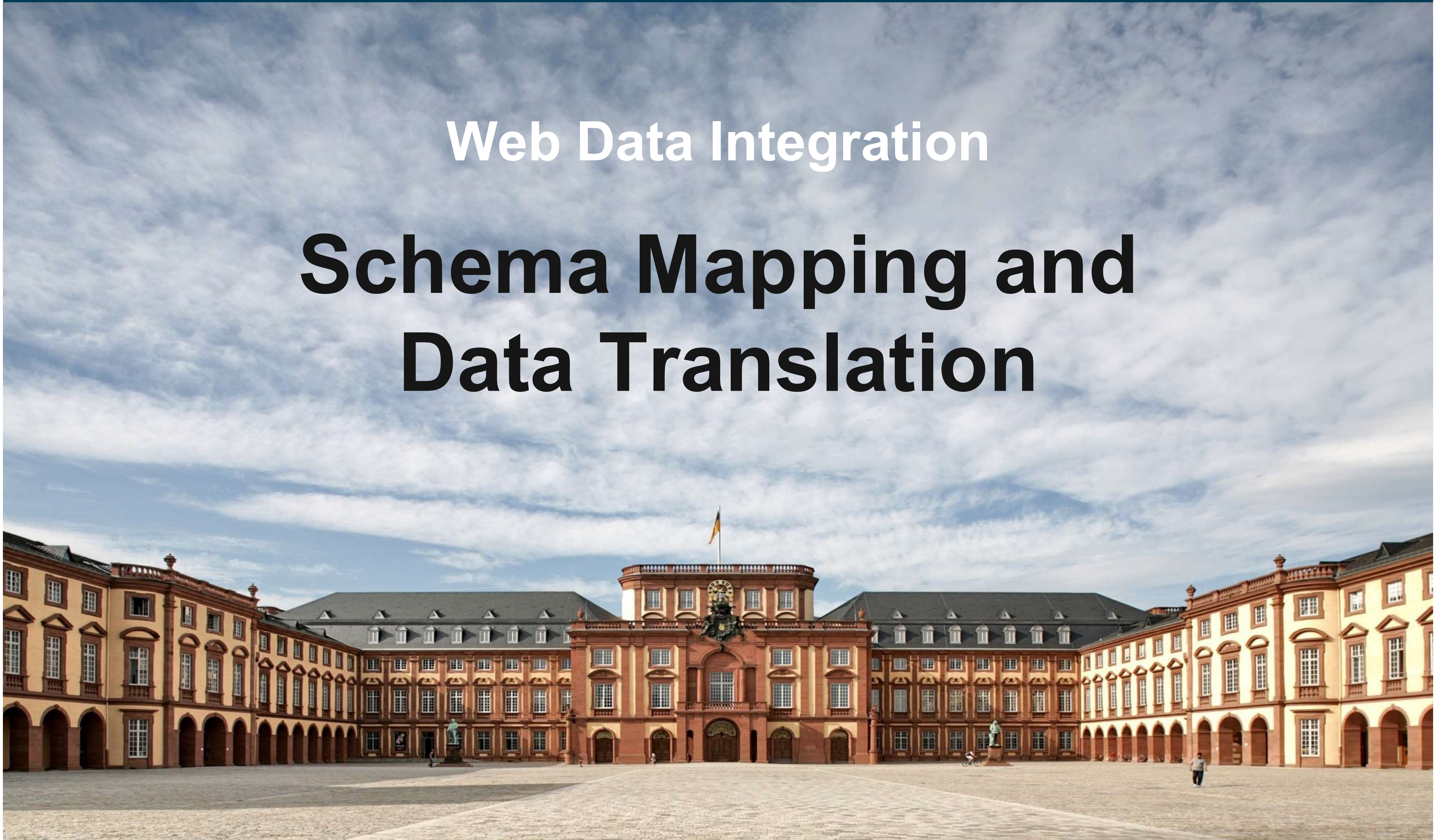
UNIVERSITY OF MANNHEIM

Web Data Integration

Schema Mapping and Data Translation



The Data Integration Process

Data Collection / Extraction

↓

Schema Mapping
Data Translation

↓

Identity Resolution

↓

Data Quality Assessment
Data Fusion

Please form Teams for the Student Projects (IE683)

- we register your teams in the **Student Project Kickoff Session next Thursday**, which everybody must attend!
- Students without a team will be grouped into teams or assigned to existing teams in this session.
- teams of **five students** realize a data integration project including
  1. data gathering
  2. schema mapping and data translation
  3. identity resolution
  4. data quality assessment and data fusion
- **Slide set about student projects** will be online early next week.
  - teams write a 12 pages report about their project, present project results
  - you may choose their own application domain and data sets
    - minimum 4 data sets with a good degree of overlap in attributes and instances
  - in addition, we will propose some suitable data sets
- 3 ECTS (70 % written project report, 30 % presentation of project results)

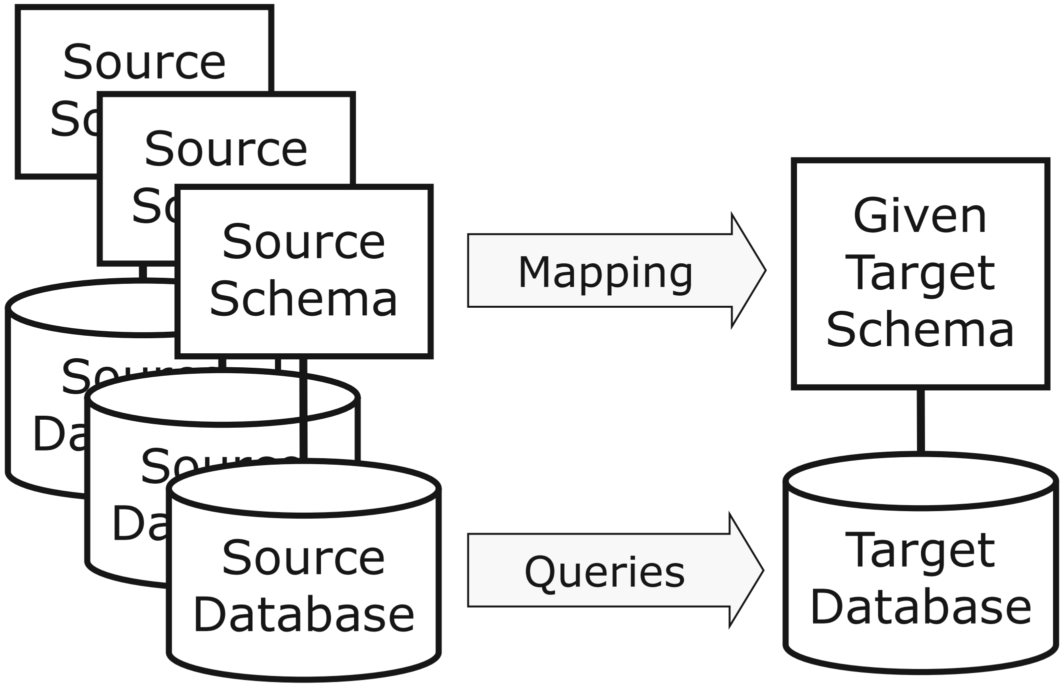
Outline

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

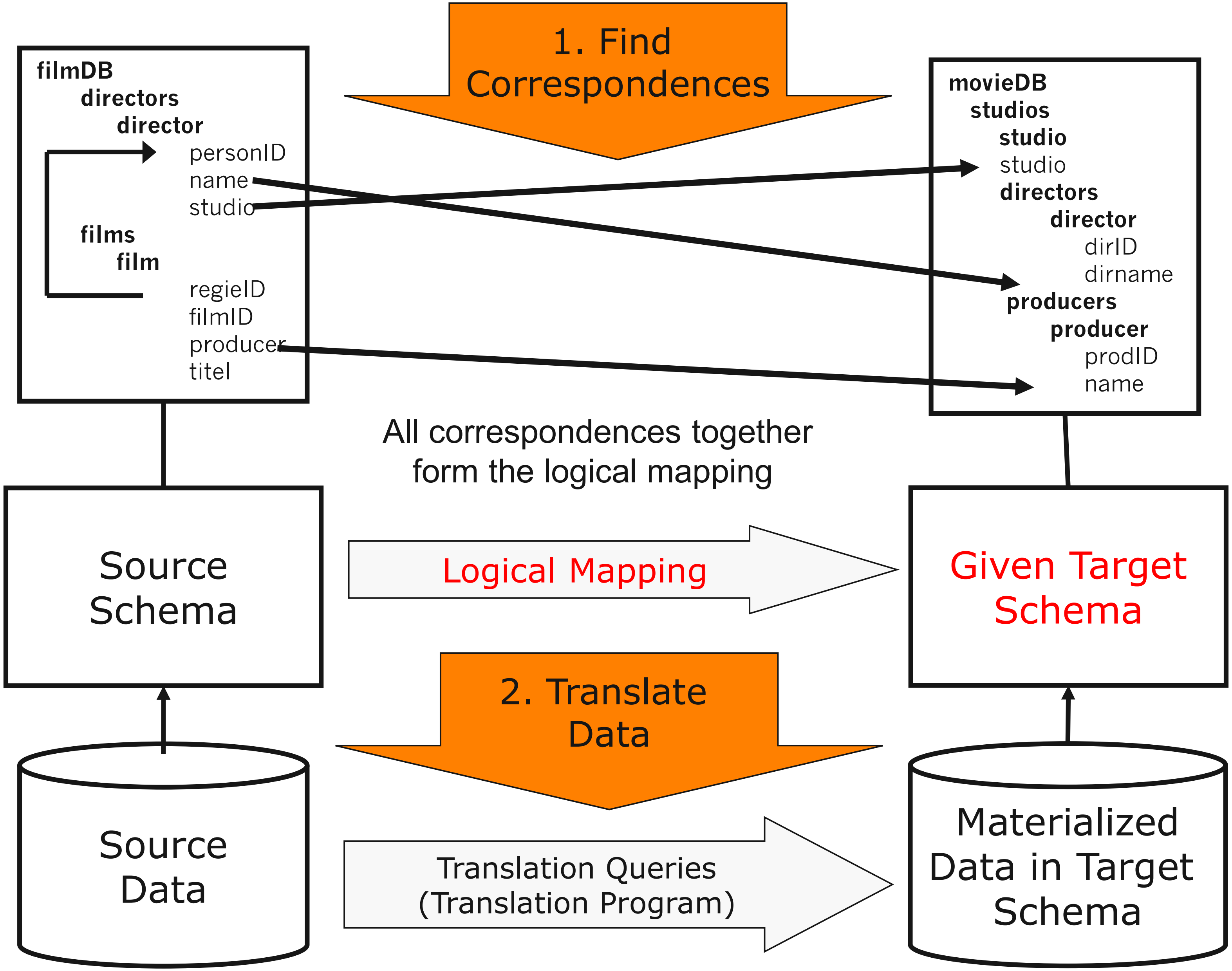
Basic Integration Situation 1: Schema Mapping

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)



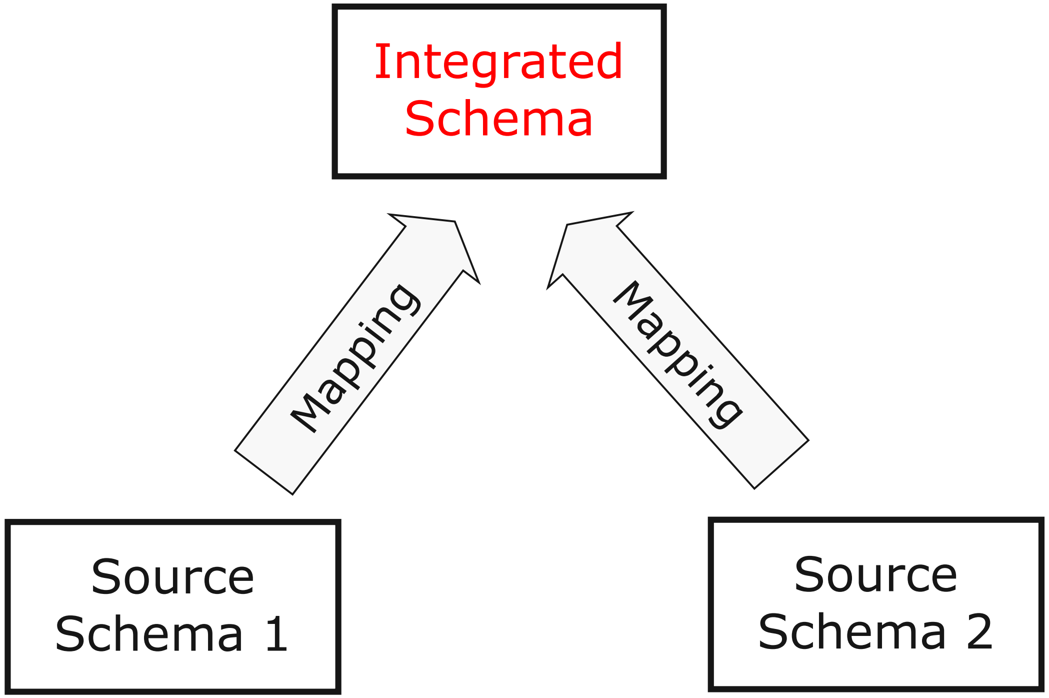
The Schema Mapping Process



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 <u>different</u> 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.

