Slides Credit: Matthias Boehm, Shafaq Siddiqi

Data Integration and Large Scale Analysis

04 Schema Matching and Mapping

Agenda

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

Motivation and Terminology

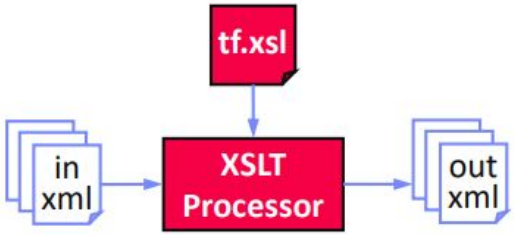
ETL/EAI Schema Transformations

Problem:

- XML often used as **external and internal data representation (BaseX, Oracle DB, MongoDB, etc.)**
- Different schemas (message types) → **requires mapping**

XSLT Overview

- XSLT processor transforms input XML document according to XML stylesheet to output XML documents
- Subtree specifications via XPath, loops, branches built-in functions for text processing, etc
- **CSV** and **JSON** input/output possible



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>
```

```
<resultssets>
  <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]

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

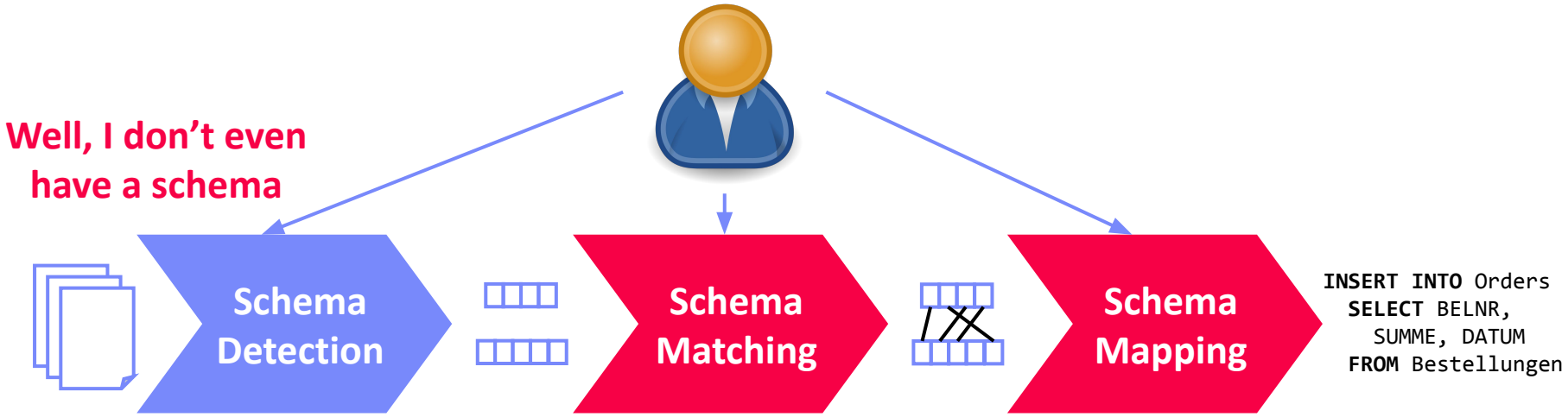
```
<resultssets>
  <resultset Tablename=“Supplier”>
    <row>
      <Suppkey>7</Suppkey>
      <SuppName>MB</Suppname>
      <SuppAddress>1035 Coleman Rd</SuppAddress>
    </row>
    <row> … </row>
  </resultset>
</resultsets>
```

```
<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]

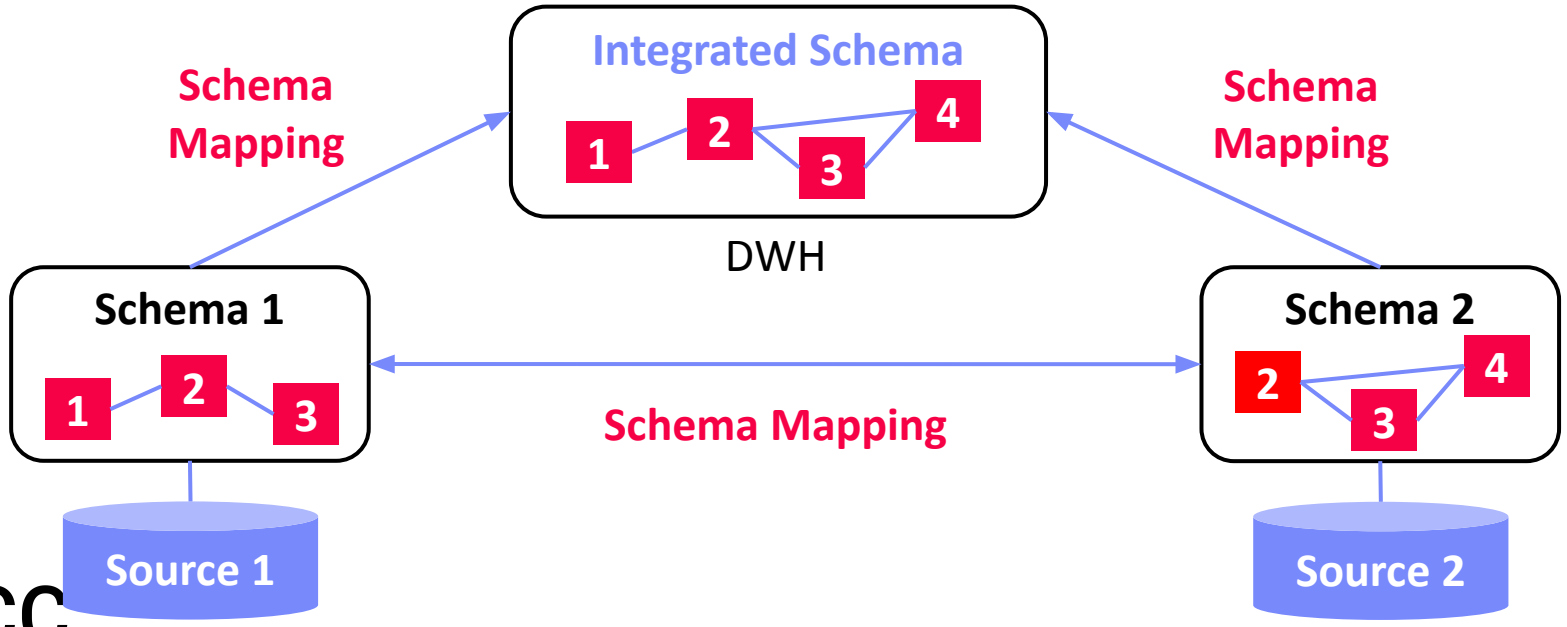
Schema Matching and Mapping

