

# Data Integration and Analysis

## 04 Schema Matching and Mapping

Matthias Boehm

Graz University of Technology, Austria  
Computer Science and Biomedical Engineering  
Institute of Interactive Systems and Data Science  
BMK endowed chair for Data Management



# Agenda

- **Motivation and Terminology**
- **Schema Detection**
- **Schema Matching**
- **Schema Mapping**

# Motivation and Terminology

# Recap: ETL/EAI Schema Transformations

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <xsl:element name="suppliers">
    <xsl:for-each select="/resultsets/resultset[@Tablename='Supplier']/row">
      <xsl:element name="supplier">
        <xsl:attribute name="ID"><xsl:value-of select="Suppkey"/></xsl:attribute>
        <xsl:element name="Name"><xsl:value-of select="SuppName"/></xsl:element>
        <xsl:element name="Address"><xsl:value-of select="SuppAddress"/></xsl:element>
      </xsl:element>
    </xsl:for-each>
  </xsl:element>
</xsl:template>
</xsl:stylesheet>
```

**Are you kidding me? I have 100s of systems/apps and 1000s\* of tables/attributes**

```
<resultsets>
  <resultset Tablename="Supplier">
    <row>
      <Suppkey>7</Suppkey>
      <SuppName>MB</Suppname>
      <SuppAddress>1035 Coleman Rd</SuppAddress>
    </row>
    <row> ... </row>
  </resultset>
</resultsets>
```



```
<suppliers>
  <supplier ID="7">
    <Name>MB</Name>
    <Address>1035 Coleman Rd</Address>
  </supplier>
  <supplier> ... </supplier>
<suppliers>
```

\* [Rudolf Munz: Datenmanagement für SAP Applikationen, BTW, 2007:  
67,000 tables, 700,000 columns, 10,000 views, 13,000 indexes]

# Schema Matching and Mapping

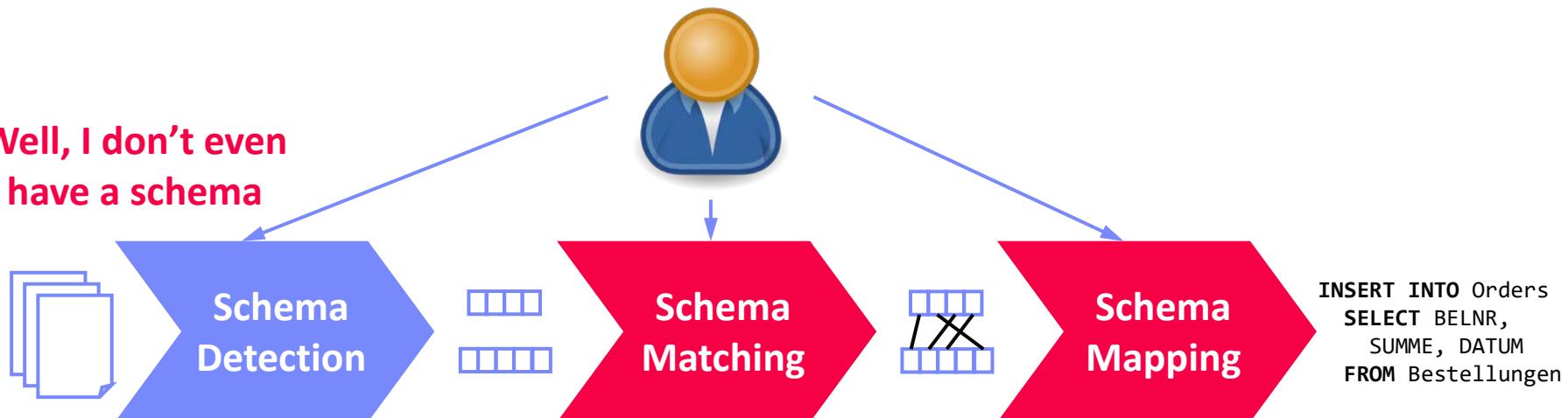
## ■ Schema Matching

- **Given:** two or more relational/hierarchical schemas (and data)
- Find schema mappings in terms of logical attribute correspondences

## ■ Schema Mapping

- **Given:** logical attribute correspondences between schemas
- Compile physical schema mapping programs ([SQL](#), [XSLT](#), [XQuery](#), etc)

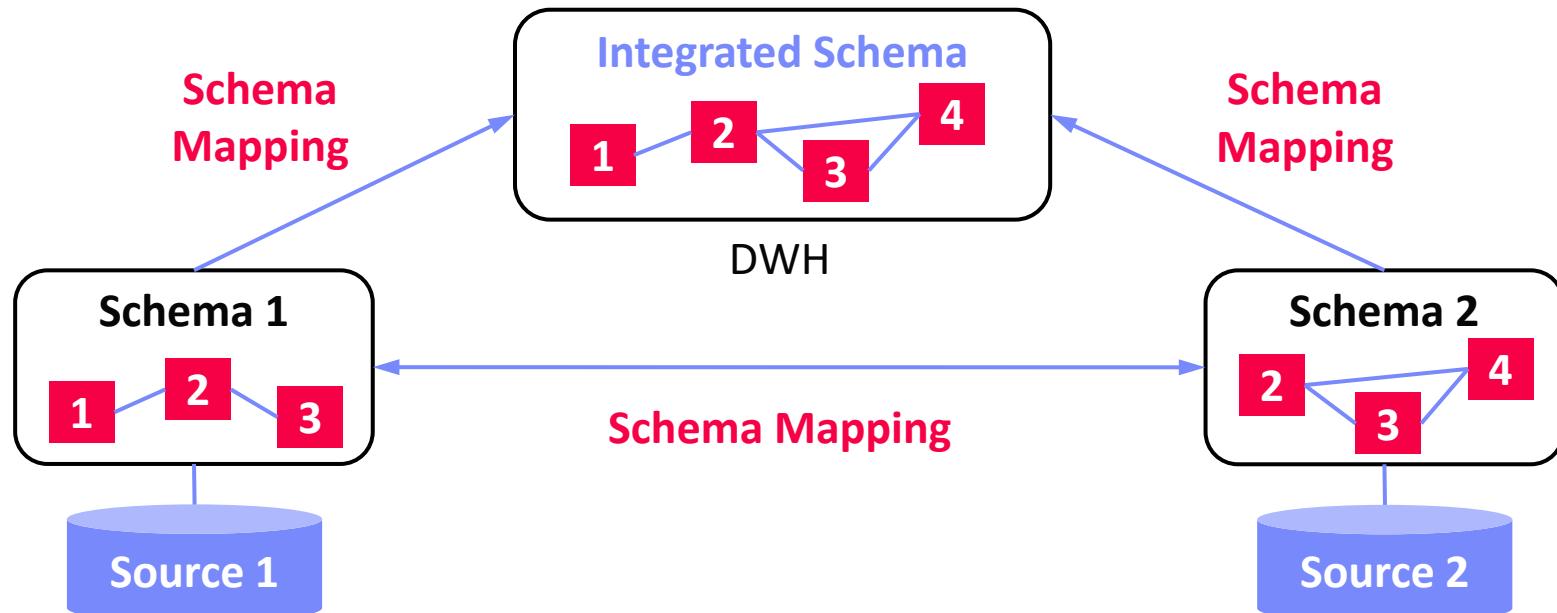
Well, I don't even  
have a schema



# Schema Mapping vs. Schema Integration

## ■ Schema Integration

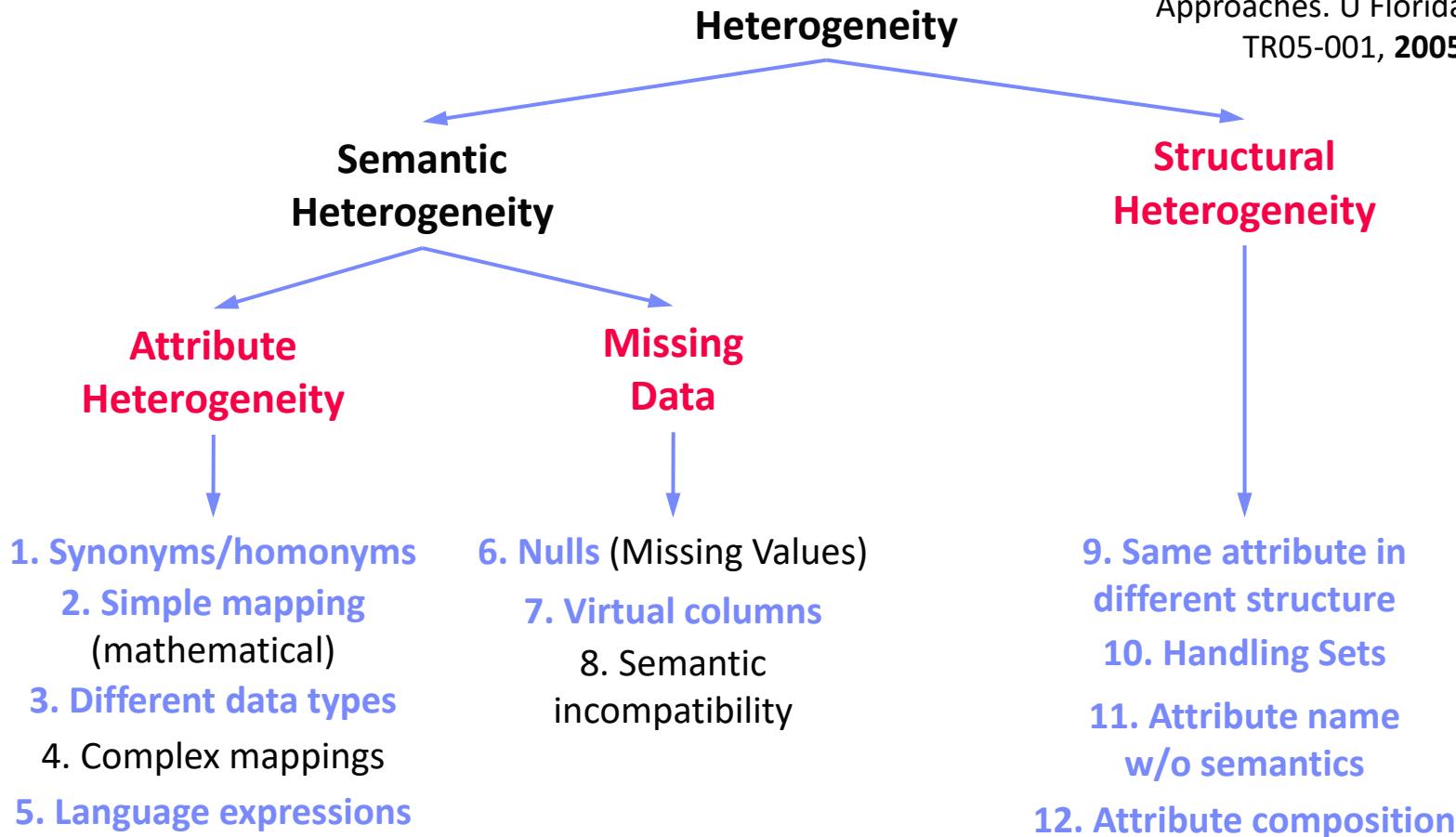
- **Use case:** DWH schema ([lecture 02](#)), virtual/federated DBMS ([lecture 03](#))
- **Schema integration:** map existing schemas into consolidated schema
- **Schema mapping** orthogonal, but both deal with semantic and structural heterogeneity



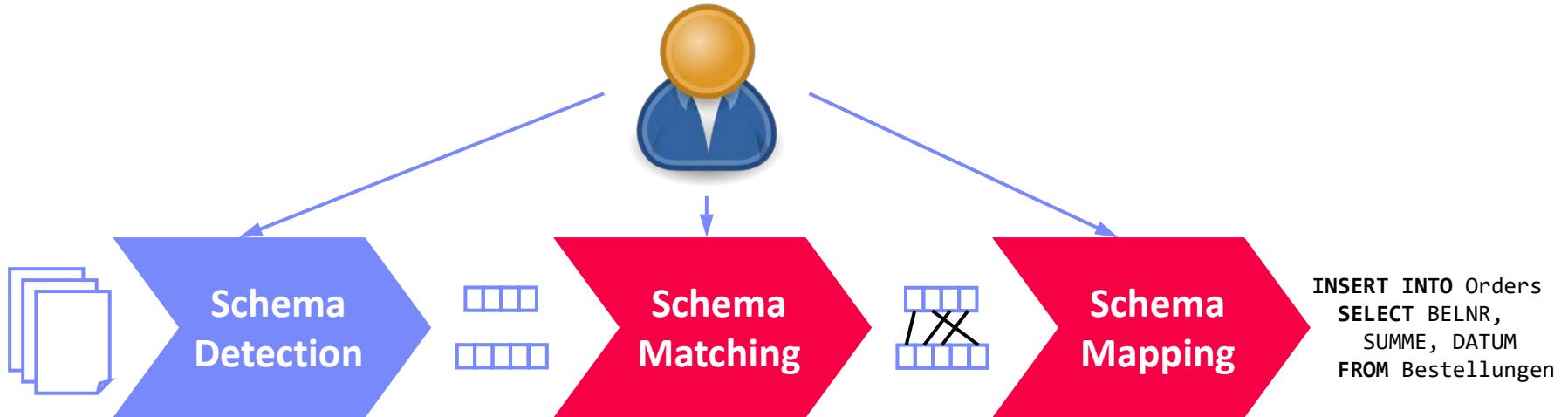
# Recap: Types of Heterogeneity

## → Scope

[J. Hammer, M. Stonebraker, and O. Topsakal: THALIA: Test Harness for the Assessment of Legacy Information Integration Approaches. U Florida, TR05-001, 2005]



# Schema Detection





# Atomic Data Type Detection

## ■ Overview

- **Problem:** Given CSV, JSON, XML files, detect data types of attributes
- **Approach:** Basic extraction rules, regular expressions over data sample

## ■ Example: Schema Inference

- Infer schema for dataframe/dataset columns from sample
- **Basic types:** Decimal, Boolean, Double, Integer, Long, String
- Infer schema from java objects via class reflection