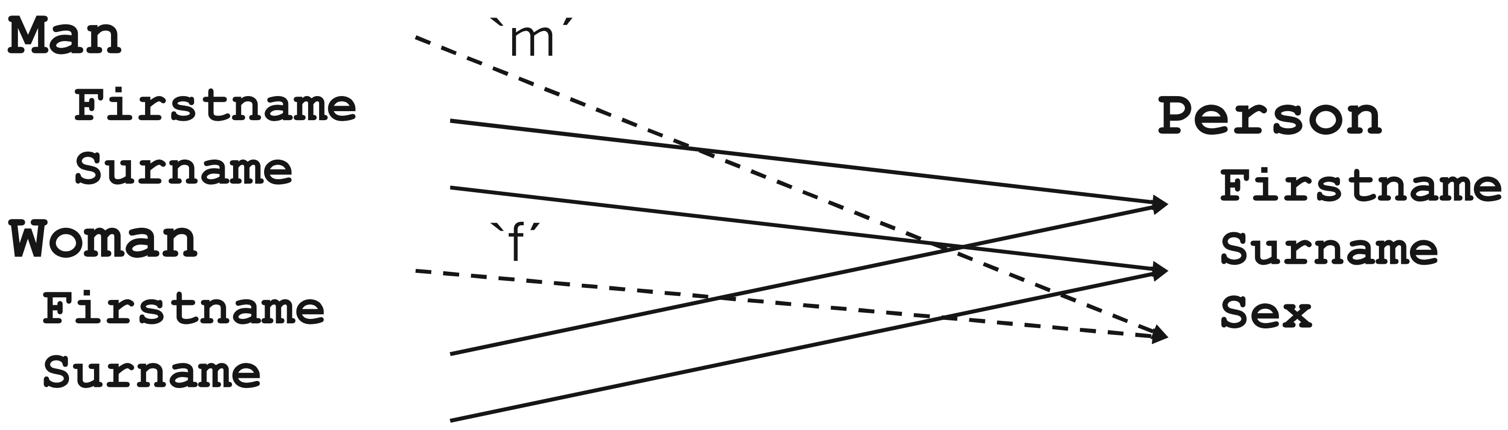
- Mapping = Set of all correspondences that relate S and T
- Correspondences are easier to specify than transformation queries
  - domain expert does not need technical knowledge about query language
  - specification can be supported by user interfaces (mapping editors)
  - step-by-step process with 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

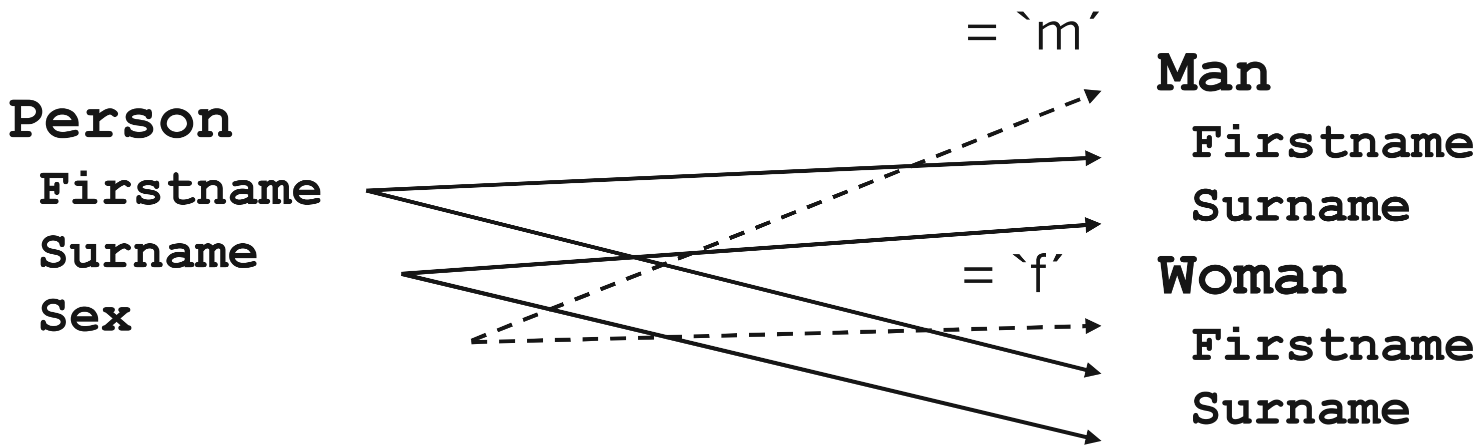
- One-to-One Correspondences
  - Movie.title → Item.name
  - Product.rating → Item.classification
  - Movie ≡ Film (equivalence: Same semantic intention)
  - Athlete ⊆ Person (inclusion: All athletes are also persons)
- One-to-Many Correspondences
  - Person.Name → split() → FirstName (Token 1)
→ Surname (Token 2)
- Many-to-One Correspondences
  - Product.basePrice * (1 + Location.taxRate) → 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

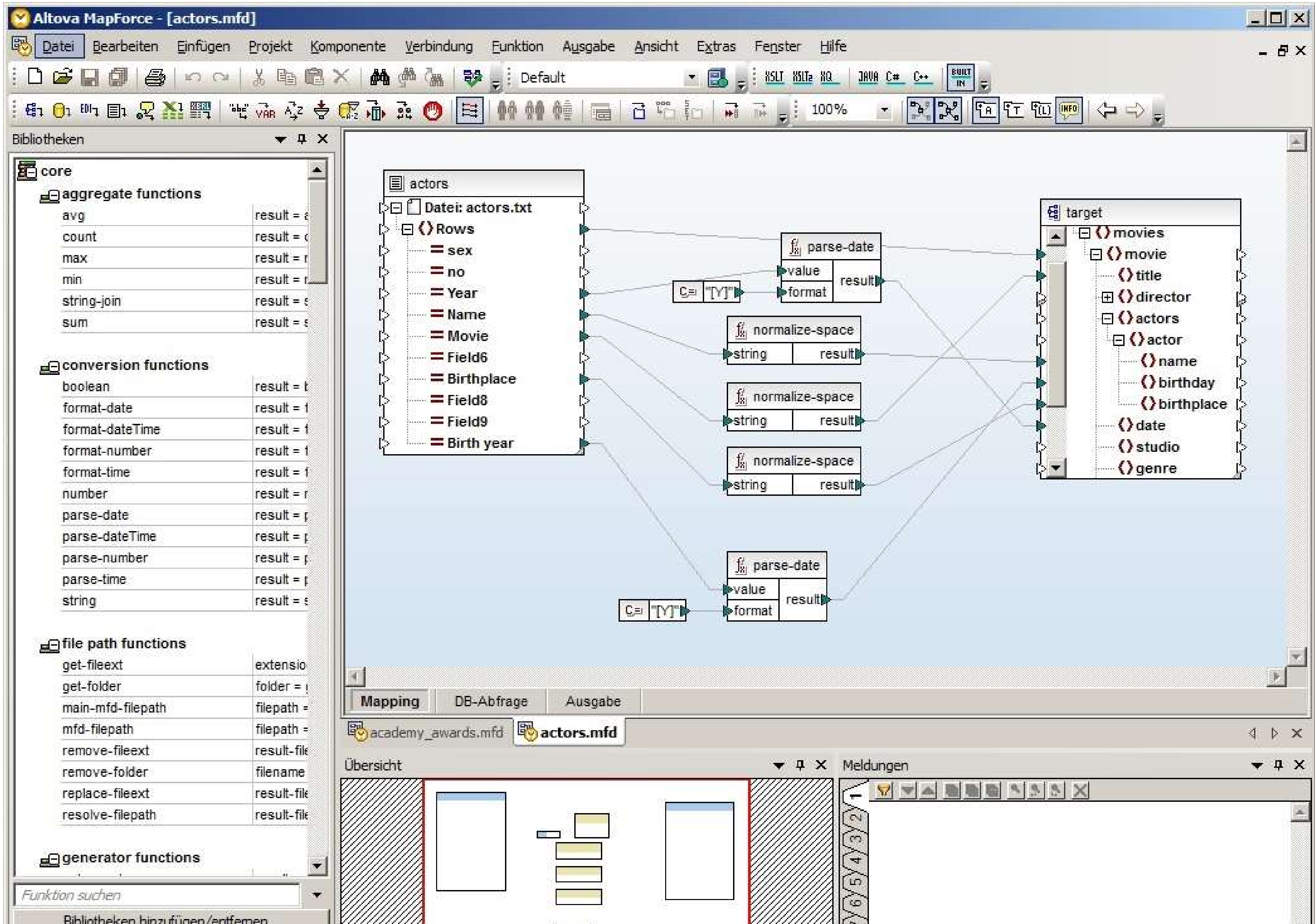
- Naming of
  - Relations
  - Attributes
- Normalized vs. Denormalized
- Nesting vs. Foreign Keys

1:1, 1:n, n:1 Correspondences

- Alternative Modelling
  - Relation vs. Value
  - Relation vs. Attribute
  - Attribute vs. Value

Higher-order Correspondences

Defining Correspondences



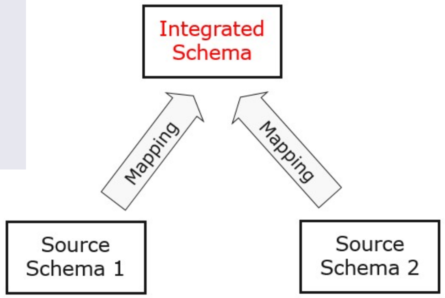
Discovering Correspondences

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 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.

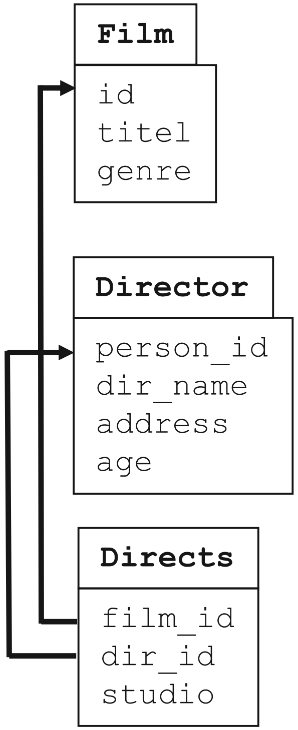


- Goals:
  - Completeness: All elements of the source schemata should be covered
  - Correctness: All data should be represented semantically correct
    - cardinalities, integrity constraints, …
  - Minimality: The integrated schema should be minimal in respect to the number of relations and attributes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

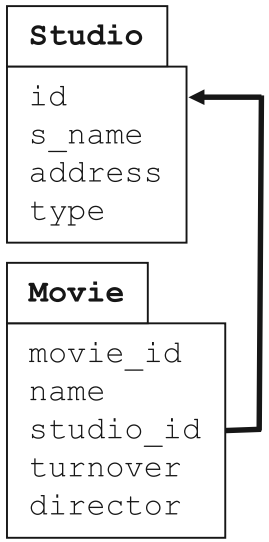
Example: Two Schemata about Films

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



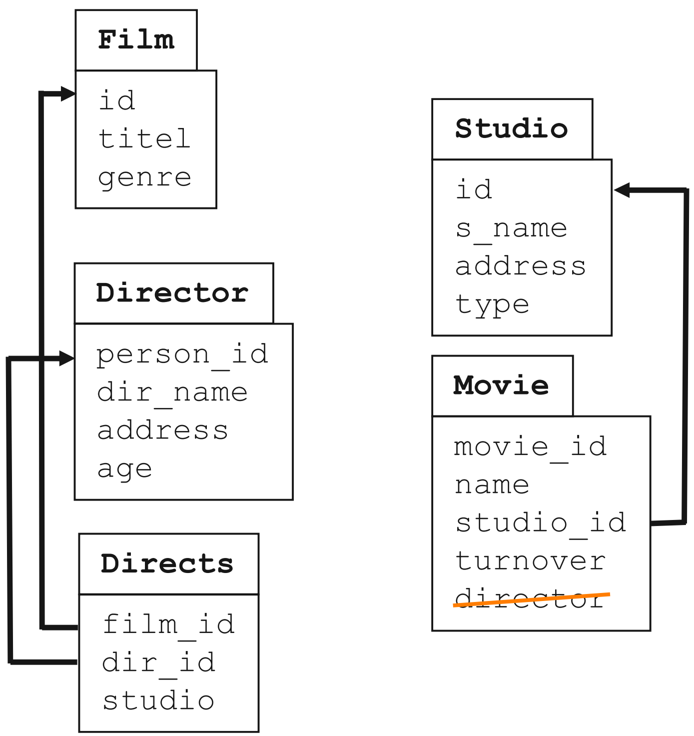
N:M Relationship

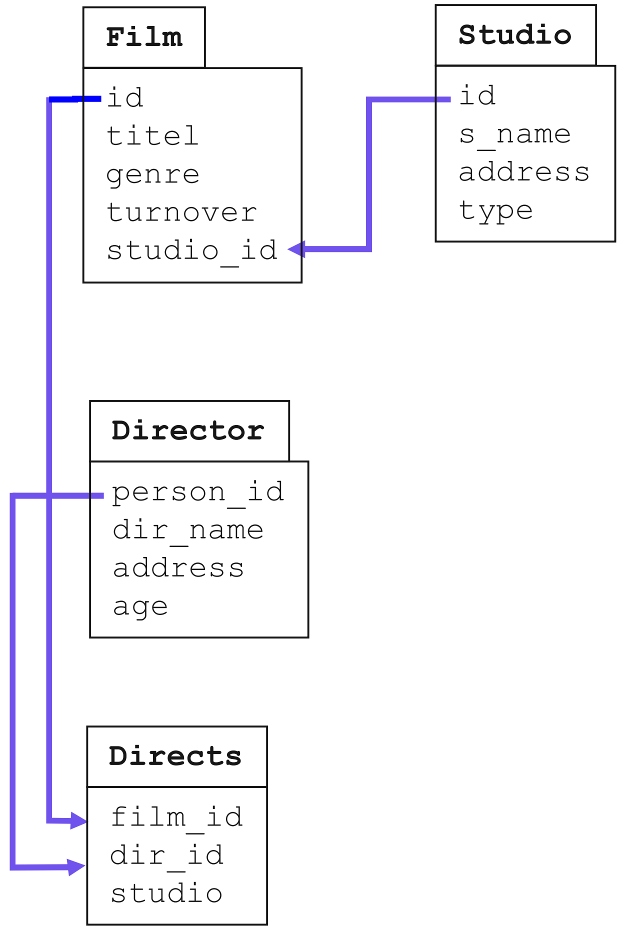


Normalized Schema

Schema Integration: Rules of Thumb

1. Merge all tables with 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 ≡ A‘, B ≡ B‘, A-B ≡ A‘-B‘ then include A-B
4. Paths between equivalent attributes and classes

   a) If A≡A‘, B ≡ B‘, A-B ≡ A‘-A_1‘-…-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≡A‘, B ≡ B‘, A-A_1-…-A_n-B ≡ A‘-A_1‘-…-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 ≡ Movie