- **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)

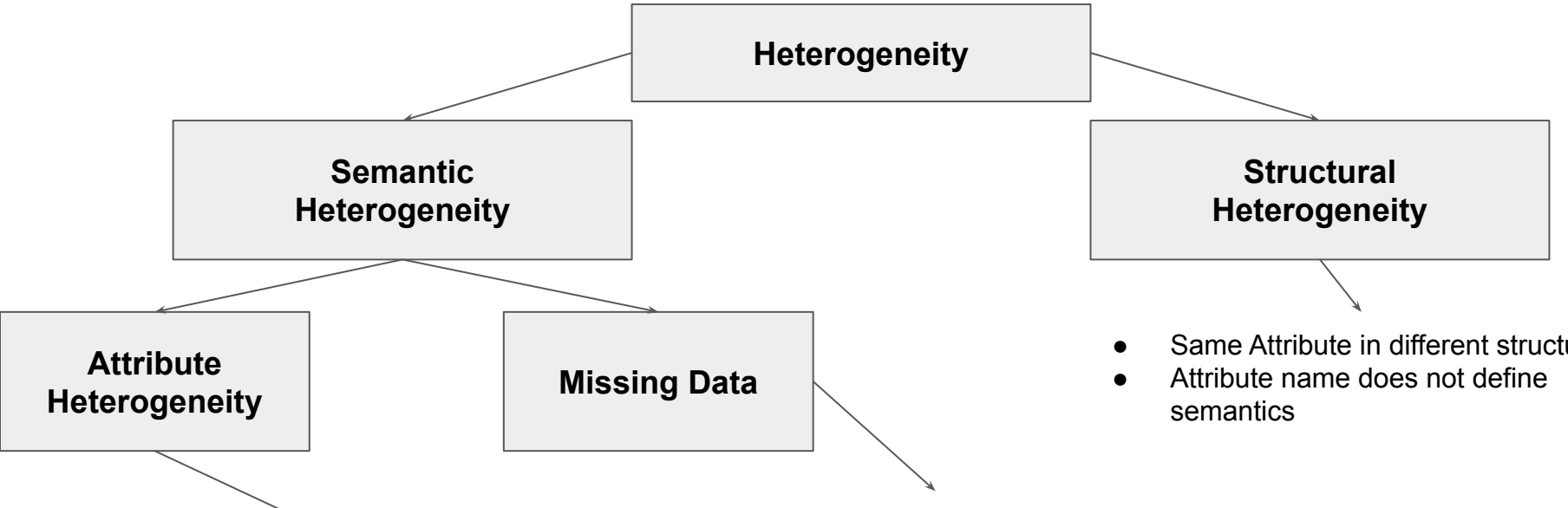


Schema Mapping vs. Schema Integration

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



Recap: Types of Heterogeneity → Scope

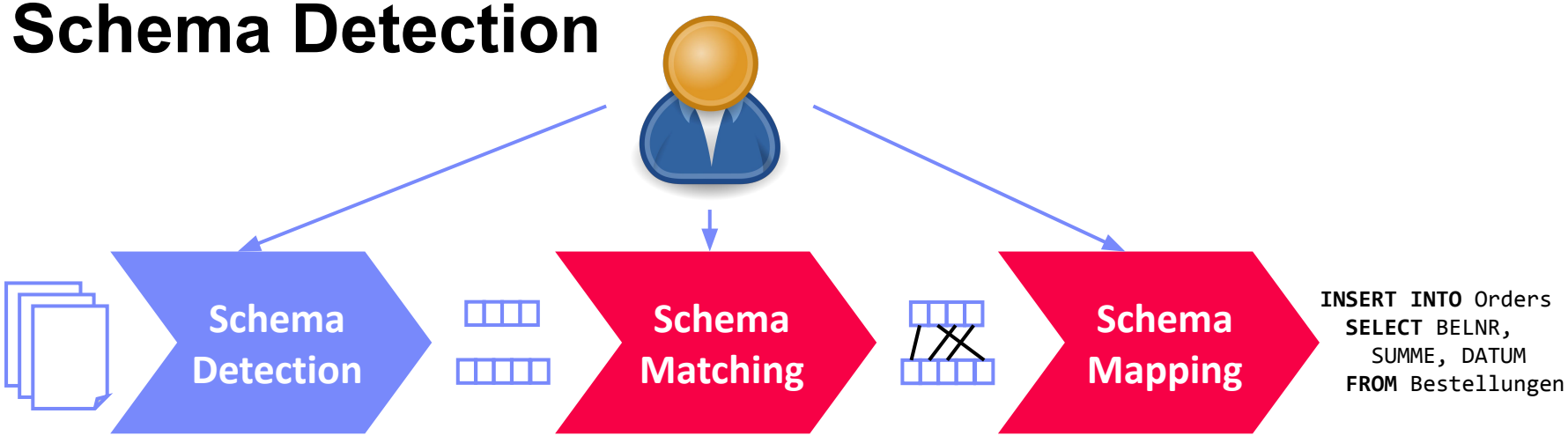


- Same Attribute in different structure
- Attribute name does not define semantics

- Nulls (Missing Values)
- Virtual Columns
- Semantic Incompatibility (Concept does not exist)

- Synonyms / Homonyms: (Instructor vs lecturer)
- Simple Mapping: (mat 12hr vs 12 hs)
- Language Expressions: (Database vs Datenbank)

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

