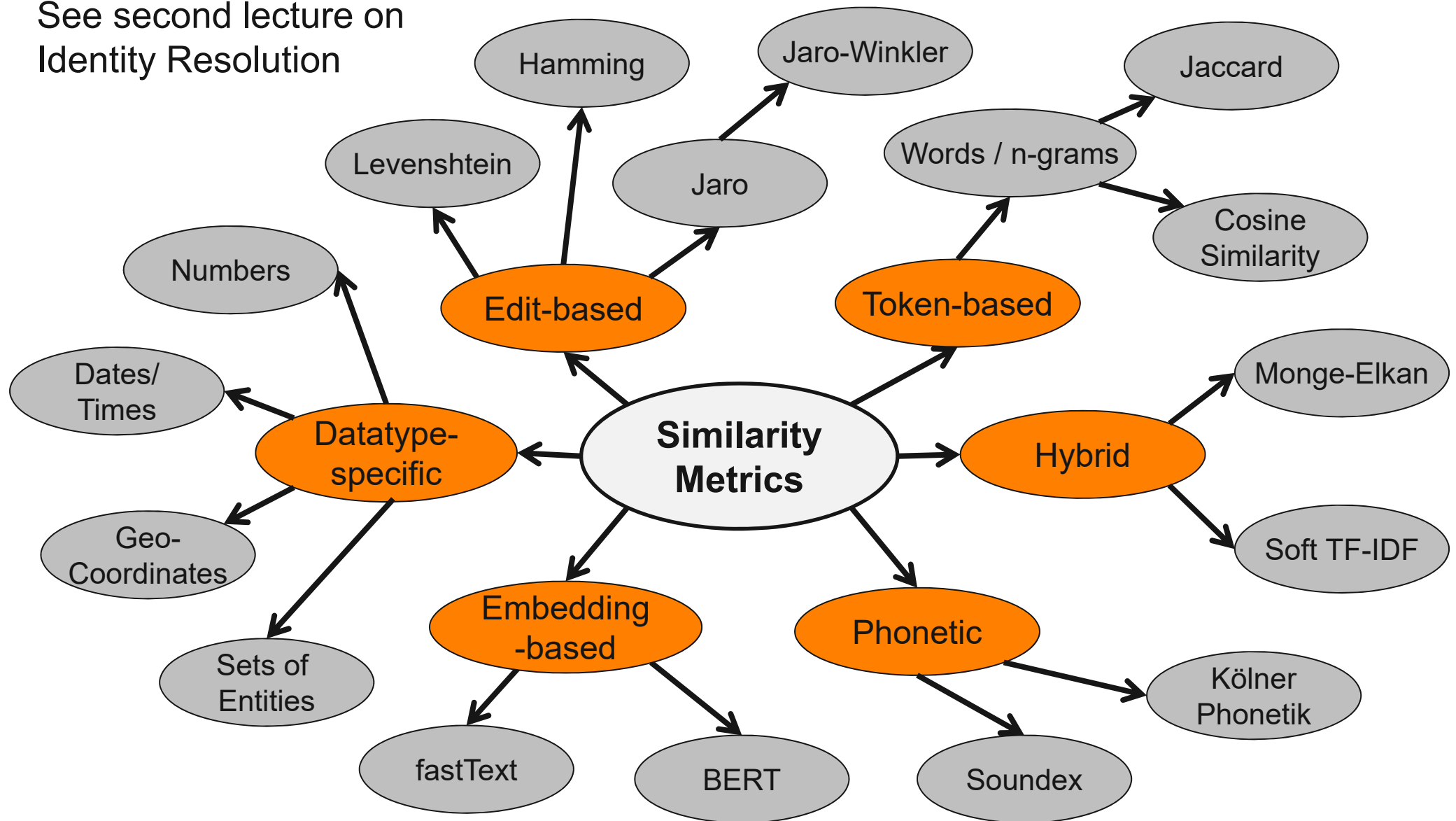
Example Metric: Levenshtein

- 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

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
- 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 dictionary of synonyms
- Expand with hypernyms (is-a relationships)
 - expand product into book, dvd, cd
- Use taxonomy/ontology containing hypernyms for matching
 - similarity = closeness of concepts within taxonomy/ontology

Useful Resources for Pre-Processing

- Google Translate
 - recognizes languages and translates terms
- WordNet
 - provides synonyms and hypernyms for English words
- Wikipedia/DBpedia
 - provides synonyms, concept definitions, category system, cross-language links
 - see Paulheim: WikiMatch. 2012.



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

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

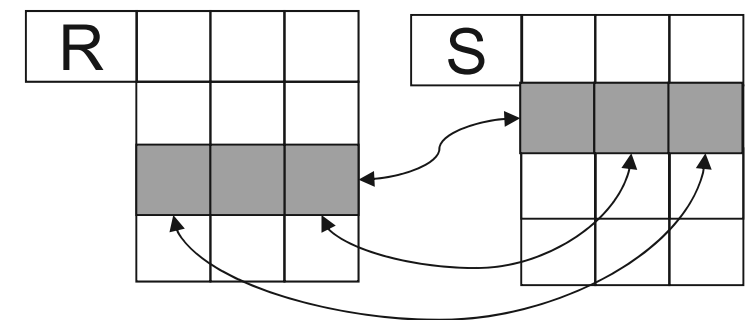
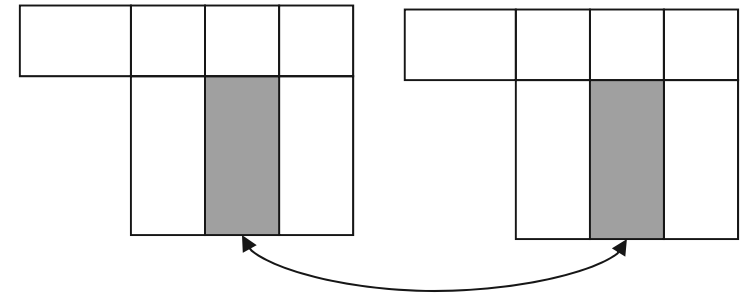
	ID	Name	Loc
Nr	$d(<0,1>, <0,1>)$	$d(<1,6>, <0,1>)$	$d(<1,18>, <0,1>)$
Adresse	$d(<0,1>, <1,16>)$	$d(<1,6>, <1,16>)$	$d(<1,18>, <1,16>)$
Telefon	$d(<0,1>, <1,11>)$	$d(<1,6>, <1,11>)$	$d(<1,18>, <1,11>)$