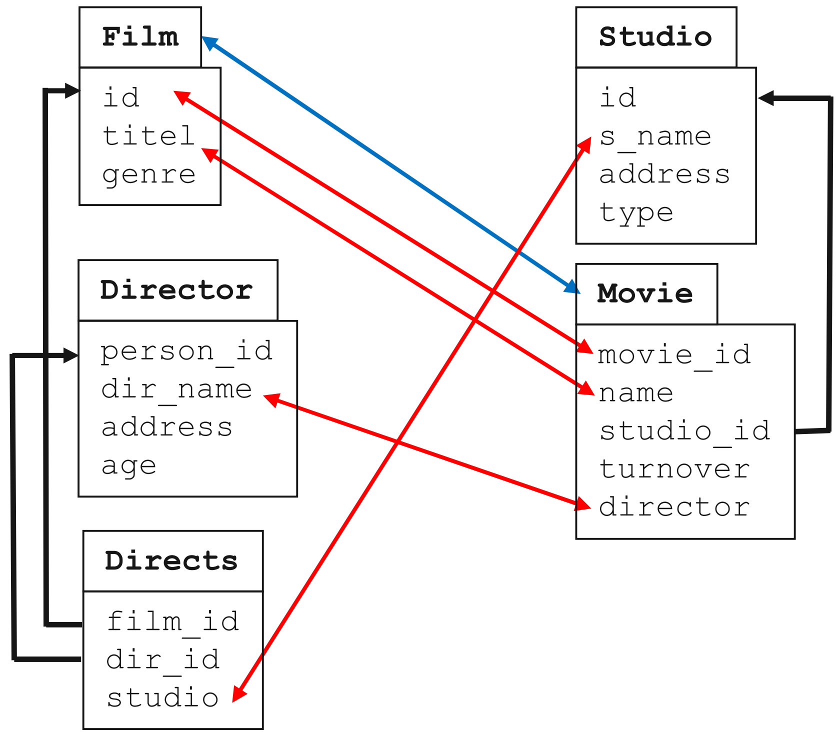
- Attribute Correspondences

  id ≡ movie_id

  titel ≡ name

  dir_name ≡ director

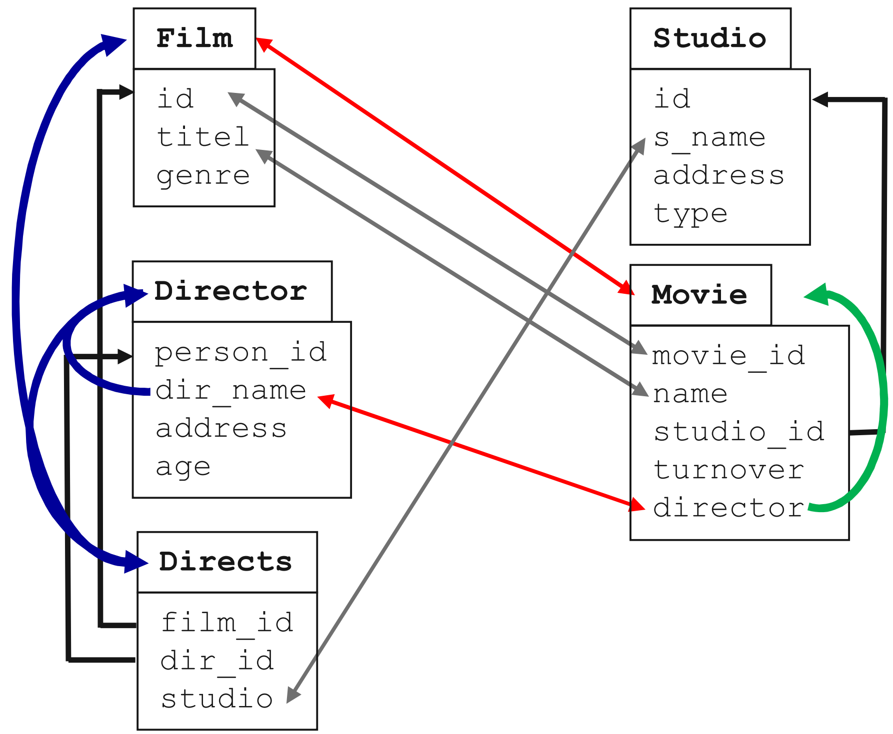
  studio ≡ s_name



Example: Relationship Path Correspondence 1

– Relationship Path Correspondence

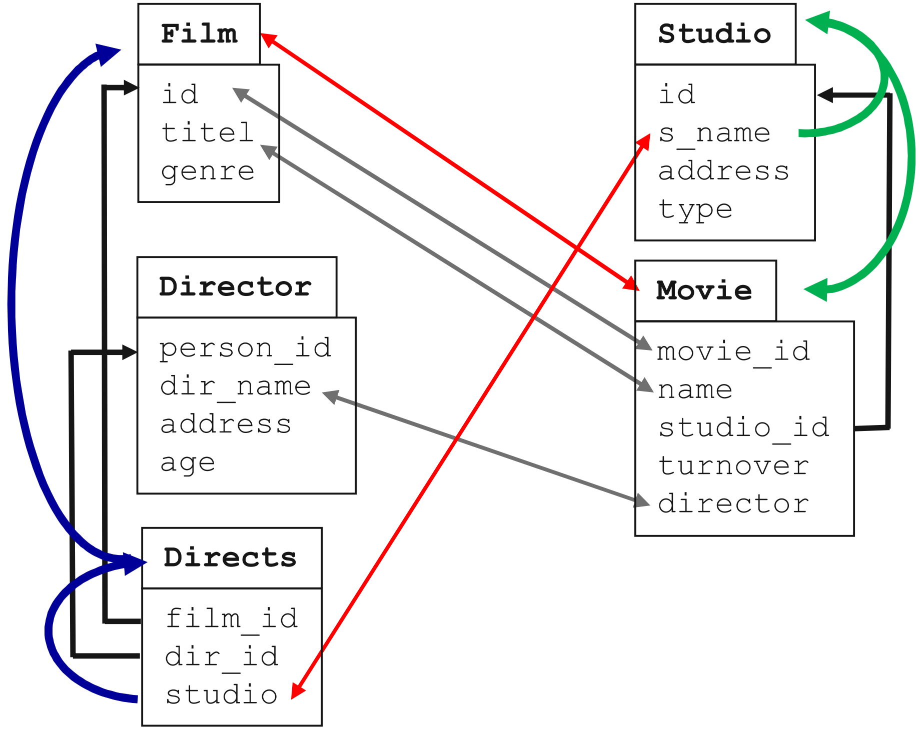
dir_name-Director-Directs-Film ≡ director-Movie



Example: Relationship Path Correspondence 2

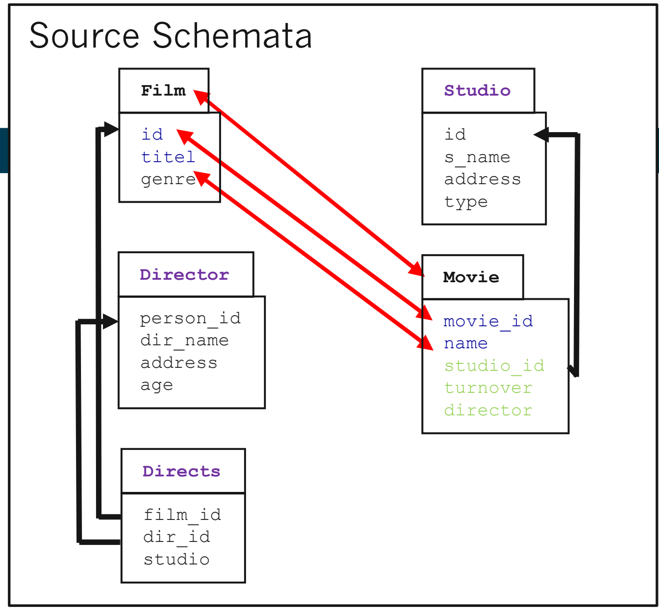
– Relationship Path Correspondence

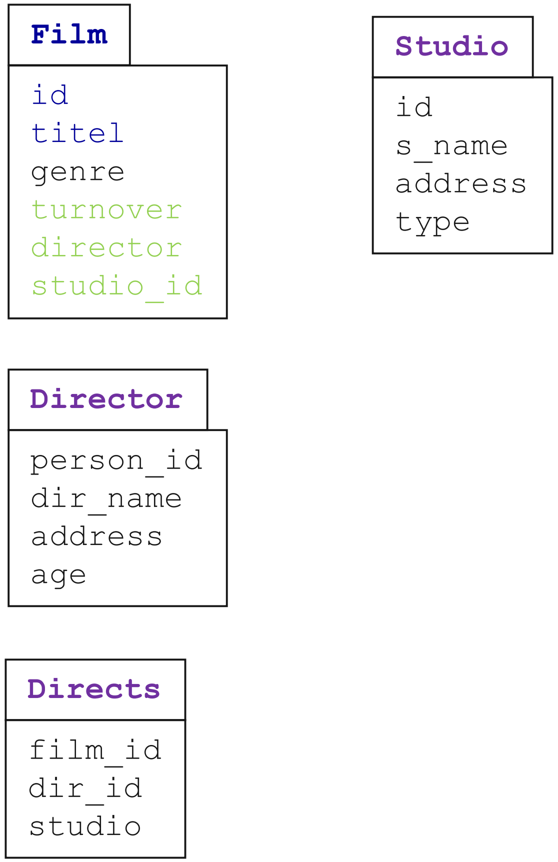
studio-Directs-Film ≡ s_name-Studio-Movie



Creation of the Integrated Schema 1

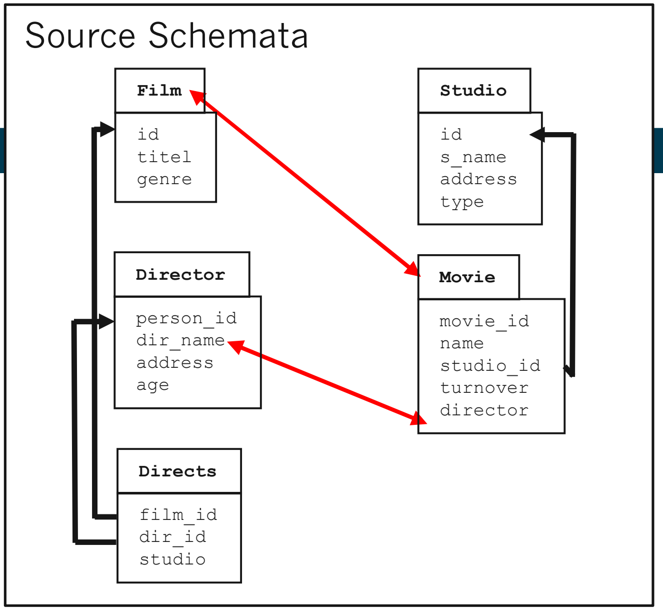
- Integration Steps
  1. Rule 1: Equivalent classes **`Film`** and **`Movie`** are merged to **`Film`**. Attributes are either merged (**`id, title`**) or simply copied (**`turnover, director, studio_id`**).
  2. Rule 2: Classes without direct equivalent are included into the integrated schema (**`Director,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

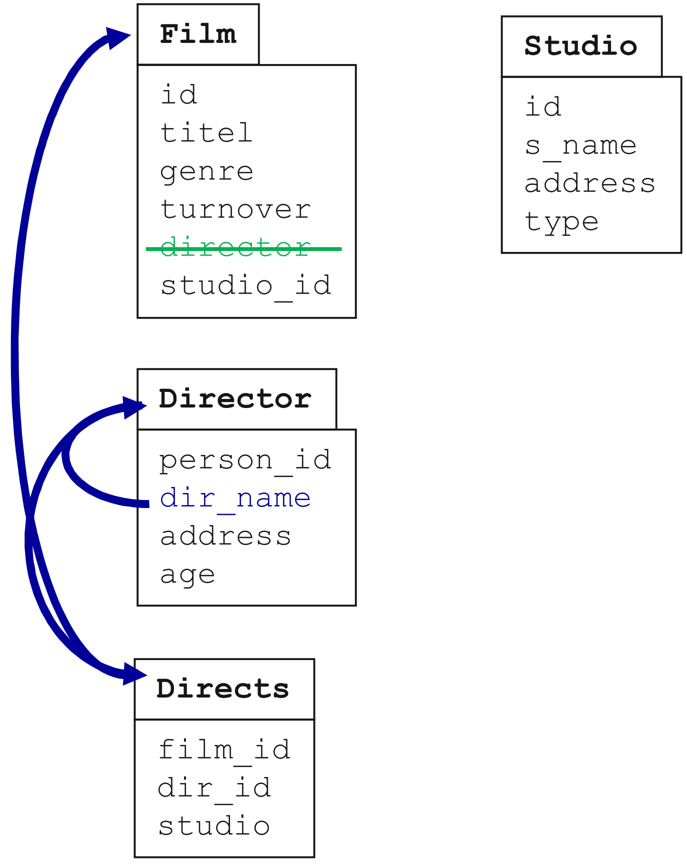




Creation of the Integrated Schema 2

- 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.