```
import re

#Check if the string contains "i5"

txt = "DILA i5" #example data
x = re.search("i5", txt)

if x:
  print("MATCH")
  print (x)
else:
  print("NO MATCH")
```

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

./data/players.csv:

```
pid,name,pos,jnum,ncid,tid
4614,Franco Mastuantuono,FW,30,1313,258
5435,Lamine Yamal,FW,10,789,144
6909,Marcel Sabitzer,FW,20,163,308
```

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

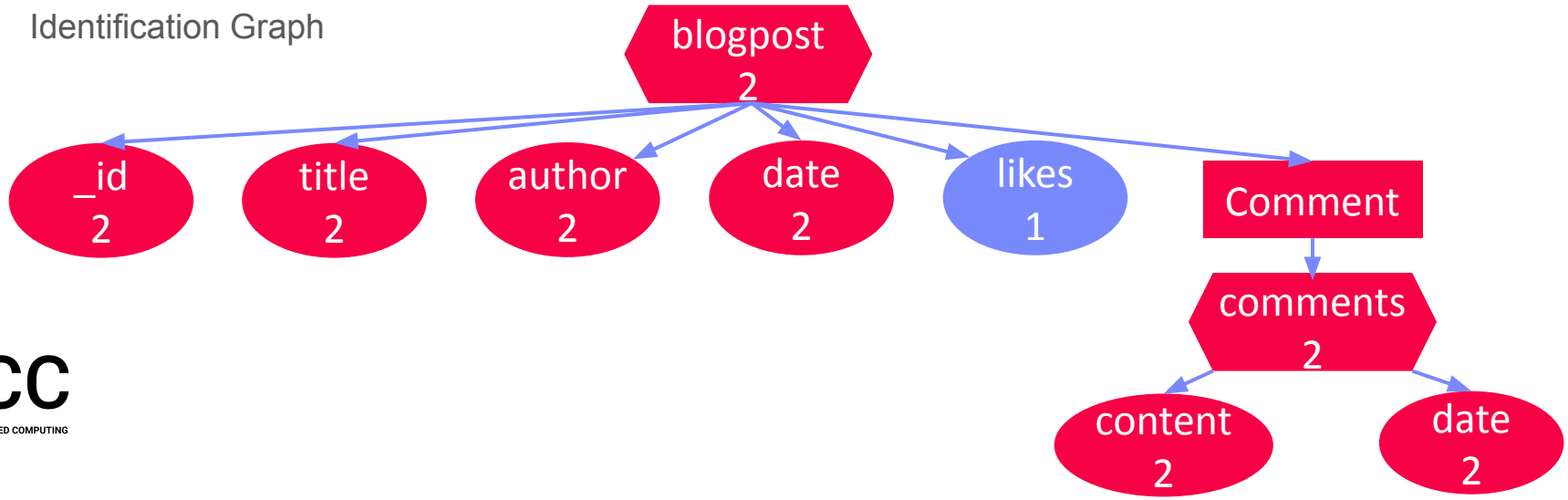
```
StructType(
  StructField(pid,IntegerType,true),
  StructField(name,StringType,true),
  StructField(pos,StringType,true),
  StructField(jnum,IntegerType,true),
  StructField(ncid,IntegerType,true),
  StructField(tid,IntegerType,true))
```

Structure Extraction from Nested Documents

- **Structure Extraction Overview**
  - **Problem:** JSON/JSONL, XML documents with optional attributes, subtrees
  - **Approach:** Scan data, build maximum tree w/ attached metadata
  - 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**]



Good to have now!

Similarity Measures: How to detect similar attributes

Metrics that compare **how similar** the structures are between different documents.

Jaccard Similarity: focuses on set overlap. It is only concerned with how many property names two documents have in common, relative to the total number of unique properties that exist between them.

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

- A and B are the sets of attributes of two documents.
- A **close to 1** means --» structures are **similar.**