Discussion: Feature-based Methods

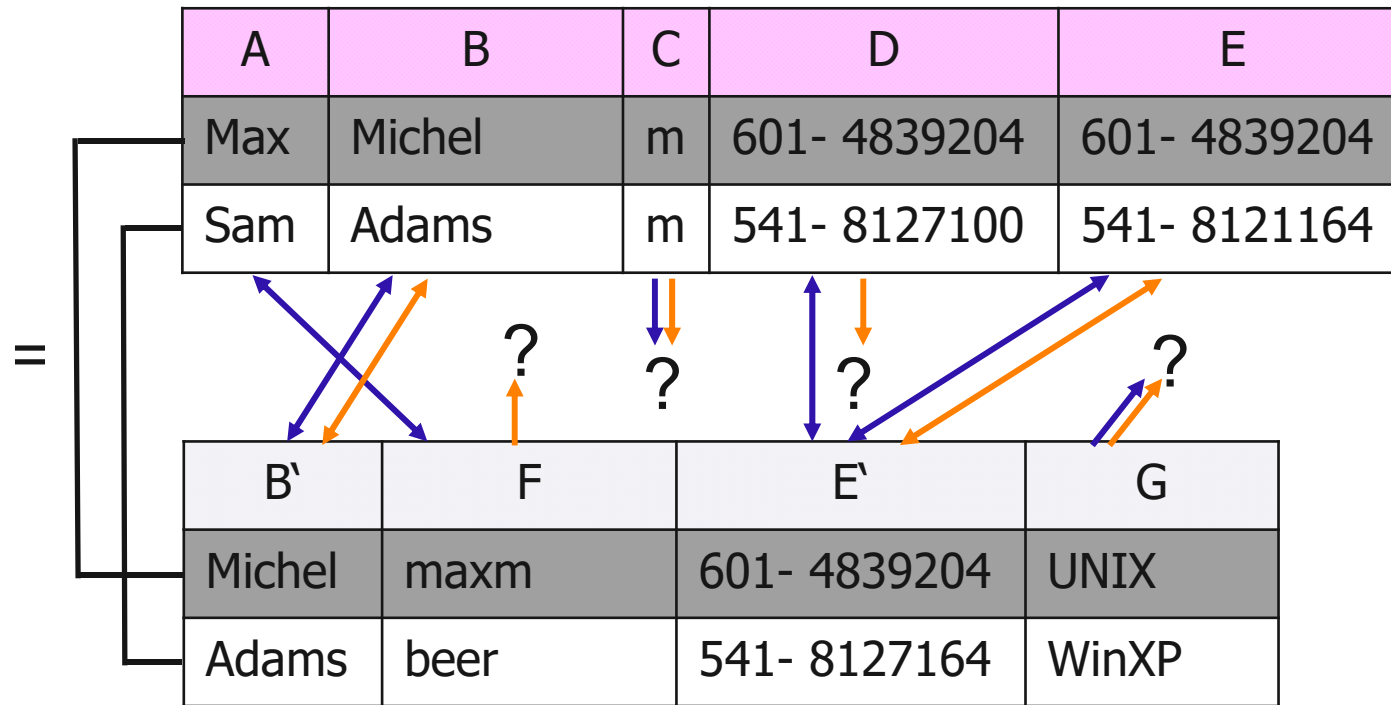
1. Require decision which features to use
 - good features depend on the attribute data type and application domain
2. Require decision how to compare and combine values
 - e.g. cosine similarity, euclidian distance (of normalized values), ...
 - different features likely require different weights
3. 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 previous 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**.



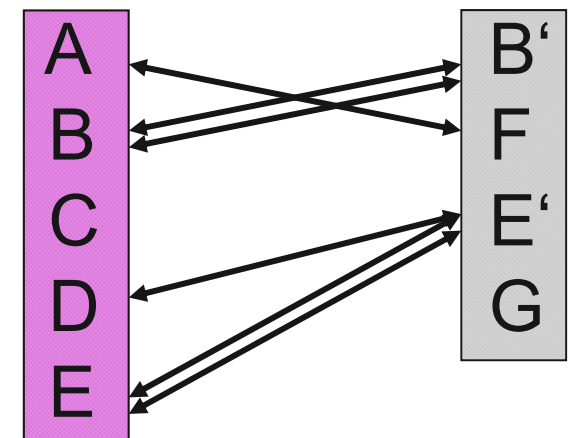
Example: Vote of Two Duplicates







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) <i>[citation needed]</i>		
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