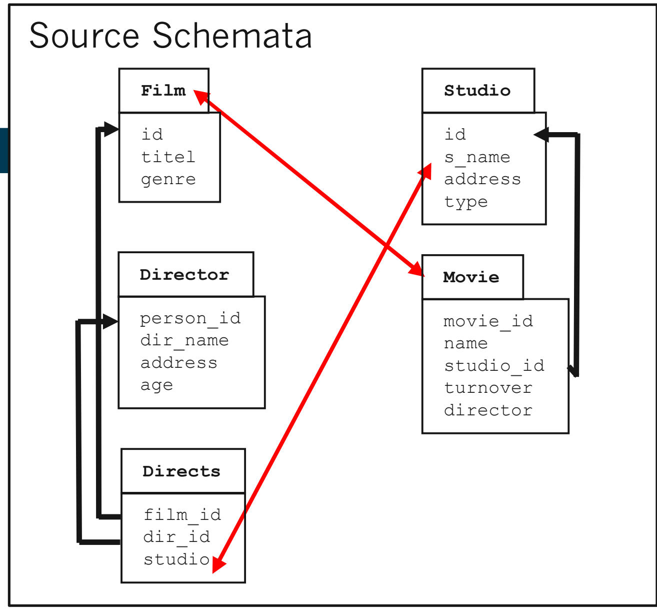
Creation of the Integrated Schema 3

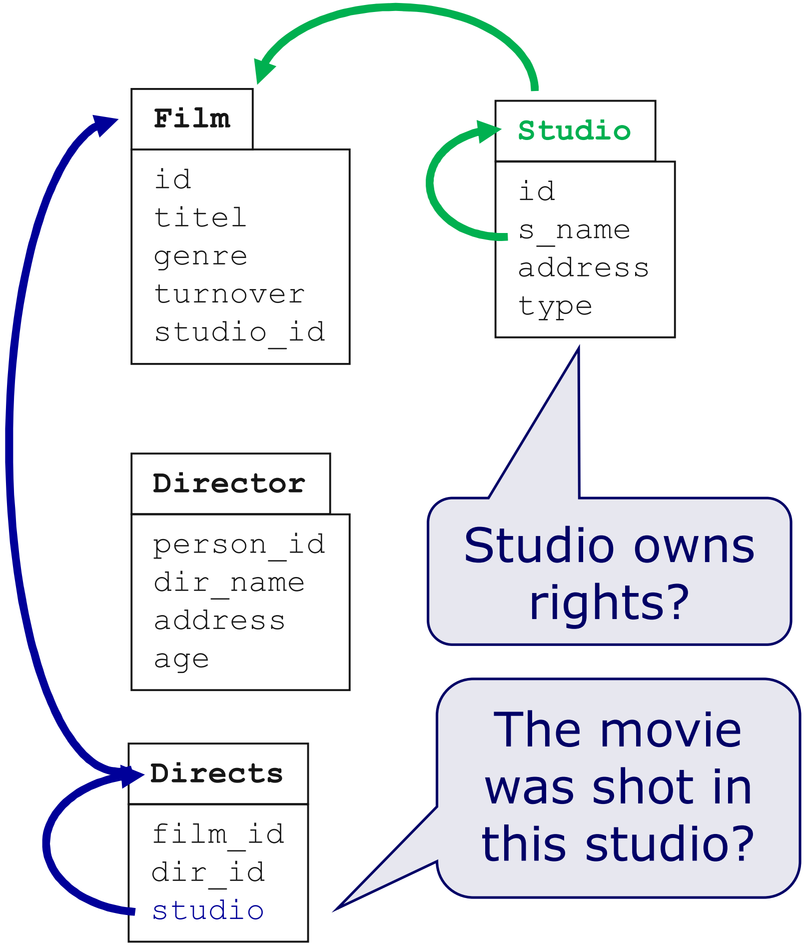
- Correspondence

  studio-Directs-Film ≡
  s_name-Studio–Movie

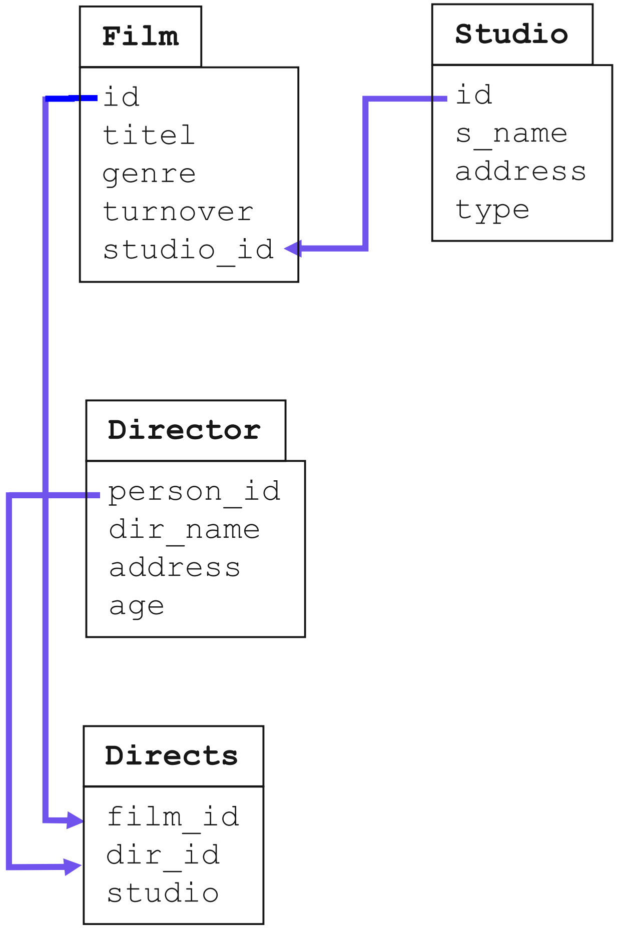
- Integration Step

  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.





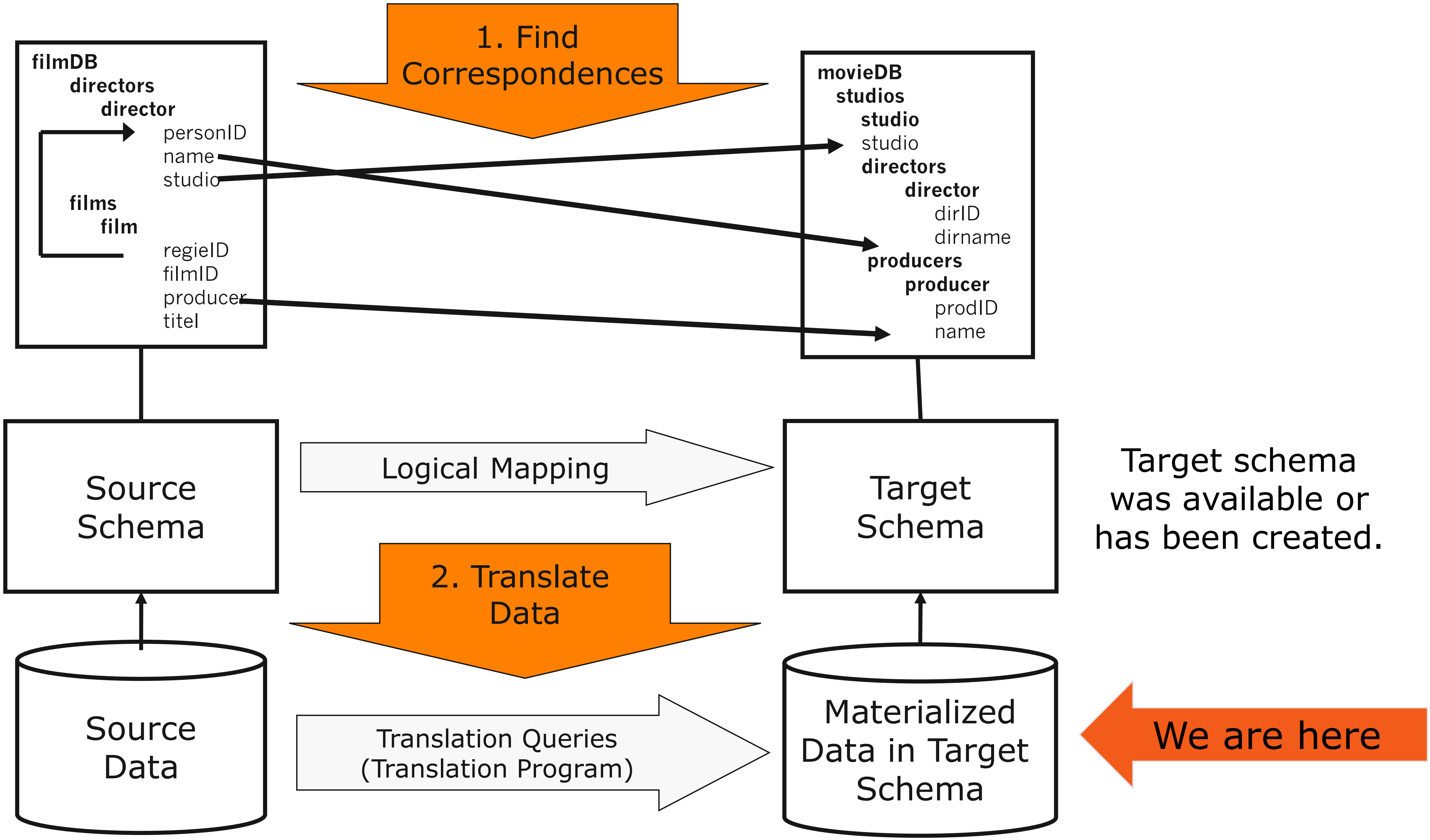
Final Integrated Schema



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



Target schema was available or has been created.

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

AUTHOR
- artFK
- name

PUBLICATION
- pubID
- title
- date
- author

```
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
```

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

Normalized → Denormalized

ARTICLE
- artPK
- title
- pages

AUTHOR
- artFK
- name

PUBLICATION
- pubID
- title
- date
- author

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

ARTICLE
- artPK
- title
- pages

AUTHOR
- artFK
- name

PUBLICATION
- pubID
- title
- date
- author

Decision: Do we want publications without author?

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

Denormalized → Normalized

PUBLICATION
- title
- date
- author

ARTICLE
- artPK
- title
- pages

AUTHOR
- artFK
- name

DISTINCT

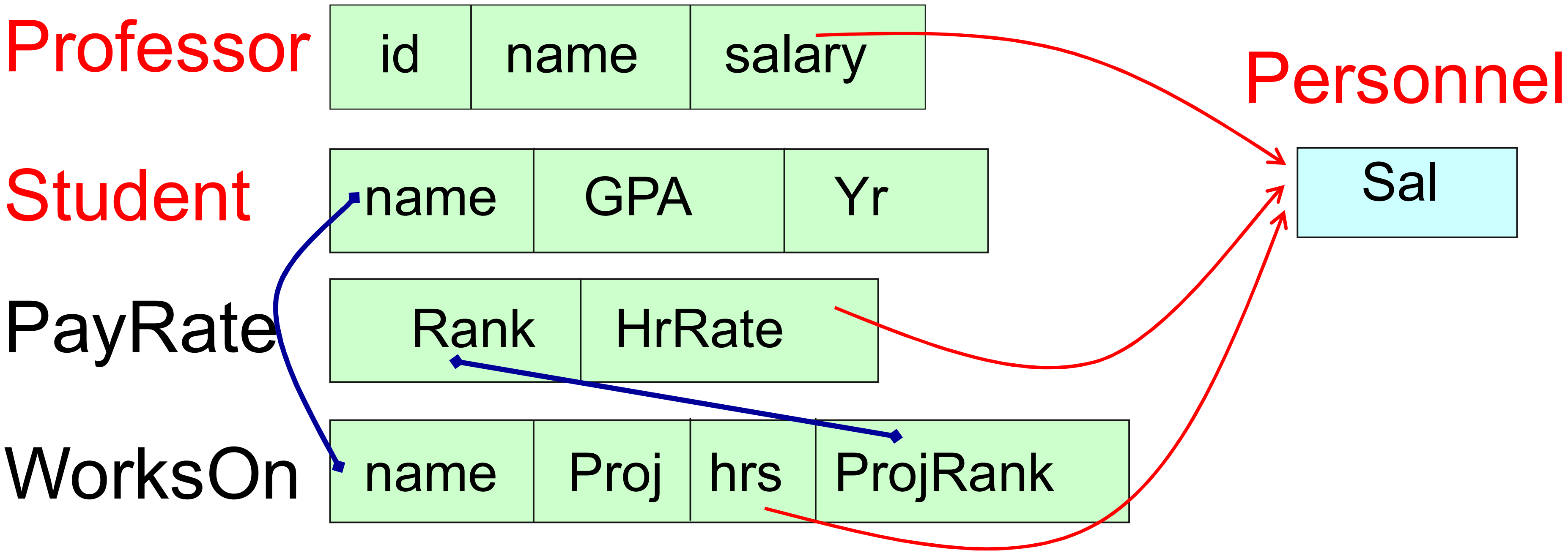
```
SELECT SK(title) AS artPK
       title AS title
       null AS pages
INTO   ARTICLE
FROM   PUBLICATION
```

```
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 → Personnel.Sal

Correspondence 2: 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
```