- A **close to 0** indicates --» structures are **different.**

Name	Age
Luciano	14

Name	Hobby
Maria	Drawing

J (**A**,**B**) = 1 / 3 = 0,33

Good to have now!

Similarity Measures: How to detect similar attributes

Metrics that compare **how similar** the structures are between different documents.

Levenshtein Similarity: compare the similarity of the names of the attributes. It measures **how many minimal operations (**insertions, deletions, or character substitutions) are required to **convert one string into another.**

$$\text{similarity} = 1 - \frac{\text{distance}}{\max(\text{len(a)}, \text{len(b)})}$$

- Smaller distance **--»** more similar
- Larger distance **--»** less similar (more edits)
- 0 **--»** strings are identical

S (**A**,**B**) = 1 - **8** /13 = 0,385

Customer_Name
Mike Wilson

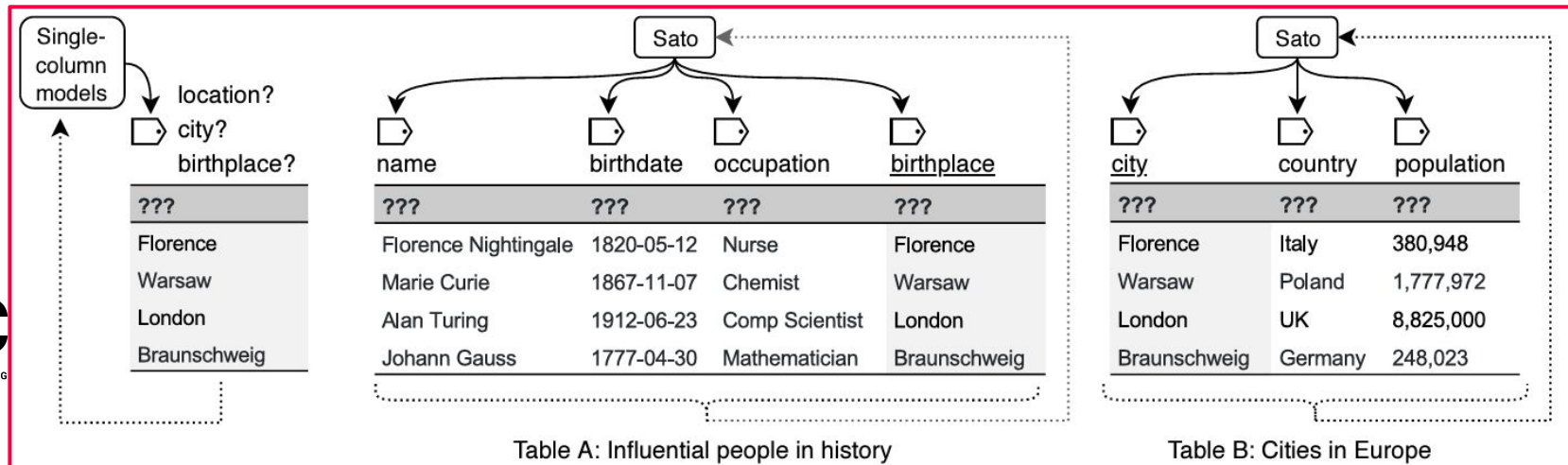
Client_Id
XXXXX

Semantic Data Type Detection

[Dan Zhang et al: Sato: Contextual Semantic Type Detection in Tables. **PVLDB 2020**]

- **Problem**
  - Detect **semantic** types of **columns** with **context**
  - **Use cases:** improved data cleaning, schema matching via semantic types

- **SATO: LDA + DNN-based Type Detection**
  - LDA model to estimate a table's intent (global context)
  - Co-occurrence of columns (local context)
  - Sherlock for single column type prediction



Schema Enforcement and Evolution

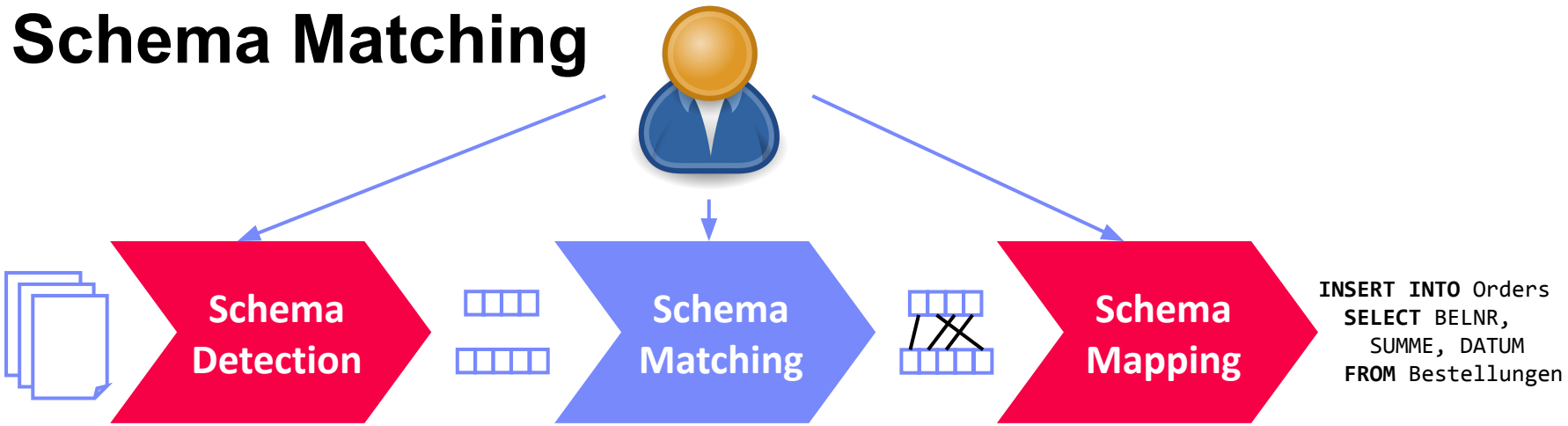
- **Schema Enforcement**
  - **Given** schema of syntactic and semantic types