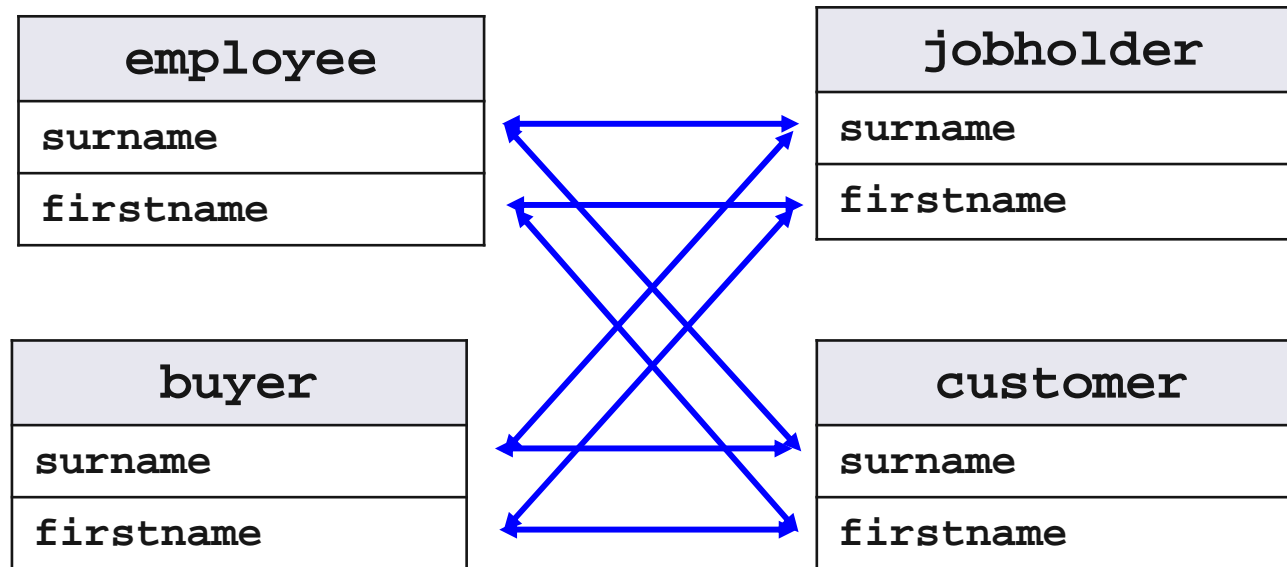
Source: Felix Naumann, ICIQ 2012 Talk

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 ISBN or GenID
- Does not work well if identity resolution is noisy
 - e.g. persons or cities of which only the name is known

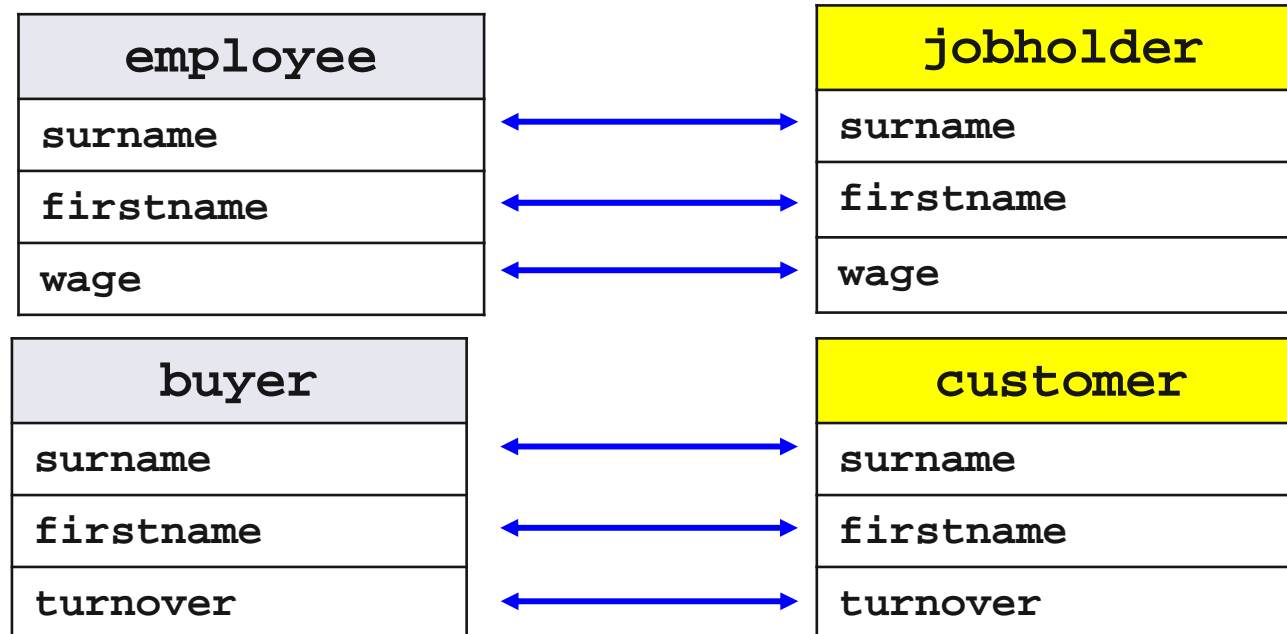
5.2.3 Structure-based Schema Matching Methods

- Addresses the following problem:



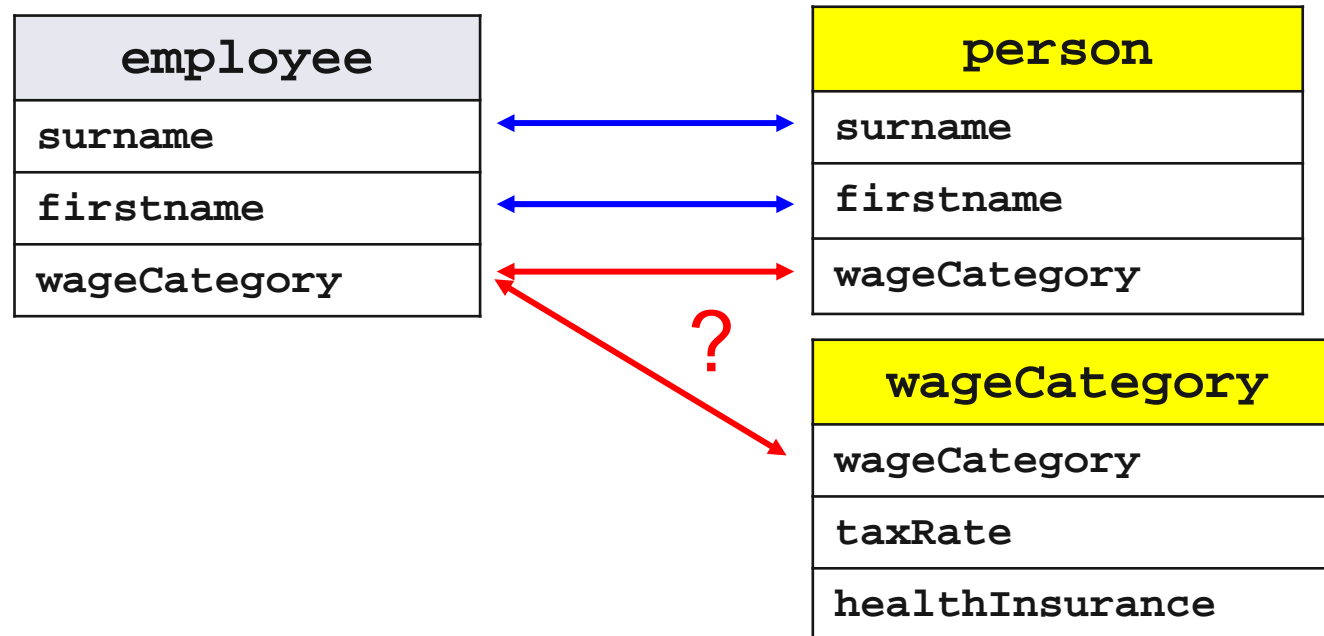
- Attribute-Attribute-Matching
 - Instance-based: Values of all attributes rather similar
 - Label-based: Labels of all attributes rather similar
 - All matchings are about equally good ☹