Complete Algorithms for Generating Translation Queries

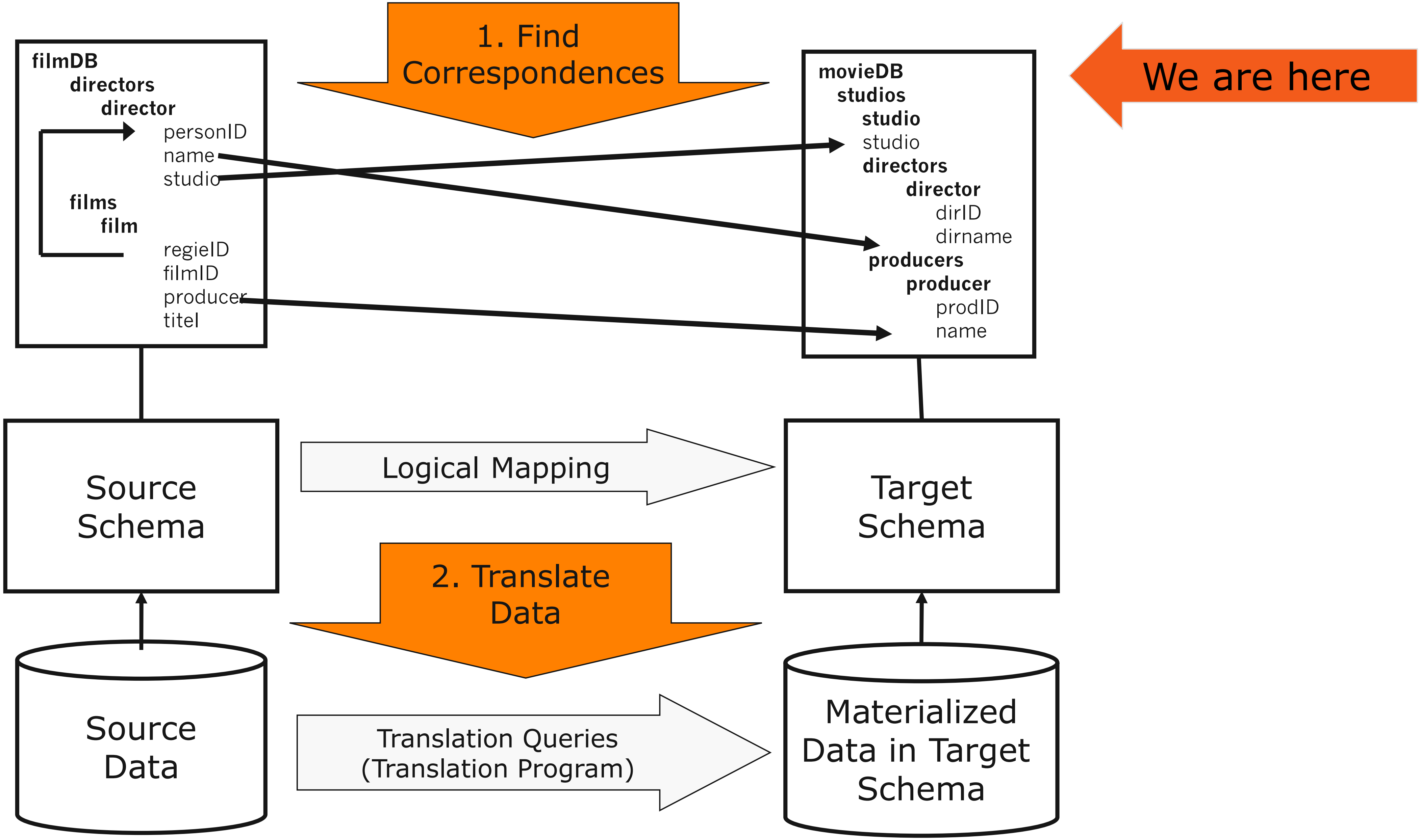
- Relational Case
  - Doan, Halevy, Ives: Principles of Data Integration. Pages 152-158.
- XML Case
  - Leser, Naumann: Informationsintegration. Pages 137-143.
- MapForce
  - implements another one which we will try out in the exercise

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 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. 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

Problem Space: Different Languages and Strange Names

Männer	
Vorname	**Nachname**
Felix	Naumann
Jens	Bleiholder

Frauen	
Vorname	**Nachname**
Melanie	Weis
Jana	Bauckmann

Persons			
firstname	**name**	**male**	**female**
Felix	Naumann	1	0
Jnes	Bleiho.	1	0
Melanie	Weiß	0	1
Jana	baukman	0	1

Pers		
FN	**NN**	**S**
F.	Naumann	M
J.	Bleiholder	M
M.	Weis	F
J.	Bauckmann	F

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 and lots of pre-training)

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

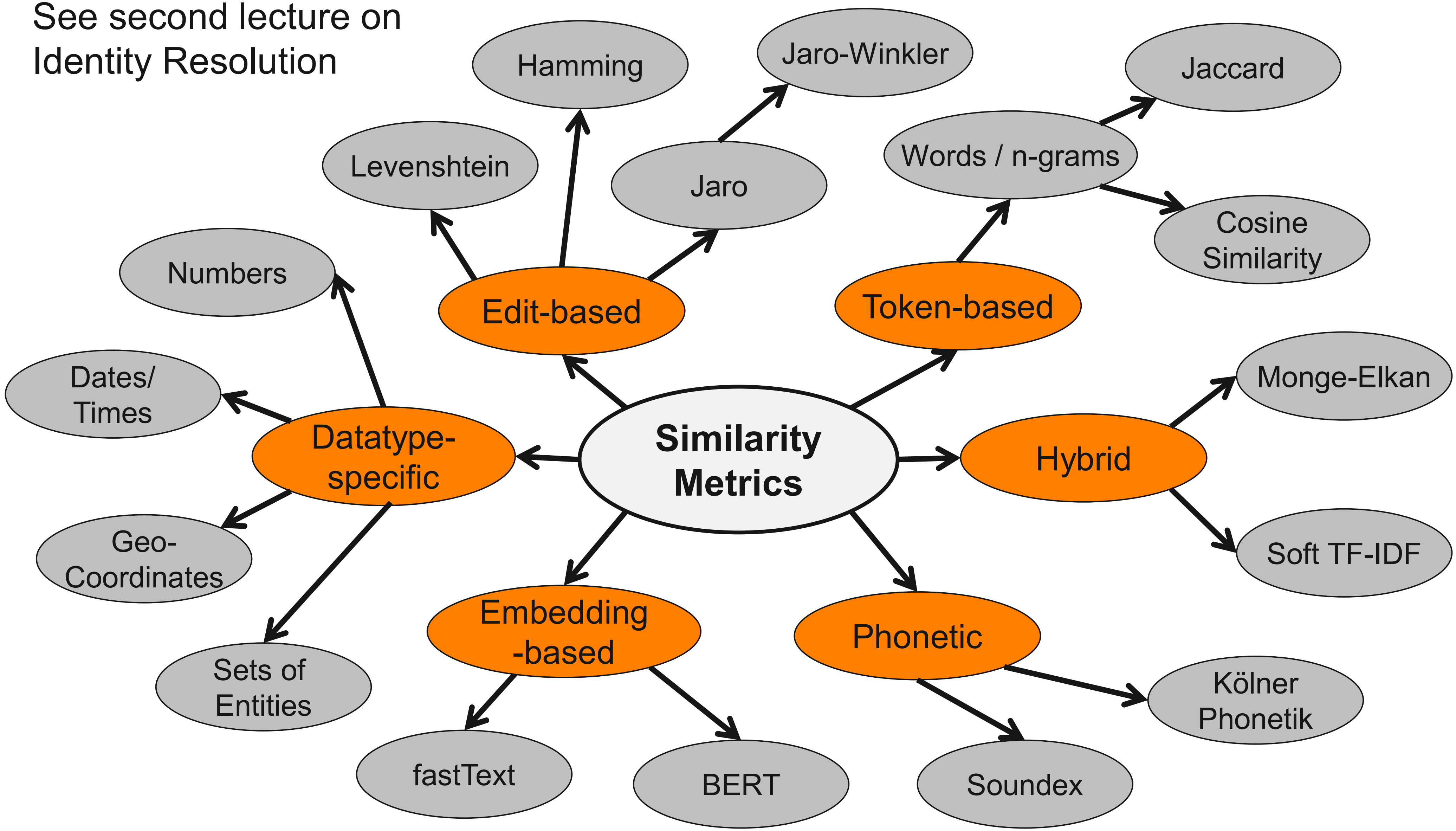
Example Metric: Levenshtein

- 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
  - levensthein('table', 'cable') = 1 (1 substitution)
  - levensthein('Chris Bizer', 'Bizer, Chris') = 11 (10 substitution, 1 insertion)
- Converting Levenshtein distance into a similarity

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

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 Goolge 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

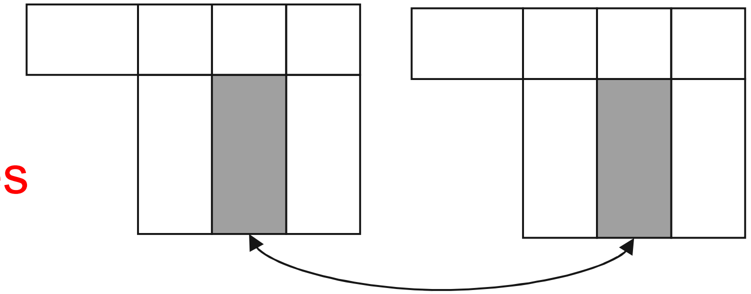
1. Attribute Recognizers
   - employ dictionaries, regexes or rules to recognize values of a specific attribute
     - Dictionaries fit attributes that only contain a relatively small set of values (e.g. age classification of movies (G, PG, PG-13, R), country names, US states
     - Regexes or rules 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, Kochel	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)

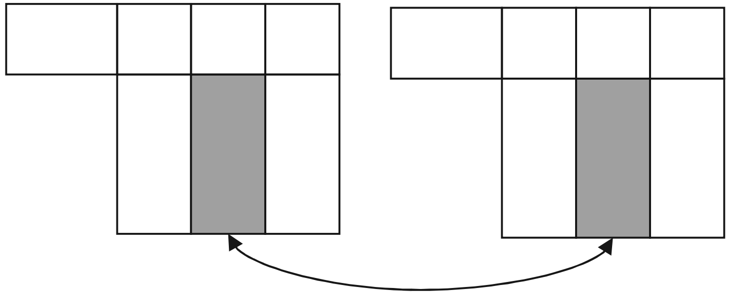
	ID	Name	Loc
Nr	d(<0,1>,<0,1>)	d(<1,6>,<0,1>)	d(<1,18>,<0,1>)
Adresse	d(<0,1>,<1,16>)	d(<1,6>,<1,16>)	d(<1,18>,<1,16>)
Telefon	d(<0,1>,<1,11>)	d(<1,6>,<1,11>)	d(<1,18>,<1,11>)

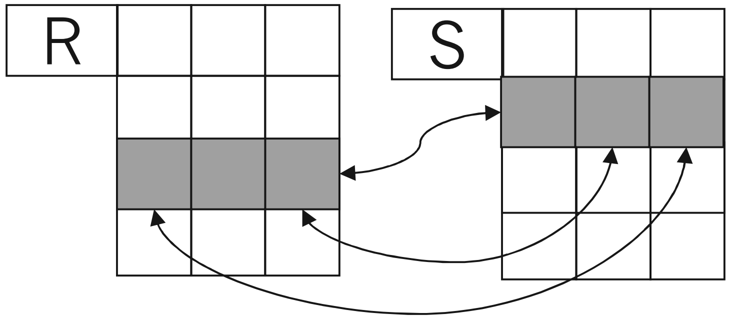
Discussion: Feature-based Methods

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 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

A	B	C	D	E
Max	Michel	m	601- 4839204	601- 4839204
Sam	Adams	m	541- 8127100	541- 8121164

=

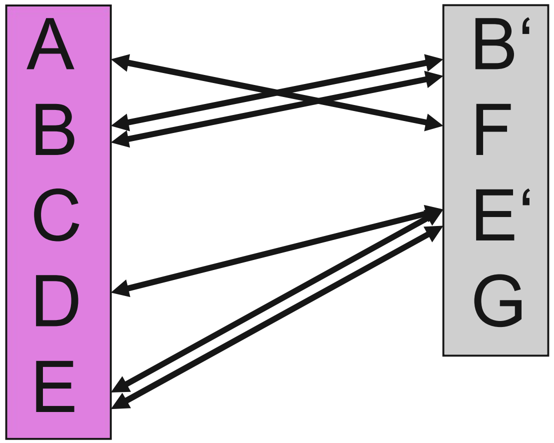
B`	F	E`	G
Michel	maxm	601- 4839204	UNIX
Adams	beer	541- 8127164	WinXP

? ? ? ?