```
StructType(  
    StructField(pid, IntegerType, true),  
    StructField(name, StringType, true),  
    StructField(pos, StringType, true),  
    StructField(jnum, IntegerType, true),  
    StructField(ncid, IntegerType, true),  
    StructField(tid, IntegerType, true))
```



```
./data/players.csv:  
pid,name,pos,jnum,ncid,tid  
4614,Hannes Reinmayr,FW,14,1313,258  
5435,Miroslav Klose,FW,11,789,144  
6909,Manuel Neuer,GK,1,163,308
```



```
Dataset<Row> ds = sc.read()  
    .format("csv")  
    .option("header", true)  
    .option("inferSchema", true)  
    .option("samplingRatio", 0.001)  
    .load("./data/players.csv");
```



# Structure Extraction from Nested Documents

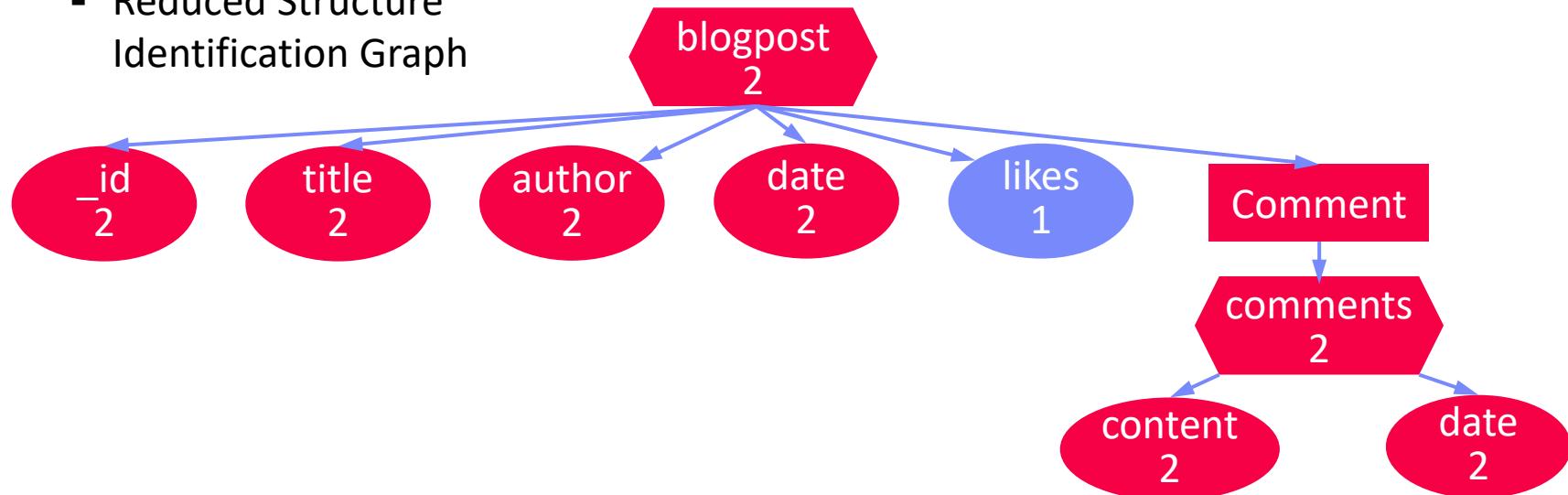
## ■ Structure Extraction Overview

- **Problem:** JSON/JSONNL, XML documents with optional attributes, subtrees
- **Approach:** Scan data, build maximum tree w/ attached meta data
- Meta data := counts or appearances (e.g., document/line IDs)

## ■ Example JSON Schema Extraction

- Structure Identification Graph  
(appearances, data types, etc)
- Reduced Structure  
Identification Graph

[Meike Klettke, Uta Störl, Stefanie Scherzinger: **Schema Extraction and Structural Outlier Detection** for JSON-based NoSQL Data Stores. **BTW 2015**]



# Data Profiling

## ■ #1 Inclusion Dependency (ID) Discovery

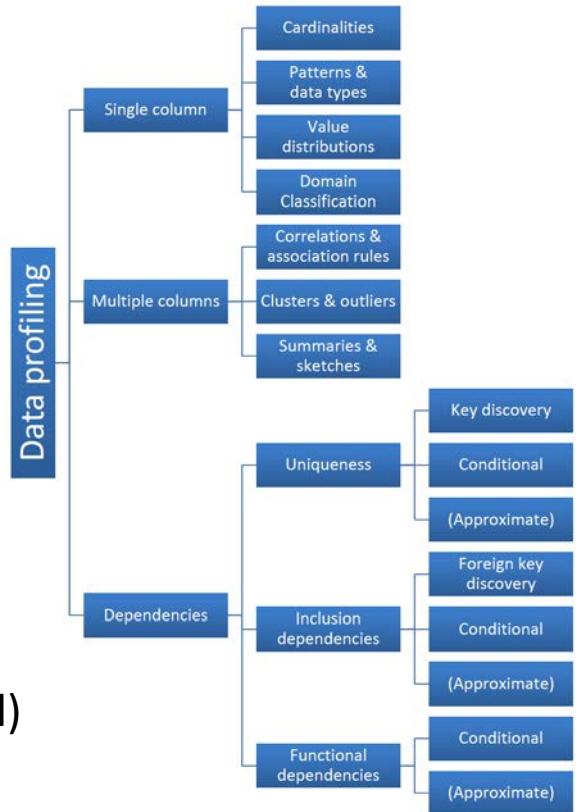
- ID from  $R[X]$  to  $S[Y]$  indicates that all values in  $R[X]$  must appear in  $S[Y]$  → potential indication of **FK relationship** from  $R[X]$  to  $S[Y]$
- **Robust inclusion dependency:**  
 $|R[X] \in S[Y]| / |R[X]| > \delta$
- PK-FK candidate refinement  
(e.g., coverage, similarity, value ranges)

## ■ #2 Functional Dependency (FD) Discovery

- FDs indicative of relational schema (**key candidates, normalization**)
- Discover minimal, non-trivial dependencies
- Extend candidates along **set containment lattice**
  - $A \rightarrow C$  allows pruning  $\{AB\} \rightarrow C$  (non-minimal)
  - $\text{NOT}(A \rightarrow B)$  allows pruning  $A \rightarrow BC$

[Ziawasch Abedjan, Lukasz Golab, Felix Naumann: Data Profiling: A Tutorial. **SIGMOD 2017**]

[Dong Deng et al: The Data Civilizer System. **CIDR 2017**]



# Semantic Data Type Detection

[Madelon Hulsebos et al:  
Sherlock: A Deep Learning  
Approach to Semantic Data  
Type Detection. KDD 2019]



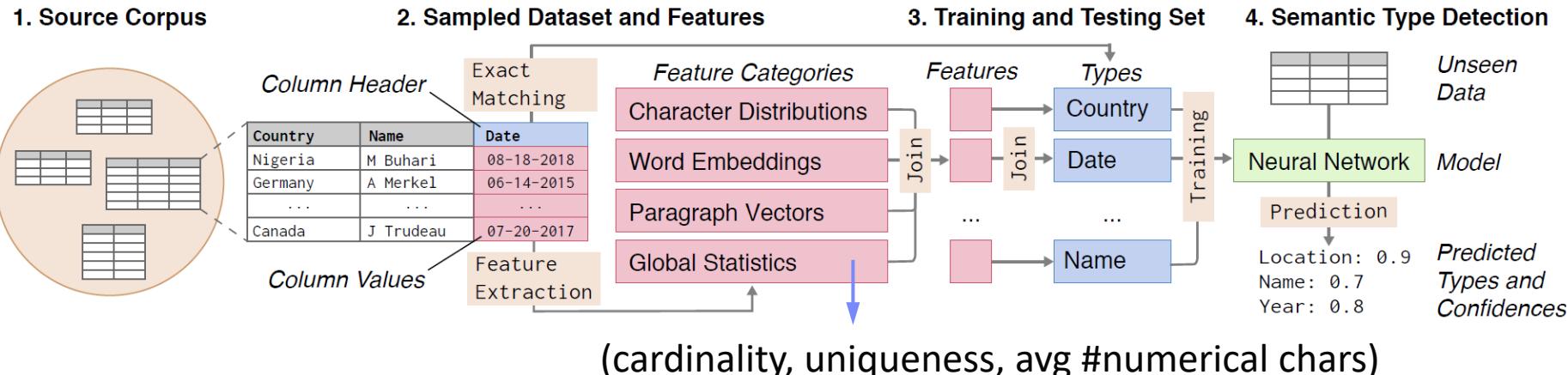
## ■ Problem

- Detect semantic types of columns (e.g., location, date, name)
- **Use cases:** improved data cleaning, schema matching via semantic types

## ■ Sherlock: DNN-based Type Detection

- 78 semantic types from T2Dv2 → VizNet extraction
- 1,588 features from each column, incl global stats
- Formulation of type detection as multi-class classification problem

[http://webdatacommons.org/  
webtables/goldstandardV2.html](http://webdatacommons.org/webtables/goldstandardV2.html)  
(DBpedia, Webtables)



# Schema Enforcement and Evolution

## ■ Schema Enforcement

- Given schema of syntactic and semantic types