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)

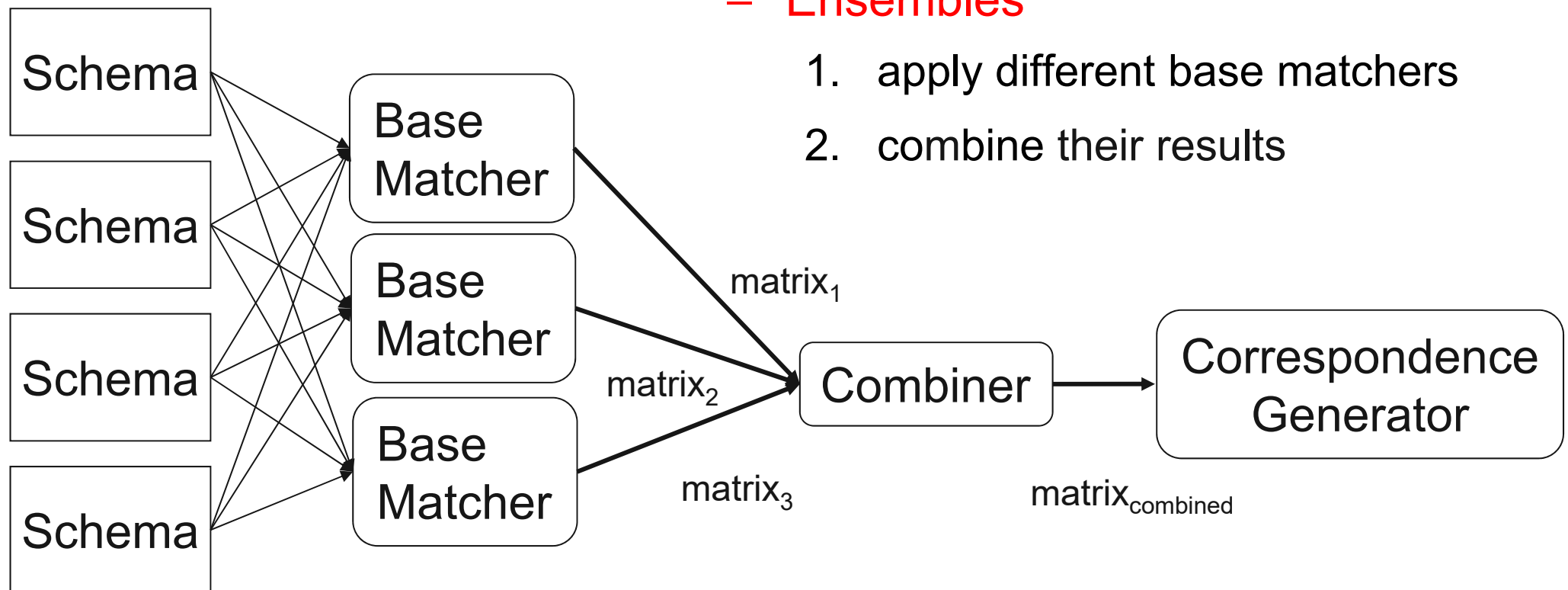
5.2.4 Combined Approaches

– Hybrid Approaches

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

– 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 \$320K *****	James Smith Mike Doan *****	(305) 729 0831 ***** (617) 253 1429	(305) 616 1822 ***** (617) 112 2315	Fantastic house ***** 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 \$230K	(206) 634 9435 (617) 335 4243	Beautiful yard 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

How to Combine the Predictions of Multiple Matchers?

- **Average combiner:** trusts all matchers the same
- **Maximum combiner:** when we trust a strong signal from a single matcher.
- **Minimum combiner:** when we want to be more conservative and require high values from all matchers
- **Weighted-sum combiner**
 - assign a weight to each matcher according to its quality
 - you may learn the weights using
 - known correspondences as training data
 - linear/logistic regression or decision tree learning algorithms
 - we will cover learning weights in detail in chapter on identity resolution

5.3 Generating Correspondences from the Similarity Matrix

Input: Matrix containing attribute similarities

Output: Set of correspondences

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

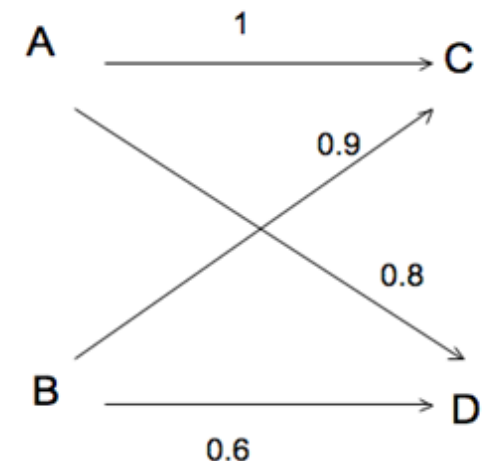
- give domain expert TopK correspondences for each attribute