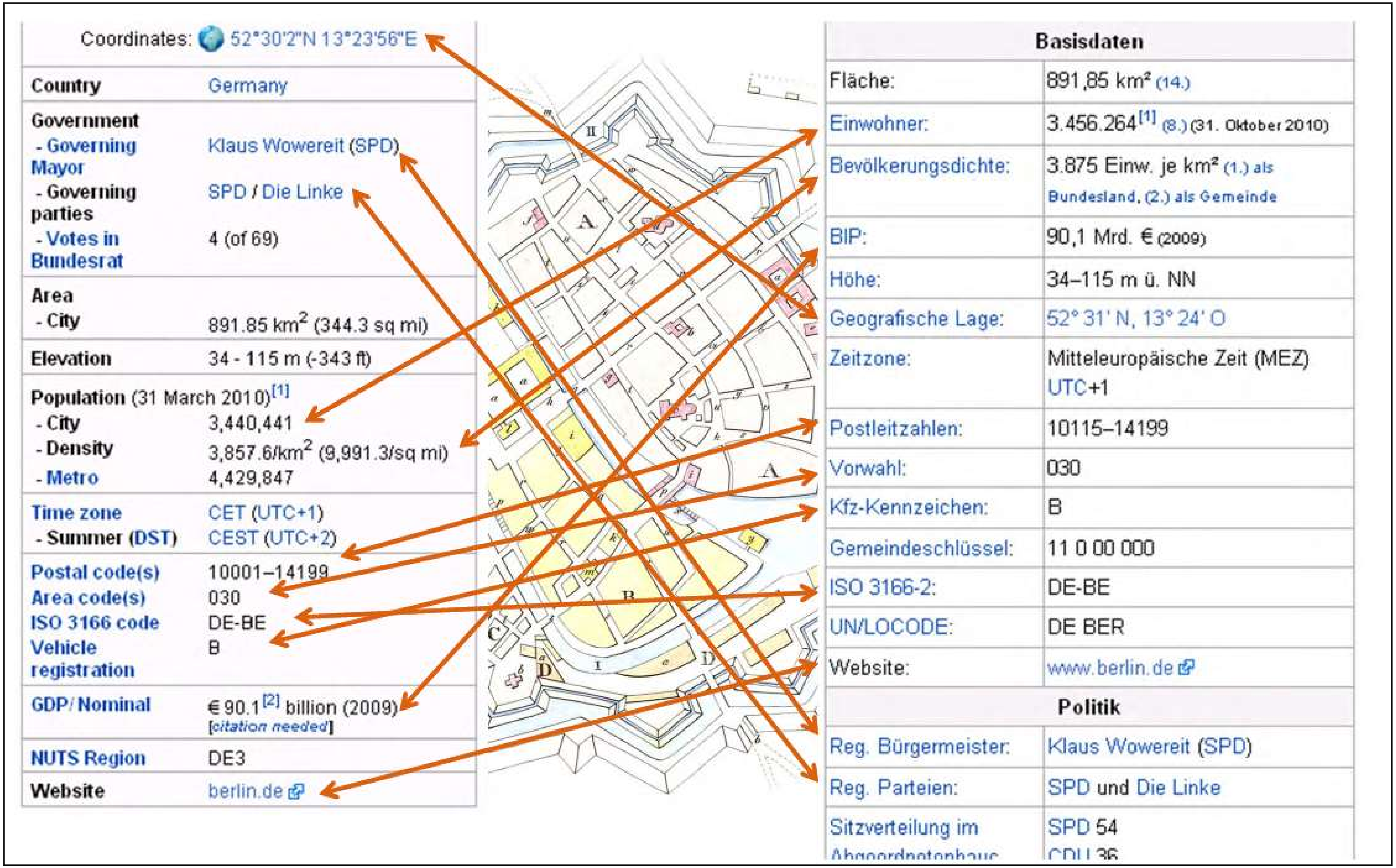
Vote of the two duplicates:

Resulting schema-level correspondences:

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



Using Duplicates for Cross-Language Infobox Matching



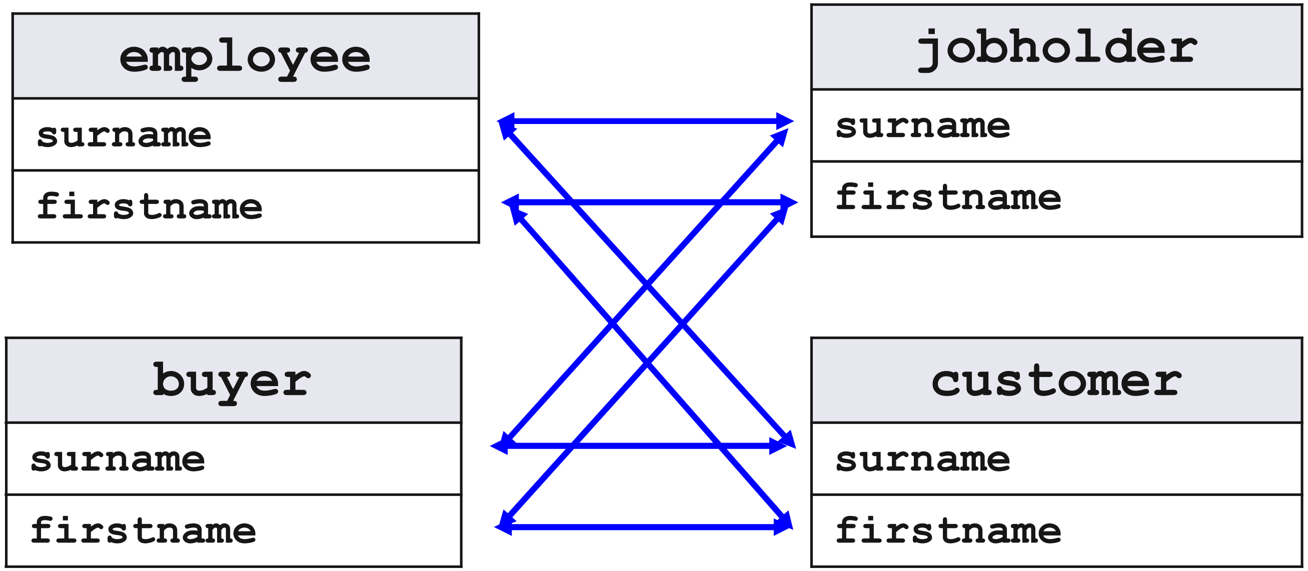
Source: Felix Naumann, ICIQ 2012 Talk

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

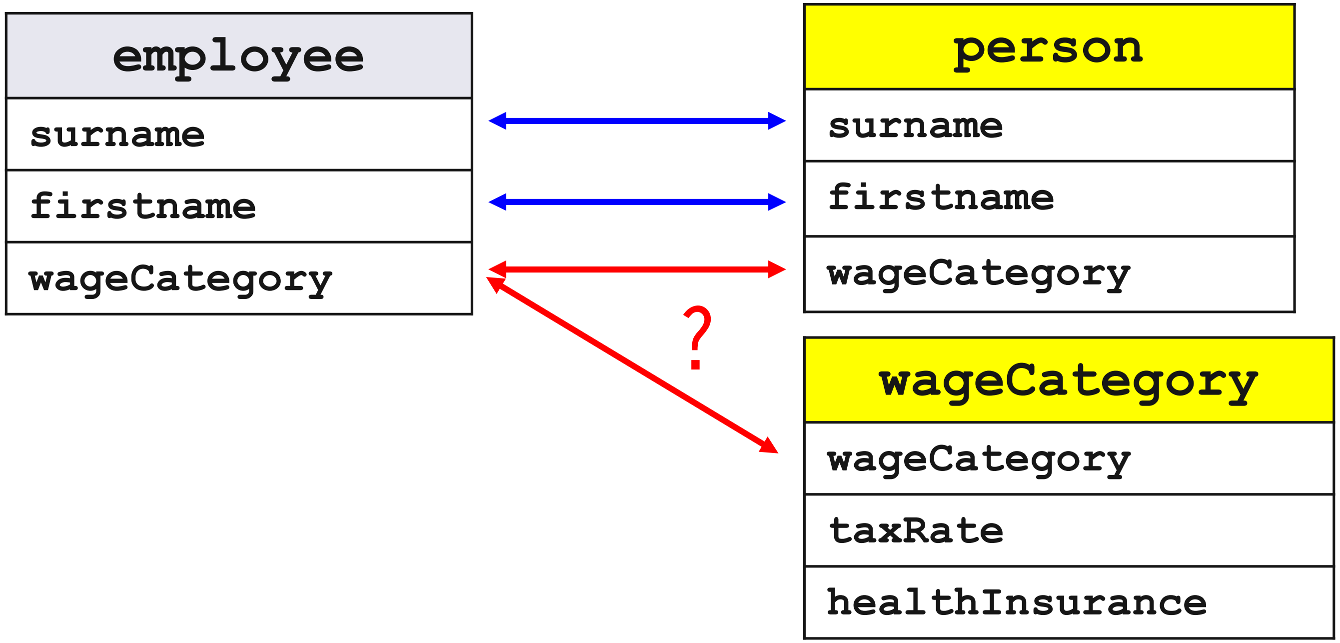
5.2.3 Structure-based Schema Matching Methods

– Addresses the following problem:



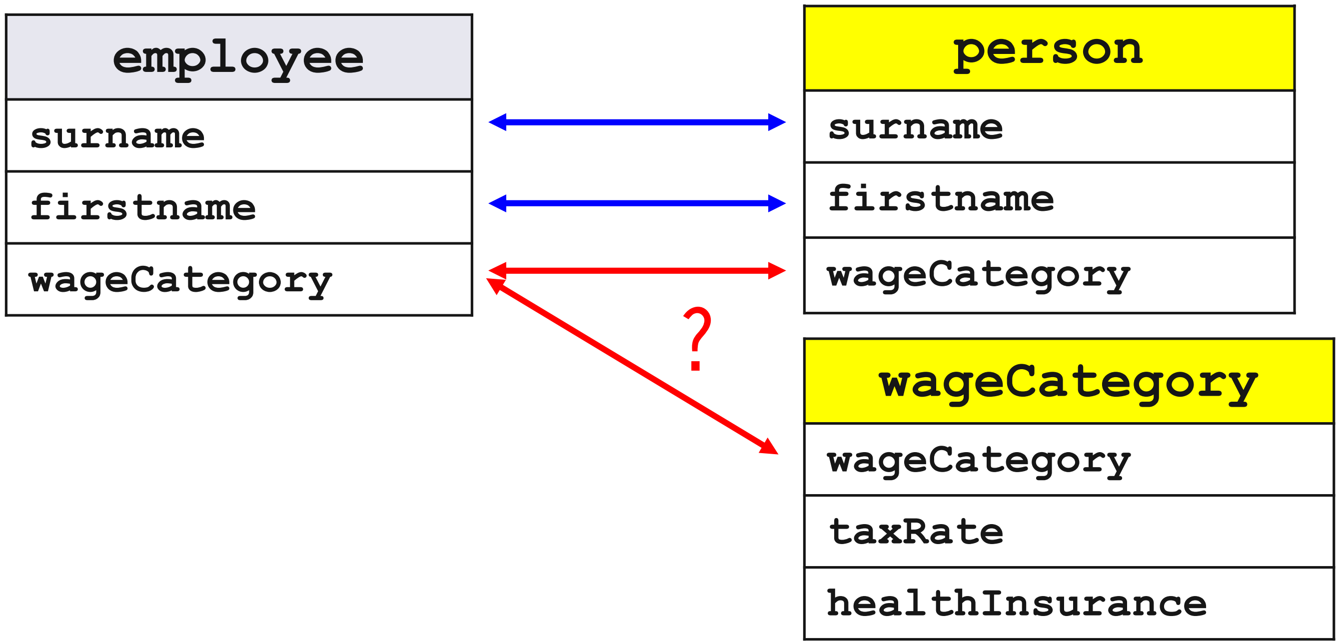
– Attribute-Attribute-Matching
  – Instance-based: Values of all attributes rather similar
  – Label-based: Labels of all attributes rather similar
  – All matchings are about equally good ☹

Better approach: Exploit the Attribute Context



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

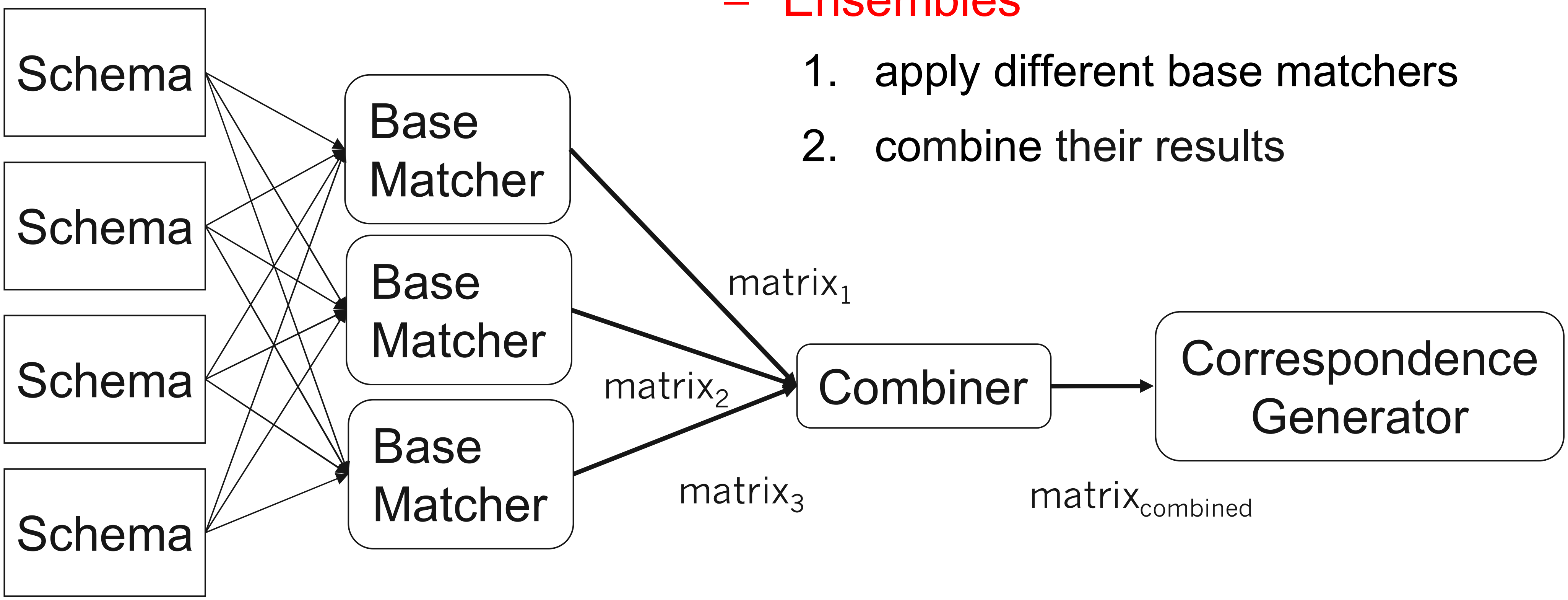
Approach: Spread Similarity to Neighbors



- 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 $320K	James Smith Mike Doan	(305) 729 0831 (617) 253 1429	(305) 616 1822 (617) 112 2315	Fantastic house Great location

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

Combiner

- Average combiner: trusts all matchers the same
- 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 trees for non-linear combiners)
    - we will cover learning weights in detail in chapter on identity resolution
- Alternative: BERT-based Schema Matching
  - Fine-tune transformer which combines similarity calculation and aggregation

Zhang et al.: Schema Matching using Pre-Trained Language Models. ICDE 2023.