- #1 **Raise errors** on invalid data ingestion (ACID consistency and atomicity)
- #2 **Leverage schema constraints** for automatic data cleaning

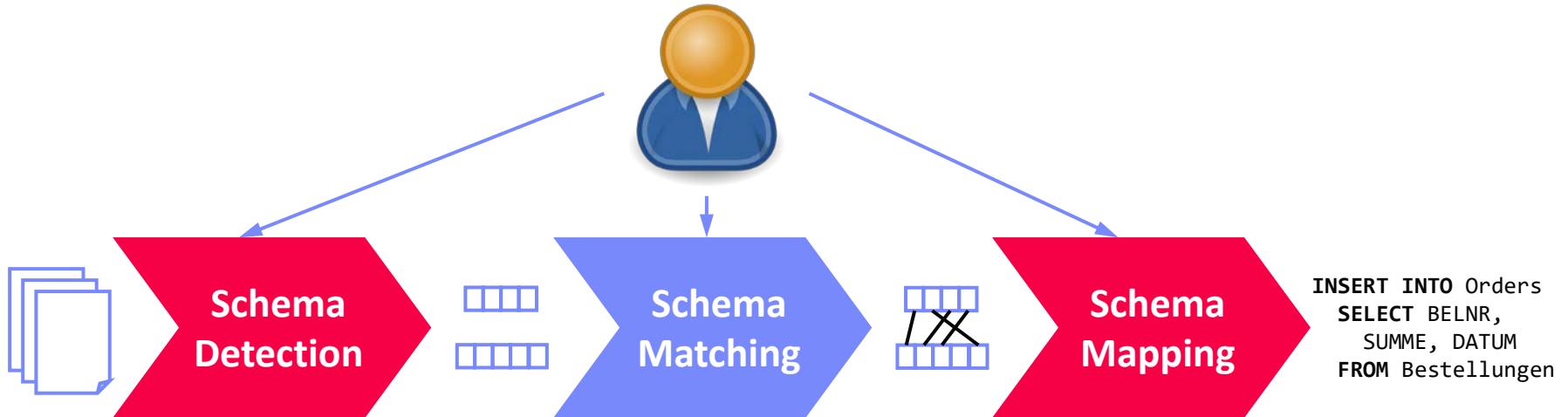
## ■ Schema Evolution

- Incremental modification of schema
- Handle changes of source data (additional columns, different data types)
- Copy vs. in-place modifications (e.g., data type int → string)
- Example: **Delta Lake**

[Andreas Neumann, Denny Lee: Enforcing and Evolving the Schema – Diving into Delta Lake Series, <https://databricks.com/de/blog/2019/09/24/diving-into-delta-lake-schema-enforcement-evolution.html>, 2020]



# Schema Matching



# Overview Schema Matching

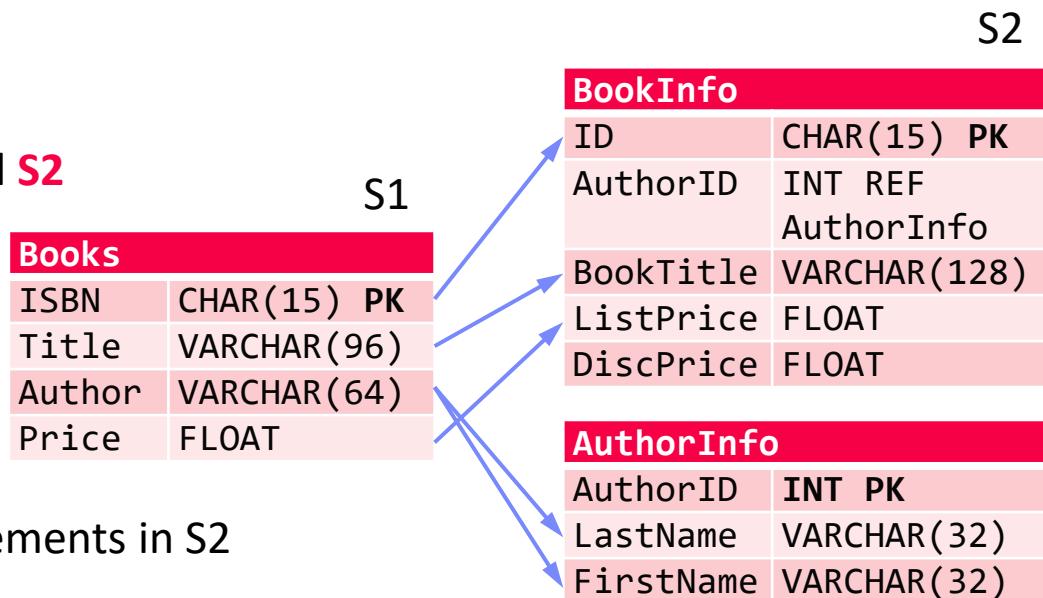
## Motivation

- Large and convoluted schemas (# tables, # attributes, structure)
- **Goal:** generation of matching candidates (refined by user)

## Problem Definition

- **Given:** two schemas **S1** and **S2**
- **Goal:** Generate correspondences between **S1** and **S2**
- **Correspondence:** relationship between M elements in S1 and N elements in S2
- **Mapping expression:** function how elements are related (directional or non-directional)

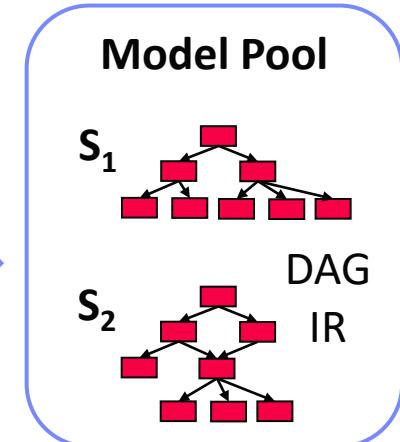
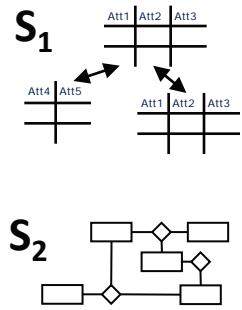
[Philip A. Bernstein, Jayant Madhavan, Erhard Rahm: Generic Schema Matching, Ten Years Later.  
**PVLDB 2011 (test of time award)**]



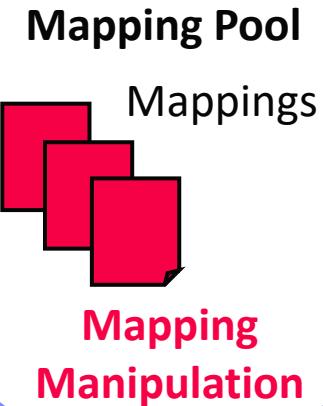


# System Architecture and Matching Process

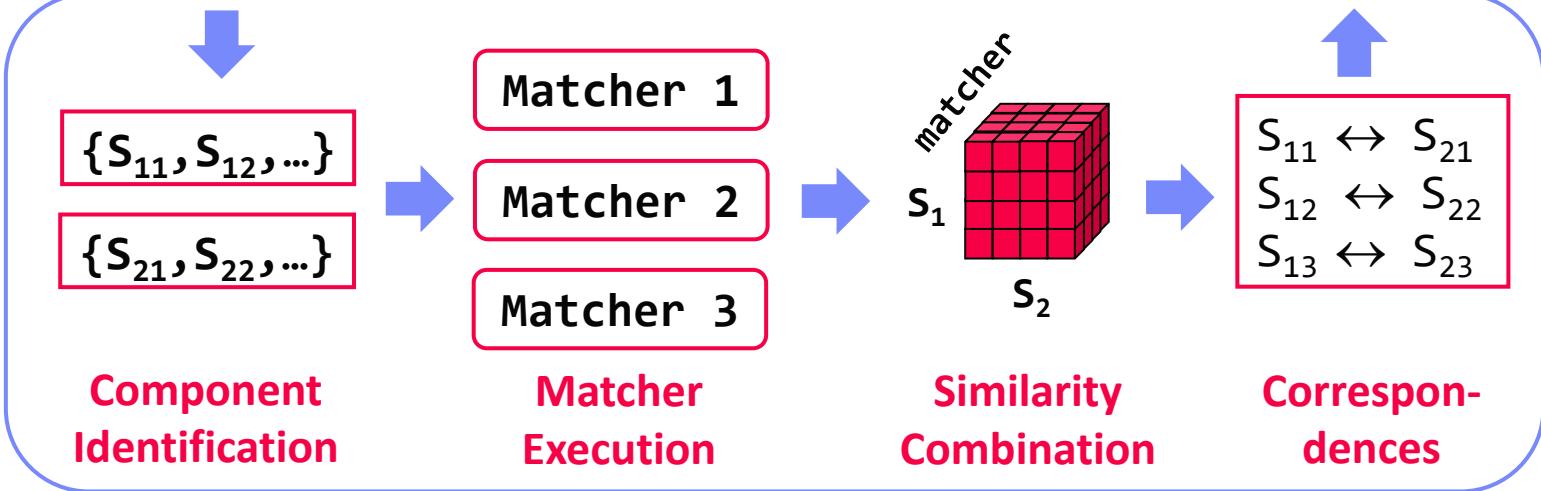
## External Schemas



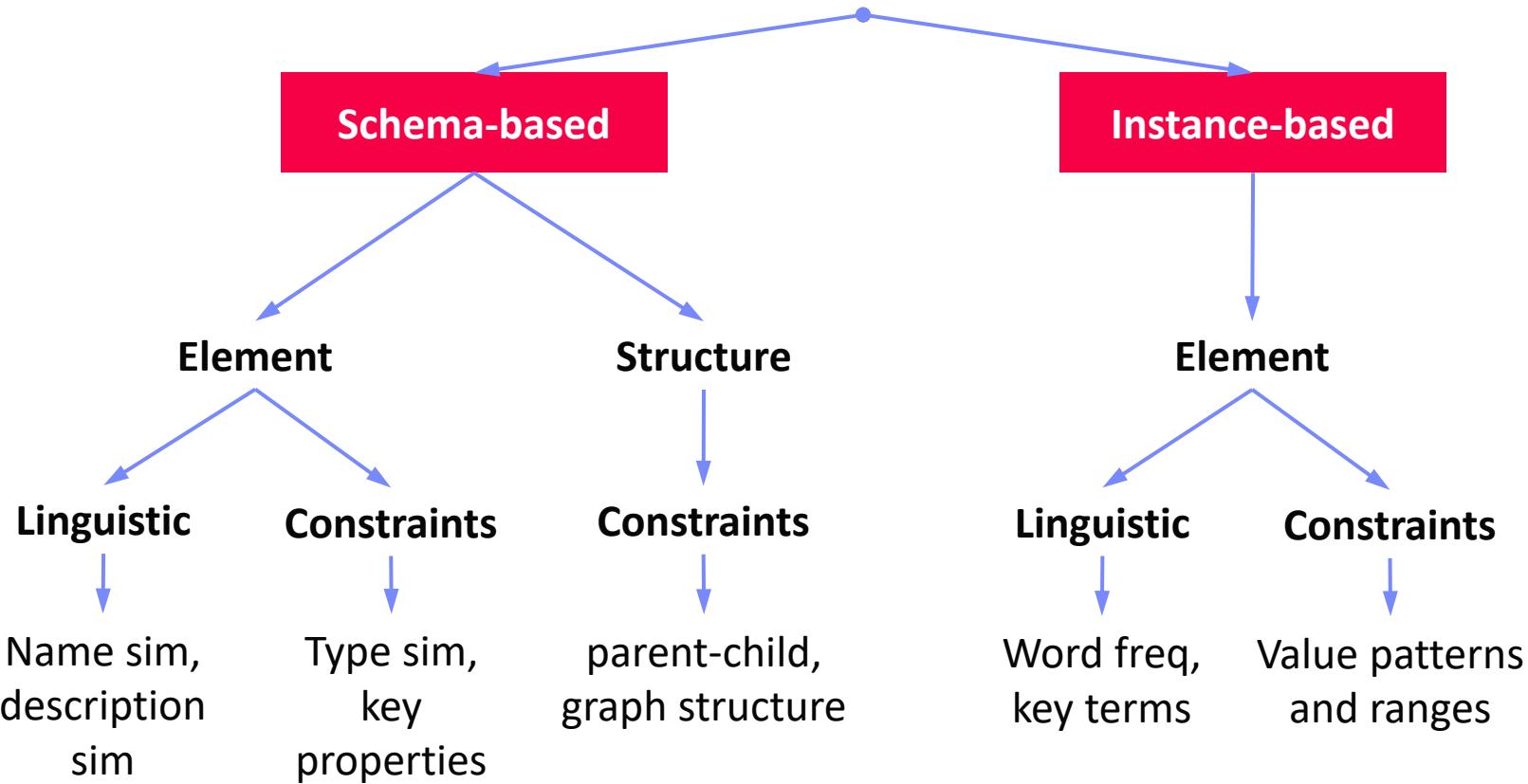
[Erhard Rahm, Hong-Hai Do David Aumueller, Sabine Massmann: Matching Large Schemas with COMA++, 2005]



## Matcher Configs



# Classification of Matching Techniques



- **Cardinalities: 1:1, 1:N, N:M**
- **Combined Matchers (hybrid, composite)**

[Erhard Rahm, Philip A. Bernstein: A survey of approaches to automatic schema matching. VLDB J. 2001] 

# Selected Matchers

Schema Matching

Schema-based

Instance-based

## Linguistic Approaches

- Syntactic
  - Affix (suffix, prefix)
  - N-grams
  - EditDistance
- Semantic
  - Synonyms
  - Hierarchy → taxonomies
  - Language → dictionaries

Name	Street	City
Emma Schmidt	Luisenstrasse 90	Munich, 80333
Klaus Schumann	Koenigsstrasse 7	Dresden, 01199

Firstname	Lastname	Street	Num	ZIP	City
Susanne	Froehlich	Weststrasse	2	01187	Dresden
Thomas	Kunze	Dammweg	45	12437	Berlin

## Example Trigram

- Similarity
 
$$= 2 |S1 \cap S2| / (|S1| + |S2|)$$

$$= 8/16 = 0.5$$

Name					_N	_Na	Nam	ame	me_	e_
Lastname	_L	_La	Las	ast	stn	tna	nam	ame	me_	e_

(alignment irrelevant)



# Selected Matchers, cont.

Schema Matching

Schema-based

Instance-based

## ■ Constraint-Based Approaches

- **Elements:** data types, domains, key attributes, constraints
- **Structure:** relationships between elements, combinations of data types, neighborhood, specialization and composition

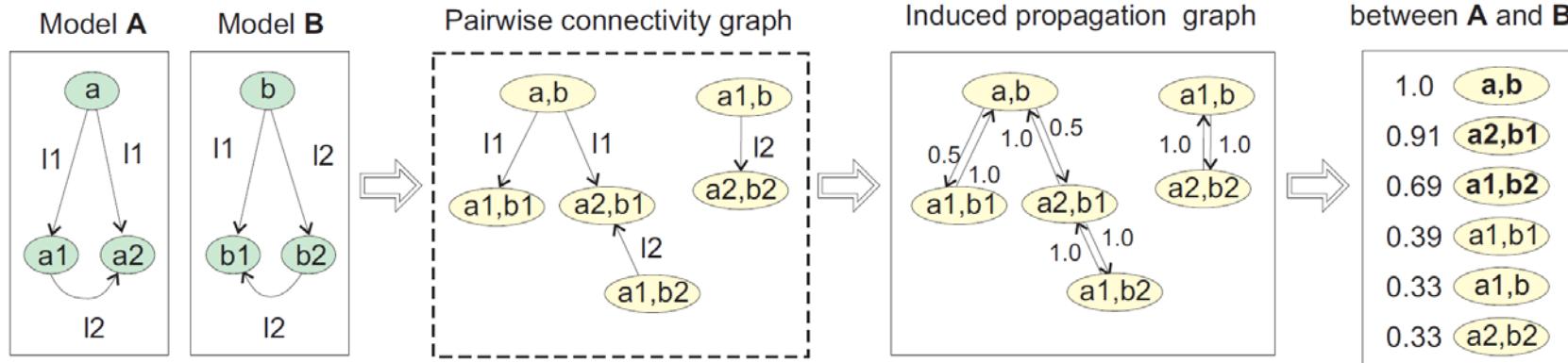
## ■ Example Similarity Flooding

- Create map pairs A x B ([PCG](#))