3. Top1

- directly return the best match as correspondence
- very optimistic, errors might frustrate domain expert

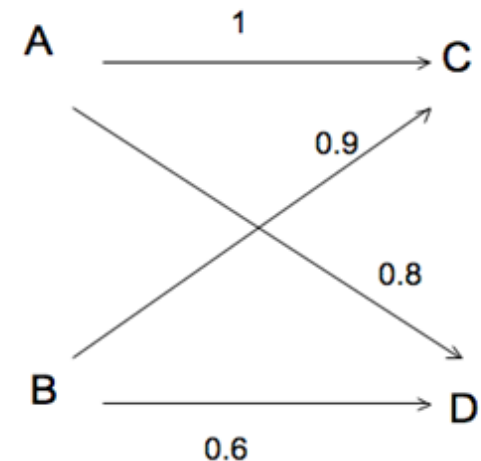
Alternative: Global Matching

- Looking at the complete mapping (all correct correspondences between A and B) gives us the additional restriction that 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

- 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



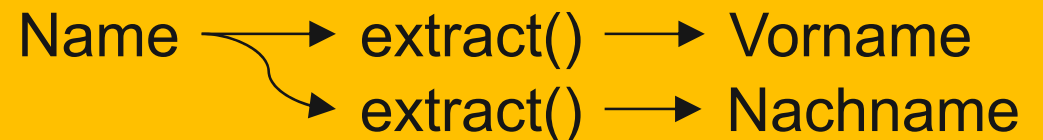
5.4 Finding Many-to-One and One-to-Many Correspondences

- Up till now all methods only looked for 1:1 correspondences
- But real-world setting might require n:1 and 1:n or even n:m correspondences
- Question:
 - How to combine values?
 - Lots of functions possible.
- Problem:
 - Should we test
 $1.2 * A + 2 * B - 32 \equiv C$
 - ... unlimited search space!

n:1 Correspondence



1:n Correspondence



m:n Correspondence



Search for Complex Correspondences

- Paper: Doan, et al.: iMAP: Discovering complex Semantic Matches between Database Schemas. SIGMOD, 2004.
- Employs specialized searchers:
 - text searcher: uses only concatenations of columns
 - numeric searcher: uses only basic arithmetic expressions
 - date searcher: tries combination of numbers into dd/mm/yyyy pattern
- Key challenge: Control the search.
 - start searching for 1:1 correspondences
 - add additional attributes one by one to sets
 - consider only top k candidates at every level of the search
 - termination based on diminishing returns

An Example: Text Searcher

Mediated-schema

price	num-baths	address
-------	-----------	---------

homes.com

listed-price	agent-id	full-baths	half-baths	city	zipcode
320K	532a	2	1	Seattle	98105
240K	115c	1	1	Miami	23591

concat(agent-id,city)
532a Seattle
115c Miami

concat(agent-id,zipcode)
532a 98105
115c 23591

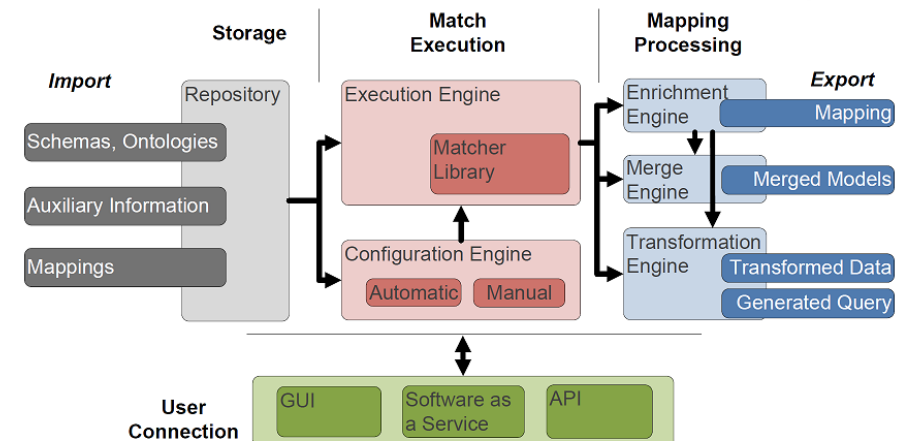
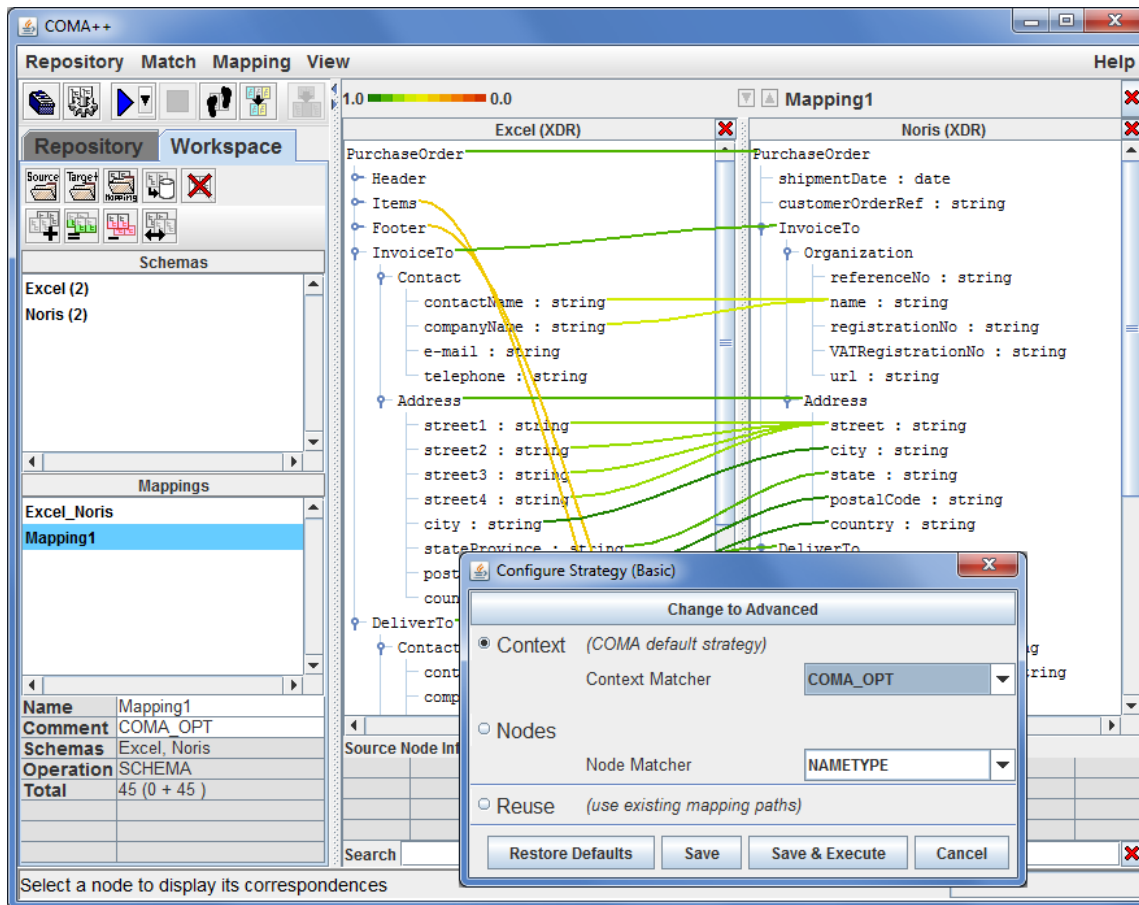
concat(city,zipcode)
Seattle 98105
Miami 23591