5.3 Generating Correspondences from the Similarity Matrix

Input: Matrix containing attribute similarities

Output: Set of correspondences

Correspondence Generator

Local Single Attribute Strategies:

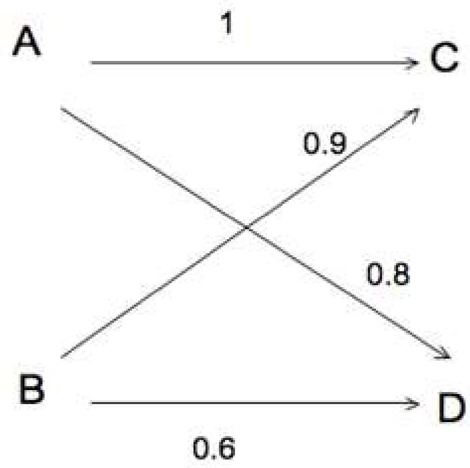
1. Thresholding
   - all attribute pairs with sim above a threshold are returned as correspondences
   - domain expert checks correspondences afterwards and selects the right ones
2. TopK
   - give domain expert TopK correspondences for each attribute
3. Top1
   - directly return the best match as correspondence (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 ≡ C and B ≡ C

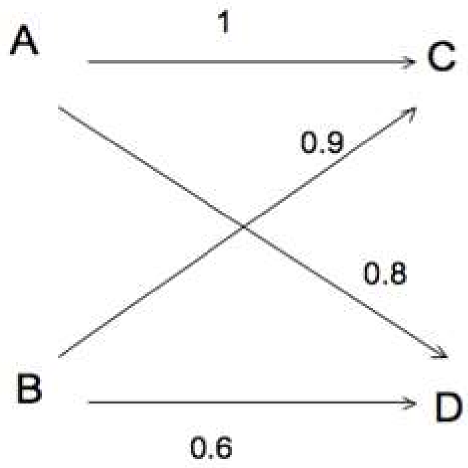
– Approach:
  – find set of bipartite pairs with the maximal sum of their similarity values



– Example:
  – A ≡ D and B ≡ C have the maximal sum of their similarity values
  – Ignores that sim(A,C) = 1

Alternative: Stable Marriage

- Setting: Elements of A = women, elements of B = men
- Sim(i,j) = degree to which $\mathbf{A_i}$ and $\mathbf{B_j}$ desire each other
- Goal: Find a stable match combination between men and women
- A match combination would be unstable if
  - there are two couples $\mathbf{A_i} = \mathbf{B_j}$ and $\mathbf{A_k} = \mathbf{B_l}$ such that $\mathbf{A_i}$ and $\mathbf{B_l}$ want to be with each other, i.e., sim(i,l) > sim(i, j) and sim(i,l) > sim(k,l)
- Algorithm to find stable marriages
  - Let 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

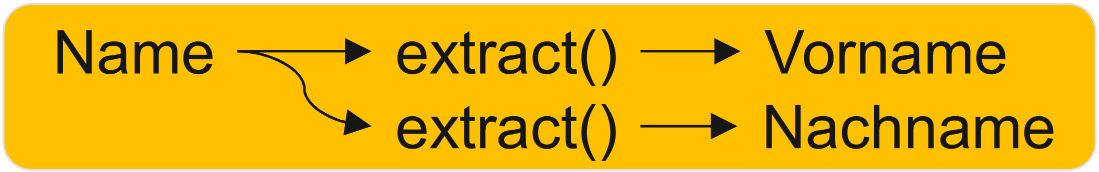


5.4 Finding Many-to-One and One-to-Many Correspondences

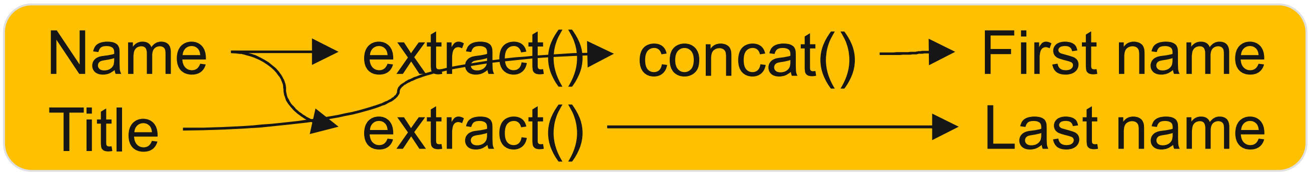
- Up till now all methods only looked for 1:1 correspondences
- But real-world setting might require n:1 and 1:n or even n:m correspondences
- Question:
  - How to combine values?
  - Lots of functions possible.
- Problem:
  - Should we test $1.2 * A + 2 * B - 32 \equiv C$
  - … unlimited search space!

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	state	city	zipcode
320K	532a	2	WA	Seattle	98105
240K	115c	1	FL	Miami	23591

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

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

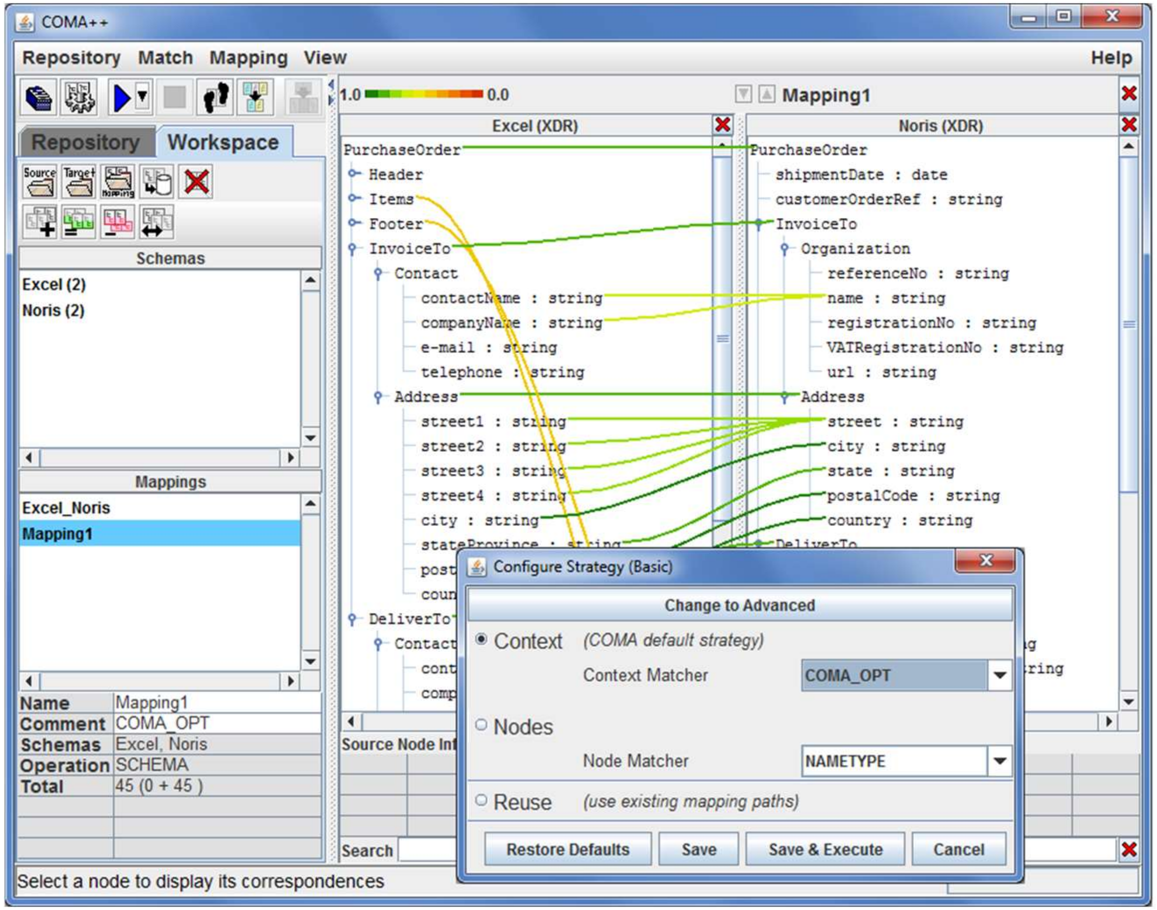
concat(city,zipcode)
Seattle 98105
Miami 23591

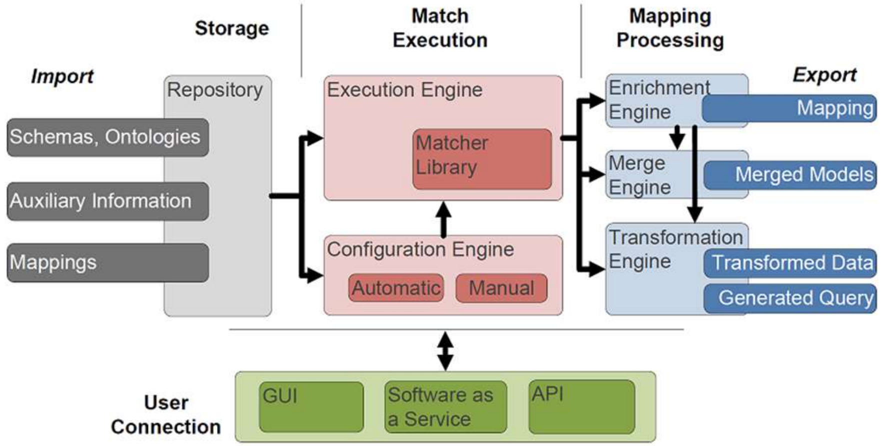
- Best match candidates for address
  - (agent-id,0.7), (concat(agent-id,city),0.75), (concat(city,zipcode),0.9)

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)
- 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
- 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

WIKIDATA

DBpedia

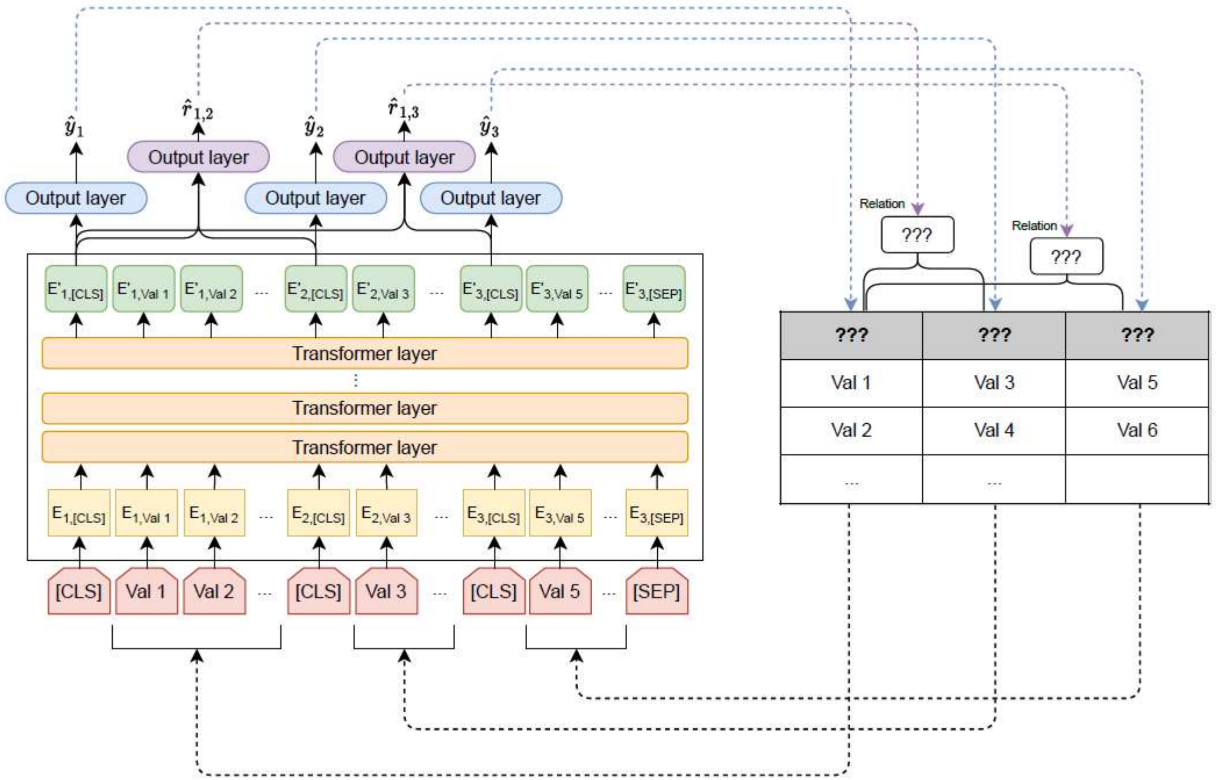
schema.org

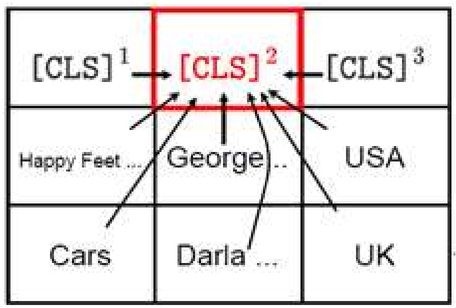
SemTab evaluation campaign: https://www.cs.ox.ac.uk/isg/challenges/sem-tab/

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

– requires lots of task-specific training data for fine-tuning

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
- 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
    - click log analysis of query results
  - Spread the manual integration effort over time
    - pay-as-you-go integration in data lakes