- Propagation coefficients:  $1/|\text{out}(v)|$  ([IPG](#))
- Initial sim, adjusted by neighborhood sim until fixpoint (converged)

[Sergey Melnik, Hector Garcia-Molina, Erhard Rahm: Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. [ICDE 2002](#)]



Fixpoint values  
for mapping  
between **A** and **B**



# Selected Matchers, cont.

Schema Matching

Schema-based

Instance-based

## ■ Problem Schema-based

- Attribute names might differ vastly (CustNa vs Name, CustSt vs Street)
- **Instance-based matching**, but need for data

## ■ Linguistic Approaches

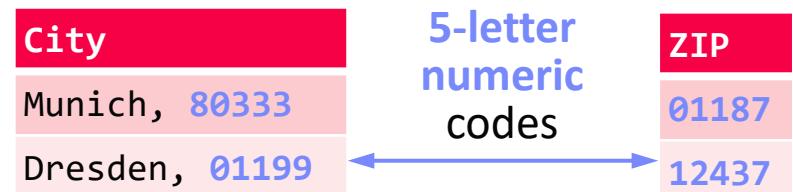
- Word frequencies, bigrams, trigrams, etc
- Keywords, abbreviations

## ■ Constraint-based Approaches

- Elements:** data types and lengths, value domains, patterns
- Structure:** combinations of element constraints

Street	City
Luisenstrasse 90	Munich, 80333
Koenigsstrasse 7	Dresden, 01199

Street	Num	ZIP	City
Weststrasse	2	01187	Dresden
Dammweg	45	12437	Berlin



# Combination of Matchers

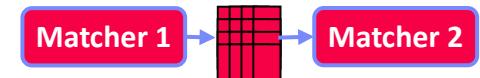
## #1 Reuse

- Exploitation of transitive similarity
- Fast and efficient matching, but potential for lost mappings (e.g., S3 misses attributes)

$$\begin{aligned} S1 \cong S3 \wedge S3 \cong S2 \\ \rightarrow S1 \cong S2 \end{aligned}$$

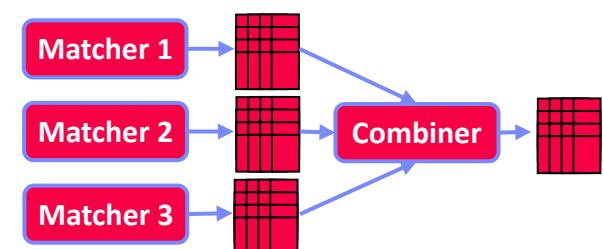
## #2 Refinement

- **Chaining:** matcher  $M_{n+1}$  works on mappings of  $M_n$   
→ efficiency but potential for lost mappings
- **Context-sensitive matching:** Matching context of schema components, matching elements within components



## #3 Composite Matchers

- Select combination of complementary matchers in form of **ensemble**
- **Aggregation** of similarity (e.g., trigram, synonym → avg/min/max)

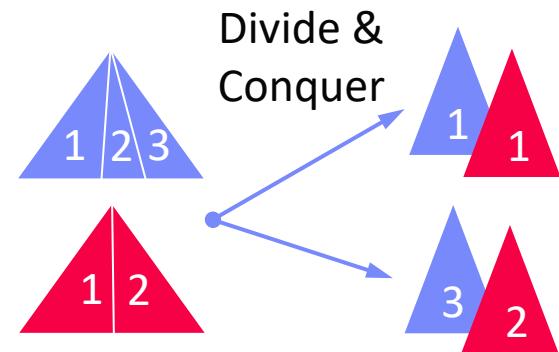


# Efficiency and Scalability

[Philip A. Bernstein, Jayant Madhavan, Erhard Rahm:  
Generic Schema Matching,  
Ten Years Later. **PVLDB 2011**]



- **Problem**
  - Large schemas and matching complexity; all-pair problem  $O(n*m)$
- **#1 Early Search Space Pruning**
  - Faster matchers used to eliminate unlikely matches
  - Reduce schema sizes for expensive matchers
- **#2 Partition-based Matching**
  - Blocking into independent fragments
  - Match elements of similar fragment
- **#3 Parallel Matching**
  - Process different steps/fragments in parallel
- **#4 Custom Similarity Matrices**
  - Sparse matrices (adjacency lists) w/ nesting



[Philip A. Bernstein, Sergey Melnik,  
Michalis Petropoulos, Christoph  
Quix: Industrial-Strength Schema  
Matching. **SIGMOD Record 2004**]





# Excursus: Stable Marriage Problem

Alternatively:  
**Maximal weight  
matching**

## ■ Problem Definition

- For common correspondences, global schema matching relates to the stable marriage (1:1) and hospitals/residence (1:N) problems

## ■ Example Stable Marriage

- Stable matching:** there is no match (A, B), preferable for both A and B over their current matches
- Input are bidirectional similarity (preference ranking)
- Deferred Acceptance Algorithm**