- Best match candidates for **address**
 - (agent-id,0.7), (concat(agent-id,city),0.75), (concat(city,zipcode),**0.9**)

5.5. Example Matching System: COMA V3.0

Developed by Database Group at University of Leipzig since 2002

- provides wide variety of matchers (label, instance, structure, hybrid)
- provides user interface for editing correspondences.
- provides data translation based on the correspondences.



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

5.6. Summary

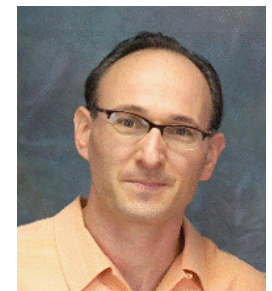
- Schema Matching is an active research area with lots of approaches
 - yearly competition: Ontology Alignment Evaluation Initiative (OAEI)
- Quality of found correspondences depends on difficulty of problem
 - many approaches work fine for toy problems, but fail for larger schemas
 - hardly any commercial implementations of the methods
- 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
 - click log analysis of query results
 - DBpedia Mapping Wiki
 - Spread the manual integration effort over time
 - pay-as-you-go integration in data spaces (aka data lakes)

The Dataspace Vision

Alternative to classic data integration systems in order to cope with growing number of data sources.

Properties of dataspaces

- may contain any kind of data (structured, semi-structured, unstructured)
- require no upfront investment into a global schema
- provide for data-coexistence
- provide give best effort answers to queries
- rely on pay-as-you-go data integration



Franklin, M., Halevy, A., and Maier, D.: From Databases to Dataspaces
A new Abstraction for Information Management, SIGMOD Rec. 2005.

Madhavan, J., et al.: Web-scale Data Integration: You Can Only Afford to Pay As You Go, CIDR 2007.

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, ...