6. Schema Heterogeneity on the Web

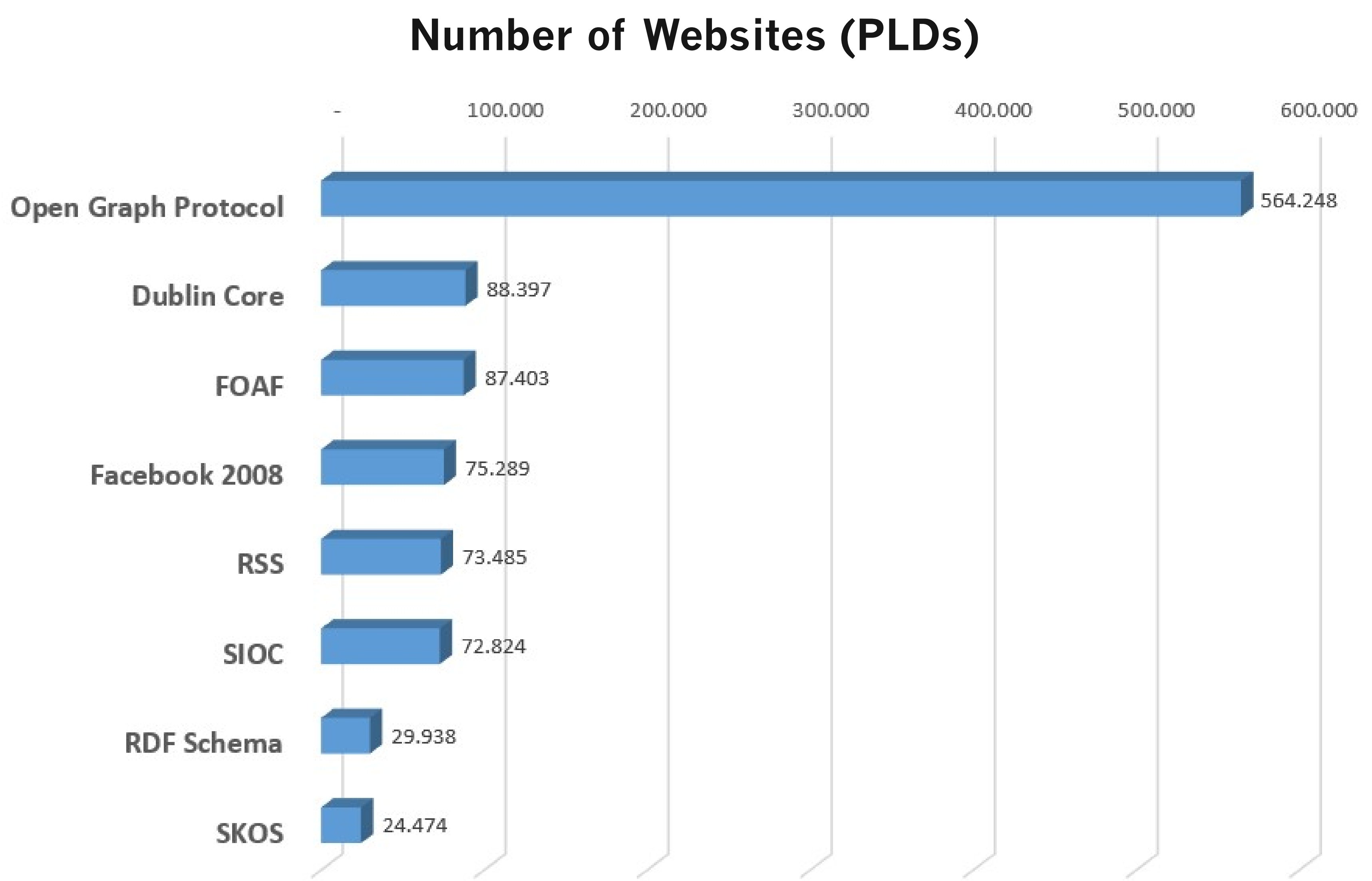
1. Role of Standards
   1. RDFa/Microdata/Microformats
   2. Linked Data
2. Self-Descriptive Data on the Web

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, …

Vocabularies used together with the RDFa Syntax



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

Properties used to Describe Schema.org Products (2020)

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 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
   - **Schema.org** for describing various types of entities
   - **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
   - **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
   - **GeneID** and **UniProt** life science identifiers
   - **XML Schema** for datatypes
   - **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, 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 .
  - 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 .

Publish Correspondences on the Web

Vocabularies are (partly) connected via vocabulary links.

Vocabulary Link

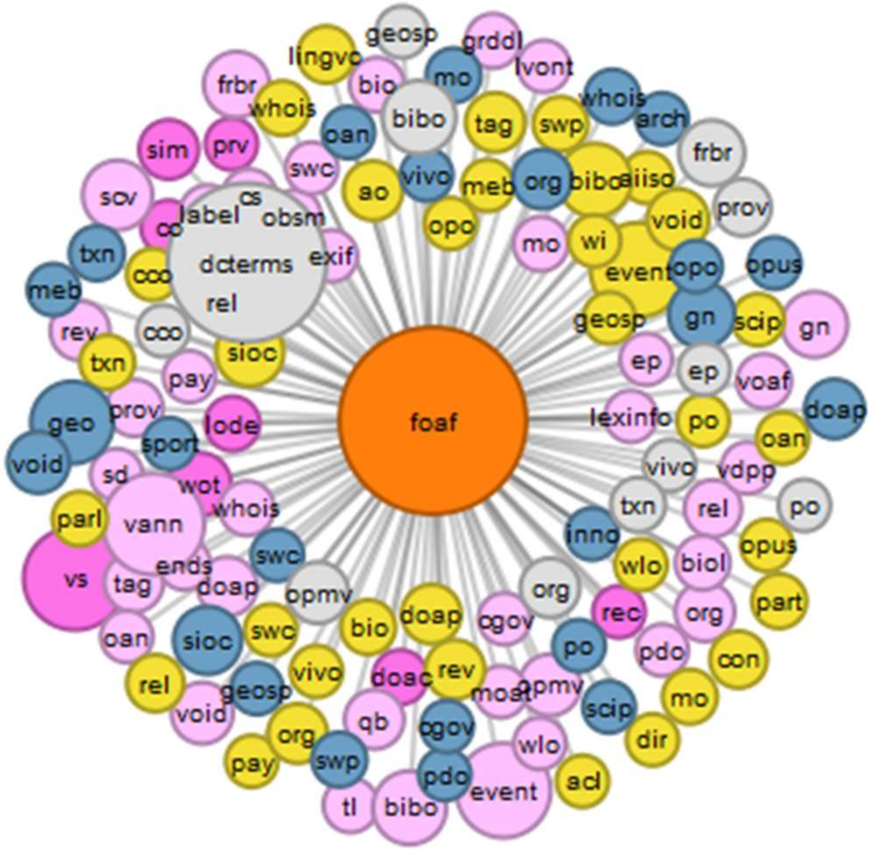
```
<http://dbpedia.org/ontology/Person>
owl:equivalentClass
<http://xmlns.com/foaf/0.1/Person> .
```

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

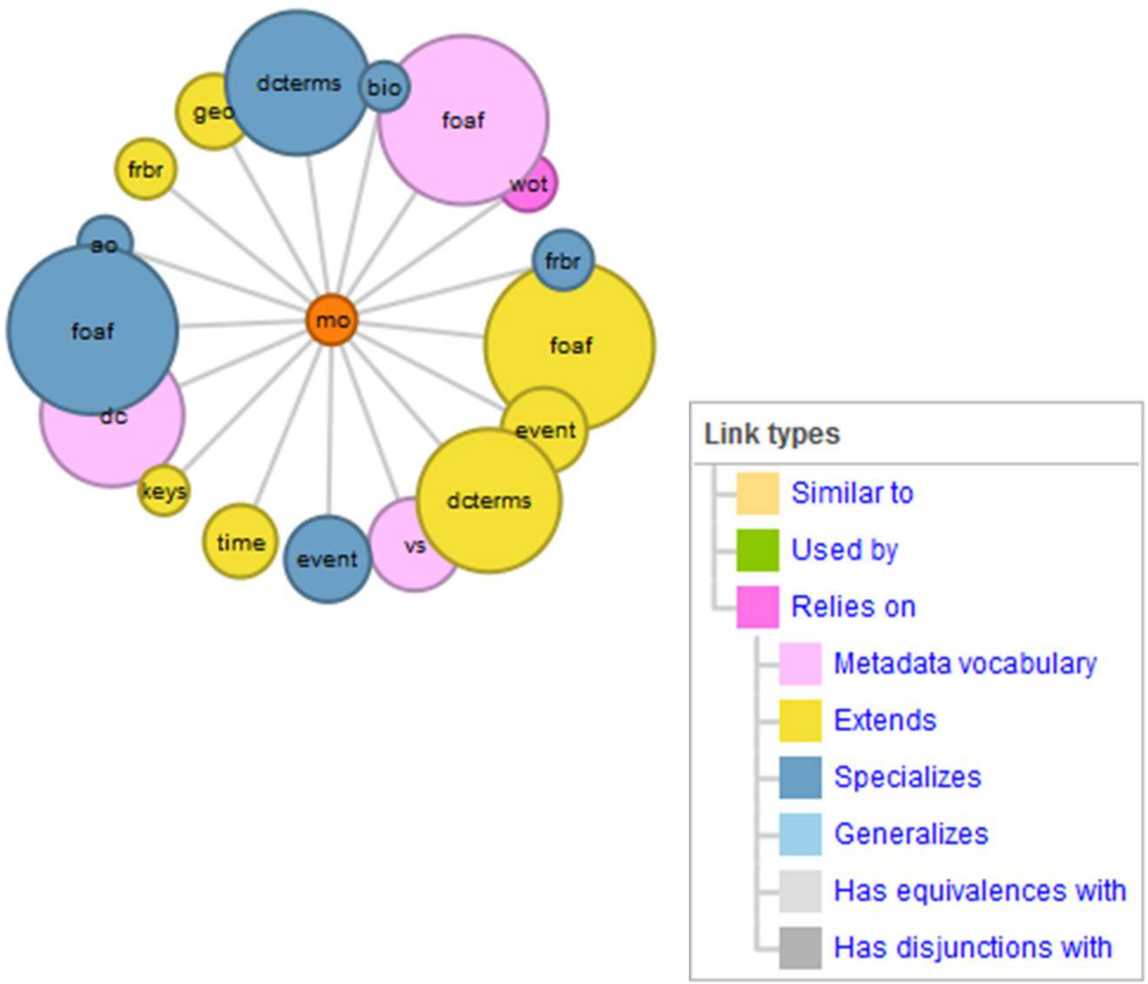
Deployment of Vocabulary Links

Vocabulary links:

Vocabularies referencing "foaf" (119)

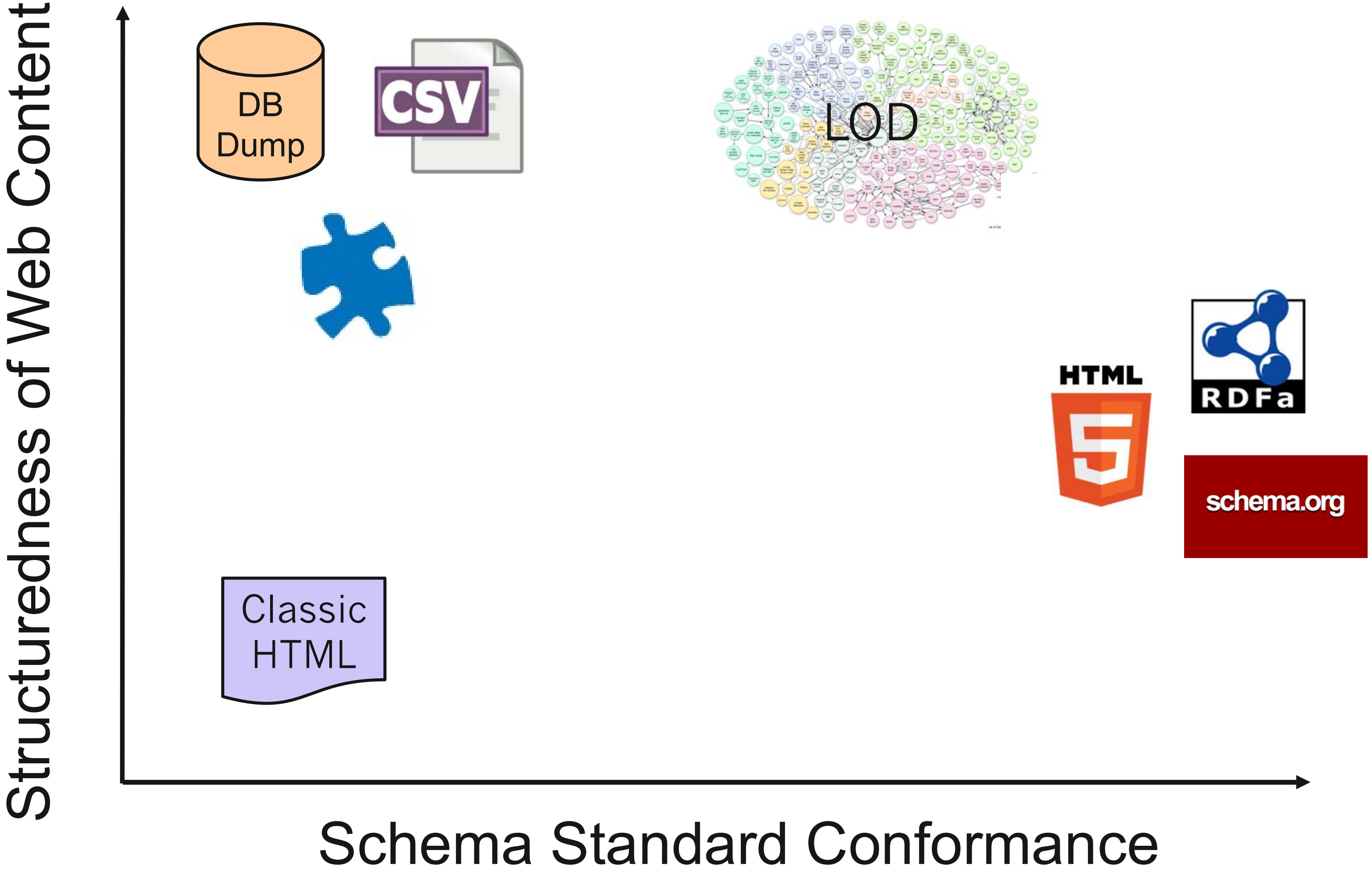


Vocabularies referenced by "mo" (17)



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

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.