```
while(!converged)
    #1 unmatched As propose to
        highest-ranked, unasked Bs
    #2 Bs accept highest-ranked
        proposal (even if matched)
```

	A,	B,	C,	D
1:	(3,6),	(9,3),	(0,9),	(6,6)
2:	(6,9),	(0,0),	(9,0),	(3,9)
3:	(3,0),	(9,6),	(6,3),	(0,0)
4:	(6,3),	(0,9),	(3,6),	(9,3)



	Group A:	Group B:
1:	B, D, A, C	A: 2, 1, 4, 3
2:	C, A, D, B	B: 4, 3, 1, 2
3:	B, C, A, D	C: 1, 4, 3, 2
4:	D, A, C, B	D: 2, 1, 4, 3

# Excursus: Stable Marriage Problem, cont.

- Example DIA WS19/20 Project
  - #4 Stable Marriage Algorithms in Linear Algebra, Thomas Wedenig

```
138 |     while(sum(S) > 0) {  
139 |         Stripped_preferences = S %*% P  
140 |         Mask_matrix = matrix(0.0, rows=n, cols=n)  
141 |  
142 |         parfor(i in 1:n) {  
143 |             max_proposal = as.scalar(Stripped_preferences[i, as.scalar(proposer_pointers[i])])  
144 |             if(max_proposal != 0) {  
145 |                 proposer_pointers[i] = as.scalar(proposer_pointers[i]) + 1  
146 |                 Mask_matrix[max_proposal, i] = 1  
147 |             }  
148 |         }  
149 |  
150 |         # Hadamard Product  
151 |         Propose_round_results = Mask_matrix * A  
152 |         best_proposers_vector = rowIndexMax(Propose_round_results)  
153 |         prev_best_proposers = rowIndexMax(Result_matrix)
```

- Others:
  - Hospitals and Residents Problem (aka Collage Admission Problem)

# Schema Matching Tools

## ■ Commercial Tools

- Most **message-oriented middleware, EAI, ETL** tools provide mapping UIs (w/ basic string similarity for matching)
- Many data modeling tools also support matching/mapping

## ■ Academic Prototypes



		COMA++	Falcon	Rimom	Asmov	AM	Harmony
year of introduction		2002/2005	2006	2006	2007	2007	2008
Input	<i>relational</i>	✓	-	-	-	-	✓
schemas	<i>XML</i>	✓	-	-	-	(✓)	✓
	<i>ontologies</i>	✓	✓	✓	✓	✓	✓
compreh. GUI		✓	(✓)	?	?	✓	✓
Matchers	<i>linguistic</i>	✓	✓	✓	✓	✓	✓
	<i>structure</i>	✓	✓	✓	✓	✓	✓
	<i>Instance</i>	✓	-	✓	✓	✓	-
use of ext.dictionaries		✓	?	✓	✓	✓	✓

[Erhard Rahm: Towards Large-Scale Schema and Ontology Matching. Schema Matching and Mapping 2011]

# Schema Matching Tools, cont.

**COMA** system for combining match algorithms in a flexible way)



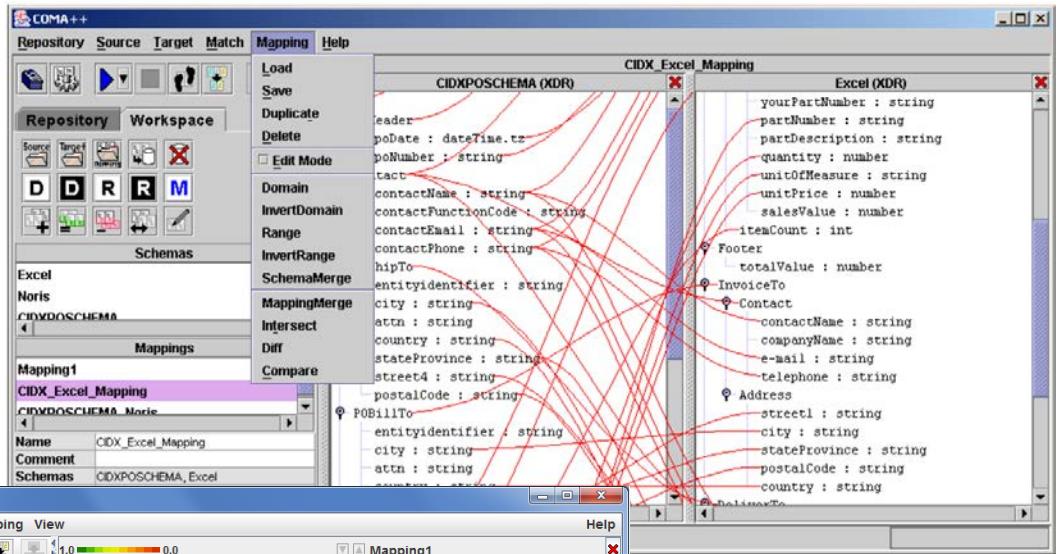
## ■ COMA ++



[Hong Hai Do, Erhard Rahm:  
COMA - A System for Flexible  
Combination of Schema Matching  
Approaches. **VLDB 2002**]



[David Aumueller, Hong Hai Do,  
Sabine Massmann, Erhard Rahm:  
Schema and ontology matching  
with COMA++. **SIGMOD 2005**]

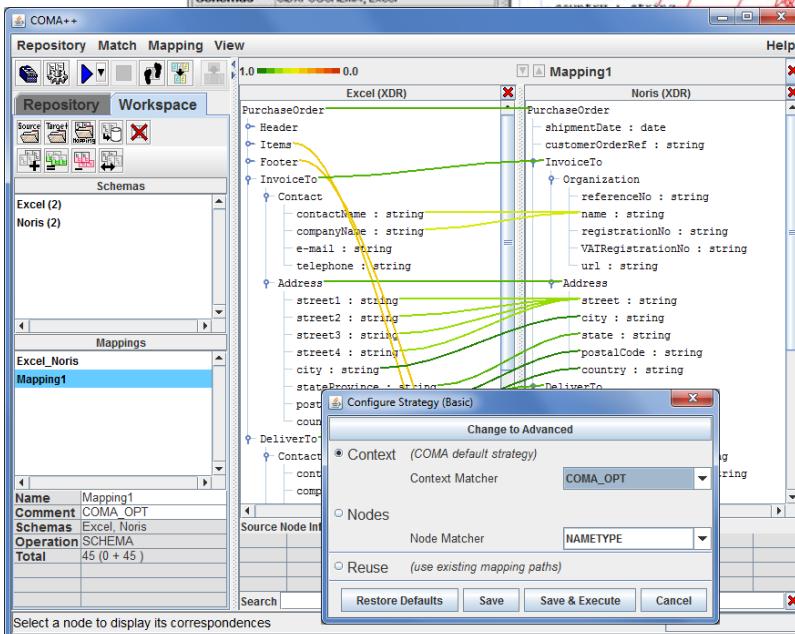


[Credit:

[https://dbs.uni-leipzig.de/  
de/Research/coma.html](https://dbs.uni-leipzig.de/de/Research/coma.html)

## ■ COMA 3.0

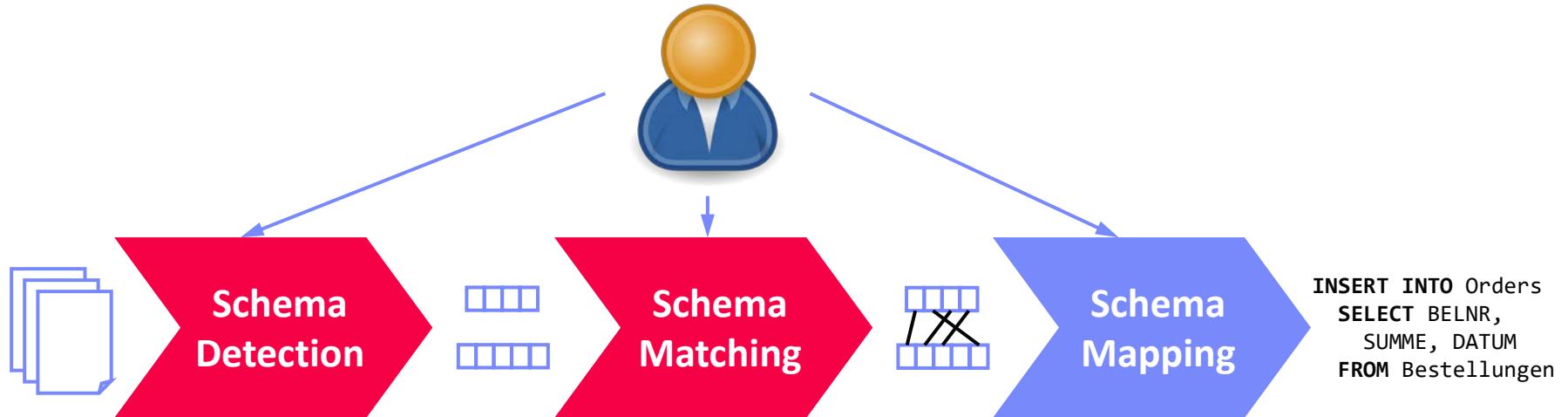
- 2011/2012
- Ontology Merging
- Workflow Management



[Felix Naumann: Informationsintegration –  
Schema Mapping, HPI Lecture, 2012]



# Schema Mapping



# Schema Mapping Overview

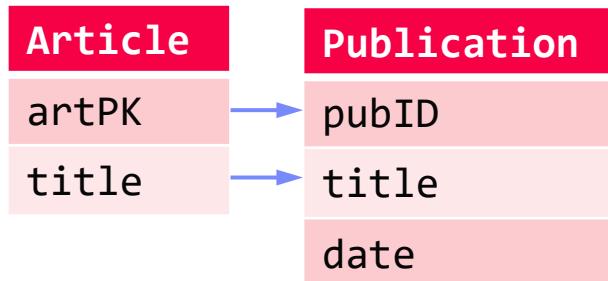
## ■ Problem

- **Given:** two schemas w/ high-level mapping
- Generate concrete transformation program/query (SQL, XSLT, XQuery)

## ■ Schema Mapping Process (systematic lowering)

- **High-level mapping:** intra- and inter-schema correspondences
- **Low-level mapping:** mapping that ensures consistency with constraints of target schema and user intent (interpretation)
- **Transformation query:** transformation program from one into the other schema, backend-specific (query generation)

## ■ Example

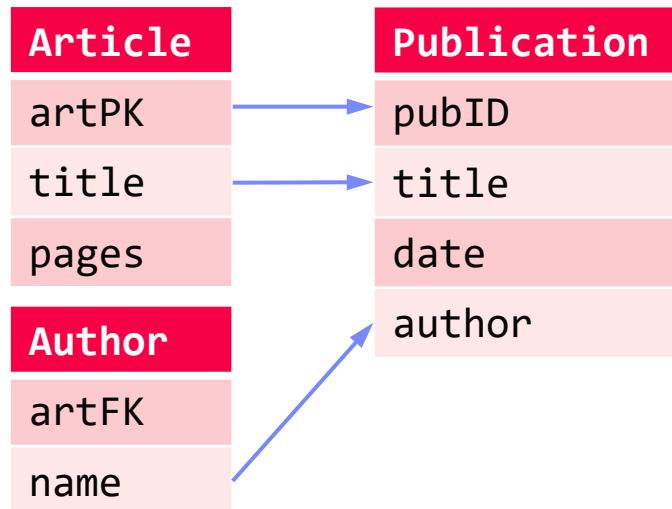


### Transformation Query:

```
INSERT INTO Publication
SELECT artPK, title, NULL
FROM Article
```

# Mapping Interpretations

- Without Intra-Schema Correspondences

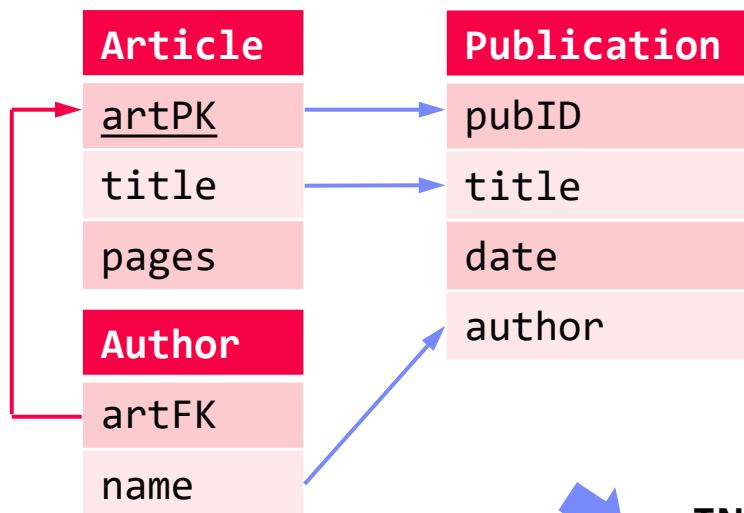