- Open Graph Protocol

- 25 Types: Event, product, place, website, book, profile, article

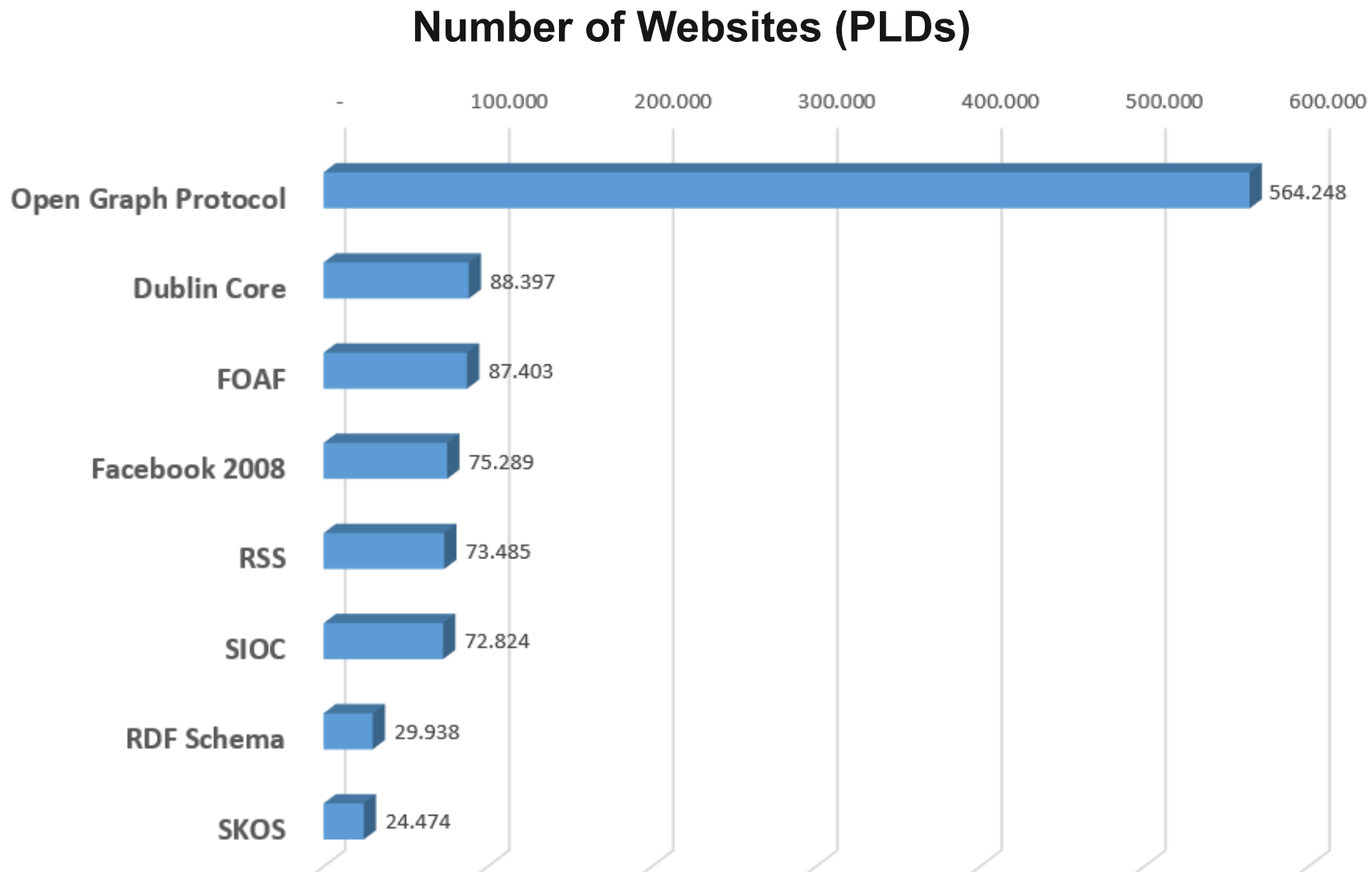


- Linked Data

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



Vocabularies used together with the RDFa Syntax



Source: <http://webdatacommons.org/structureddata/2018-12/stats/html-rdfa.xlsx>

Properties used to Describe Schema:Products

Top Attributes	PLDs Microdata	
	#	%
schema:Product/name	754,812	92 %
schema:Product/offers	645,994	79 %
schema:Offer/price	639,598	78 %
schema:Offer/priceCurrency	606,990	74 %
schema:Product/image	573,614	70 %
schema:Product/description	520,307	64 %
schema:Offer/availability	477,170	58 %
schema:Product/url	364,889	44 %
schema:Product/sku	160,343	19 %
schema:Product/aggregateRating	141,194	17 %
schema:Product/brand	113,209	13 %
schema:Product/category	62,170	7 %
schema:Product/productID	47,088	5 %
...

Source: <http://webdatacommons.org/structureddata/2018-12/stats/html-md.xlsx>

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 vocabularies terms only if no common terms exist.
- LOD Cloud Statistics 2014
 - 378 (58.24%) proprietary vocabularies, 271 (41.76%) are non-proprietary
- 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

Reuse Terms from Common Vocabularies

1. Common Vocabularies

- **Friend-of-a-Friend** for describing people and their social network
- **SIOC** for describing forums and blogs
- **SKOS** for representing topic taxonomies
- **Organization Ontology** for describing the structure of organizations
- **GoodRelations** provides terms for describing products and business entities
- **Music Ontology** for describing artists, albums, and performances
- **Review Vocabulary** provides terms for representing reviews

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

- **LinkedGeoData** and **Geonames** locations
- **GenID** and **UniProt** life science identifiers
- **DBpedia** and **Wikidata** wide range of things

Enable Clients to Retrieve the Schema

Clients can resolve the URIs that identify vocabulary terms in order to get their RDFS or OWL 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 .`
 - `ex2:hasName rdfs:label "The name of a person"@en .`
 - `ex2:hasName rdfs:label „Der Name einer Person"@de .`
- Additional description of the term
 - `ex1:name rdfs:comment "Usually the family name"@en .`
 - `ex2:name rdfs:comment`
 `"Usual order: family name, given name"@en .`

Vocabularies are (partly) connected via vocabulary links.

Vocabulary Link

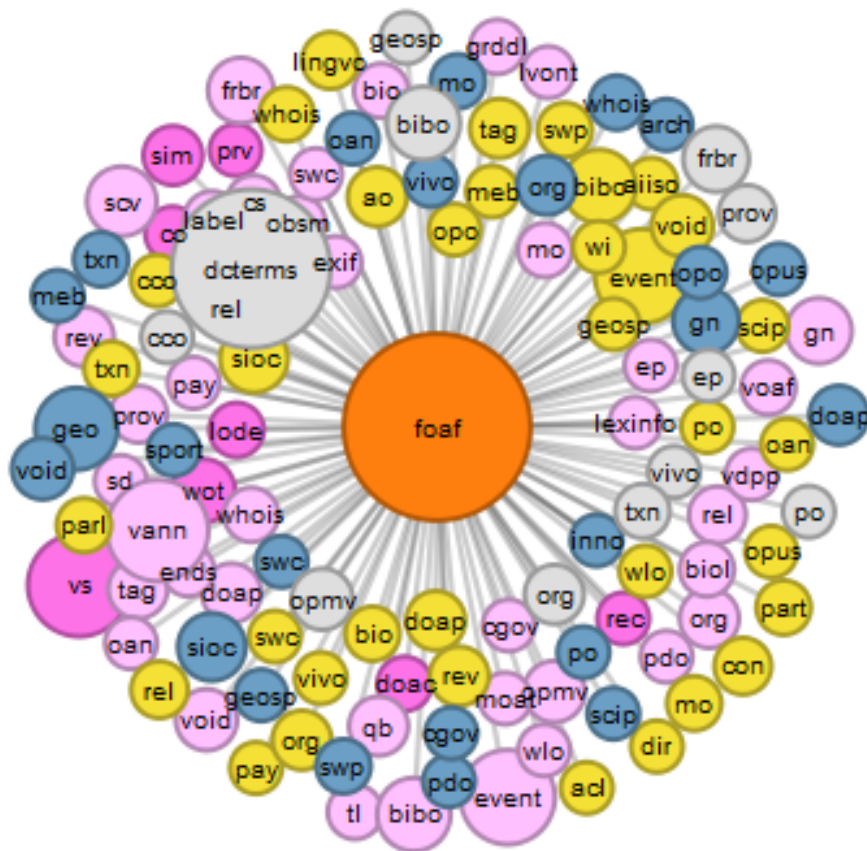
```
<http://dbpedia.org/ontology/Person>  
owl:equivalentClass  
<http://xmlns.com/foaf/0.1/Person> .
```