```
INSERT INTO Publication
  (SELECT artPK AS pubID,
    title AS title,
    NULL AS date,
    NULL AS author
  FROM Article)
UNION ALL
  (SELECT NULL AS pubID,
    NULL AS title,
    NULL AS date,
    name AS author
  FROM Author)
```



# Mapping Interpretations, cont.

- With Intra-Schema Correspondences



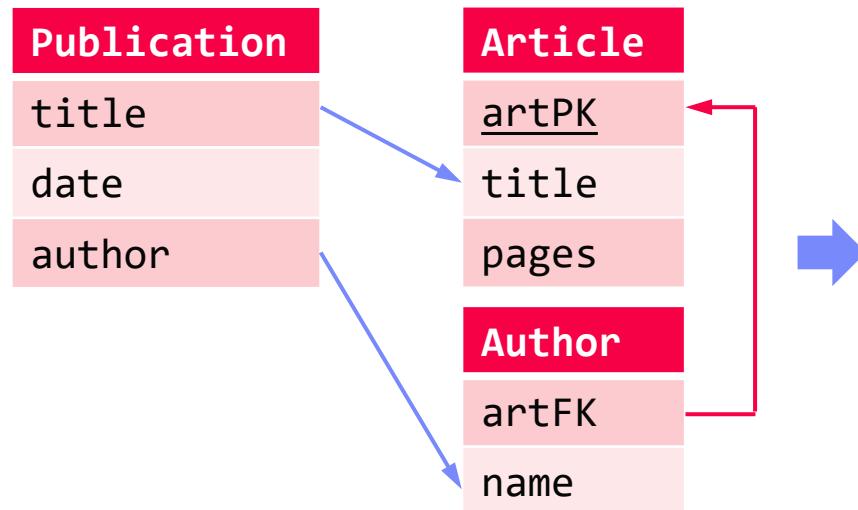
```
INSERT INTO Publication
SELECT artPK, title,
      NULL, name
FROM Article, Author
WHERE artPK = artFK
```

**Note:** Inner join acts as filter (no articles w/o authors, NOT NULL constraints)

```
INSERT INTO Publication
SELECT artPK, title, NULL, name
FROM Article LEFT OUTER JOIN Author
ON artPK = artFK
```

# Mapping Interpretations, cont.

- From Denormalized to Normalized Form



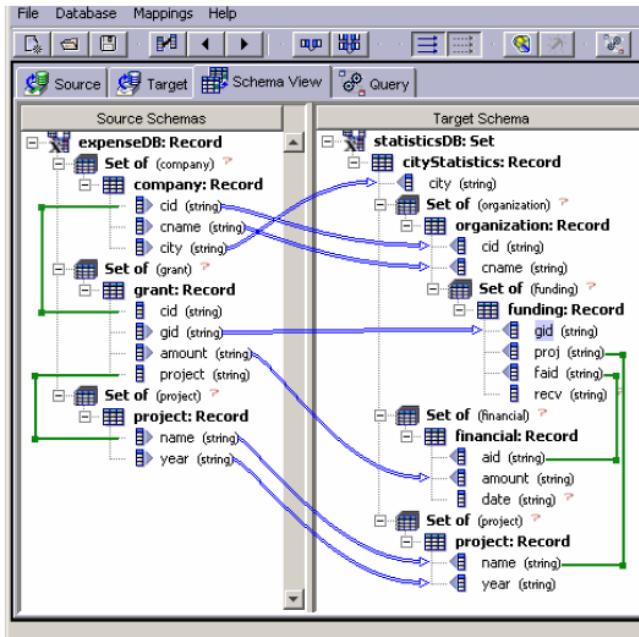
```
INSERT INTO Article
SELECT DISTINCT
    SK(title), title, NULL
FROM Publication
```

```
INSERT INTO Author
SELECT SK(title), author
FROM Publication
```

- Skolem Function (SK):** unique key generation from tuple values  $t[X]$  of an attribute set  $X$

# Schema Matching Tools

- **Clio (IBM Research – Almaden)**



This screenshot shows the Clio industrial tool interface. It features three main panes: "Source Schemas" (left), "Target Schema" (right), and "Query" (bottom). The "Source Schemas" pane shows two schemas: S1 (Record) and S3 (Set). The "Target Schema" pane shows a single schema: ARTICLEAUTHOR (Record). A context menu is open over the S3 schema, with "Attribute Match" selected. The "Query" pane displays an XQuery script for mapping between the source and target schemas. The XQuery code includes various declarations like \$x0L1, \$x1L1, and \$x2L1, and it defines mappings for fields like cid, cname, gid, aid, and bookid. At the bottom, buttons for "Preview", "Execute Query", and "Copy to Clipboard" are visible.

[Mauricio A. Hernández, Renée J. Miller, Laura M. Haas: Clio: A Semi-Automatic Tool For Schema Mapping. SIGMOD 2001]

[Laura M. Haas, Mauricio A. Hernández, Howard Ho, Lucian Popa, Mary Roth: Clio grows up: from research prototype to industrial tool. SIGMOD 2005]

# Summary and Q&A

- **Schema Detection**
- **Schema Matching**
- **Schema Mapping**

[Philip A. Bernstein, Sergey Melnik:  
Model Management 2.0: Manipulating  
Richer Mappings. **SIGMOD 2007**]



“Given the existence of all these tools, **why is it still so labor-intensive to develop engineered mappings?** To some extent, it is an **unavoidable consequence of ambiguity** in the meaning of the data to be integrated. If there is a specification of the schemas, it often says little about integrity constraints, units of measure, data quality, intended usage, data lineage, etc. Given that the specification of meaning is weak and the mapping must be precisely engineered, it seems hopeless to fully automate the process anytime soon. A human must be in the loop.”

## ■ Next Lectures (**Data Integration Architectures**)

- **05 Entity Linking and Deduplication** [Nov 05]
- **06 Data Cleaning and Data Fusion** [Nov 12]
- **07 Data Provenance and Blockchain** [Nov 19]

The 80%  
Argument