- 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)

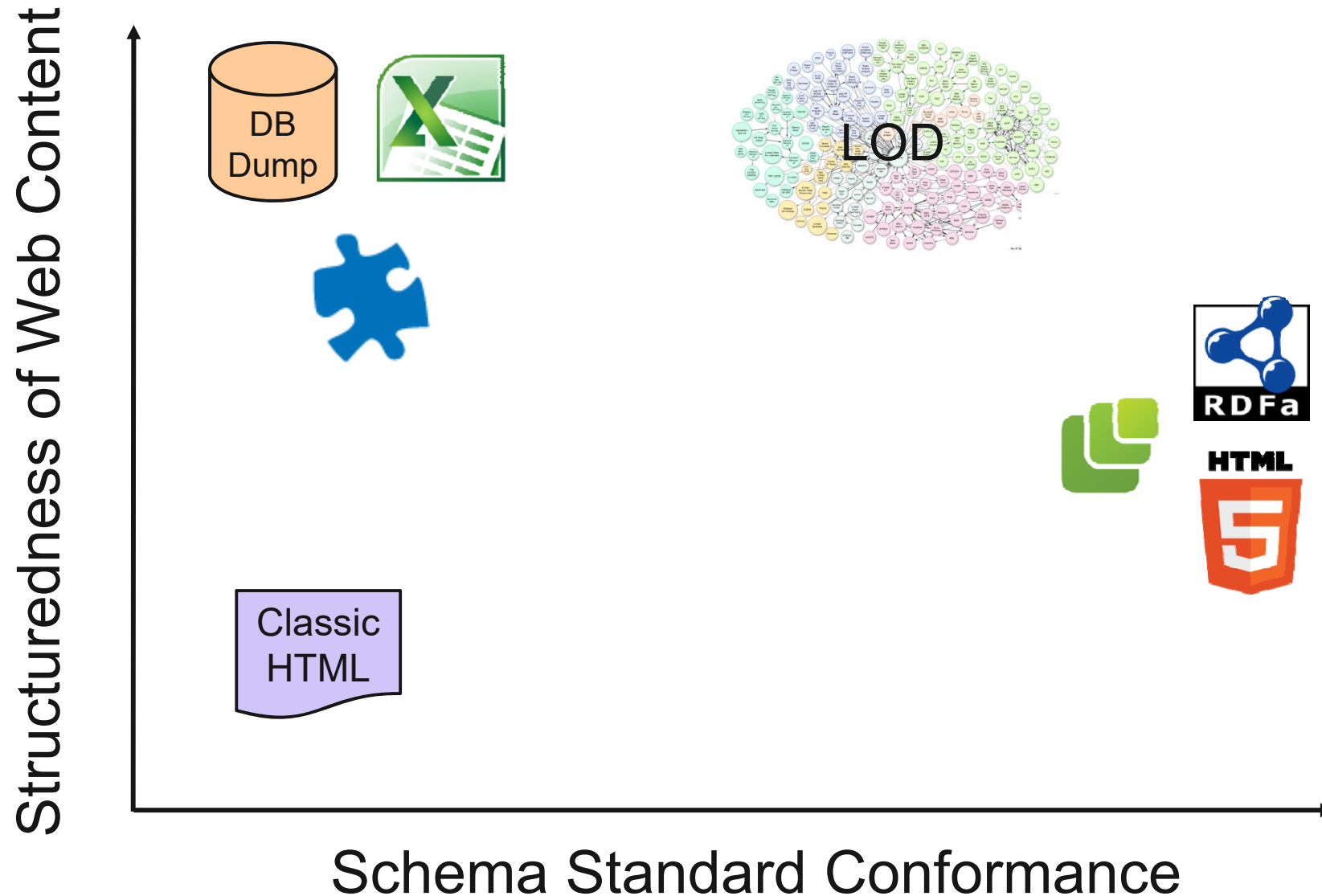


Source: Linked Open Vocabularies,
<https://lov.linkeddata.es/dataset/lov/>

Diagram illustrating 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.
 - Ron Fagin, et al.: Translating Web Data. VLDB, 2002.
- Schema Matching
 - Doan, Halevy, Ives: Principles of Data Integration. Chapter 5, Morgan Kaufmann, 2012.
 - Leser, Naumann: Informationsintegration. Chapter 5.3, DBunkt Verlag, 2007.
 - Dong, Srivastava: Big Data Integration. Chapter 2. Morgan & Claypool Publishers, 2015.
 - Euzenat, Shvaiko: Ontology Matching. Springer, 2013.
 - Rahm, Madhavan, Bernstein: Generic Schema Matching, Ten Years Later. VLDB, 2011.
 - Hertling, Paulheim: WikiMatch - Using Wikipedia for Ontology Matching. Proceedings of the 7th International Workshop on Ontology Matching, 2012.
 - Doan: iMAP: Discovering complex semantic Matches between Database Schemas. SIGMOD, 2004.

References

- Schema Matching (continued)
 - Rinser, Lange, Naumann: Cross-lingual Entity Matching and Infobox Alignment in Wikipedia. *Information Systems (IS)* 38(6):887–907, 2013.
 - Melnik, et al.: Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. *ICDE*, 2002.
 - Avigdor Gal: *Uncertain Schema Matching*. Morgan & Clypool, 2011.
- Data Spaces
 - Franklin, M., Halevy, A., and Maier, D.: From Databases to Dataspaces A new Abstraction for Information Management. *SIGMOD Rec.*, 2005.
 - Madhavan, J., et al.: Web-scale Data Integration: You Can Only Afford to Pay As You Go. *CIDR*, 2007.
- 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.
 - Heath, Bizer: *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.