  - **#1 Raise errors** on invalid data ingestion (ACID **consistency and atomicity**)
  - **#2 Leverage schema constraints** for automatic data cleaning

Schema Enforcement and Evolution

- **Schema Evolution**
  - Incremental modification of schema
  - Handle changes of source data (additional columns, different data types)
  - Copy (a new table) vs. in-place (original table) 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://www.databricks.com/blog/2019/09/24/diving-into-delta-lake-schema-enforcement-evolution.html , **2019**]

Schema Matching



Overview Schema Matching

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

- **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

S1

Books	
ISBN	CHAR(15) **PK**
Title	VARCHAR(96)
Author	VARCHAR(64)
Price	FLOAT

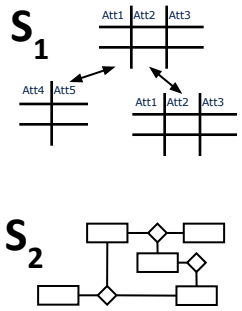
S2

BookInfo	
ID	CHAR(15) **PK**
AuthorID	INT REF AuthorInfo
BookTitle	VARCHAR(128)
ListPrice	FLOAT
DiscPrice	FLOAT

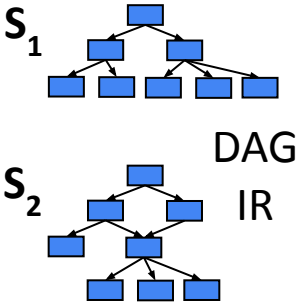
AuthorInfo	
AuthorID	**INT PK**
LastName	VARCHAR(32)
FirstName	VARCHAR(32)

System Architecture and Matching Process

External Schemas

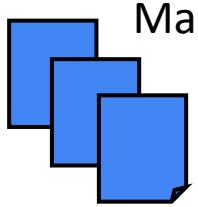


Model Pool



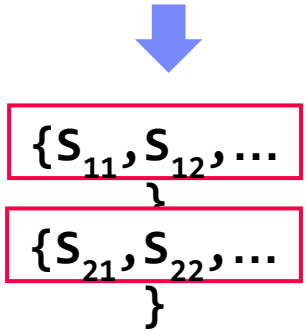
[Erhard Rahm, Hong-Hai Do David Aumueller, Sabine Massmann: Matching Large Schemas with COMA++, **2005**]

Mapping Pool

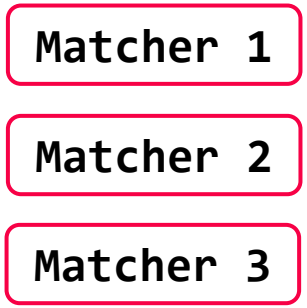


Mapping Manipulation

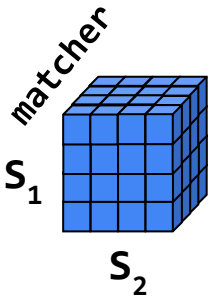
Matcher Configs



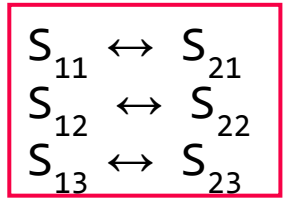
Component Identification



Matcher Execution



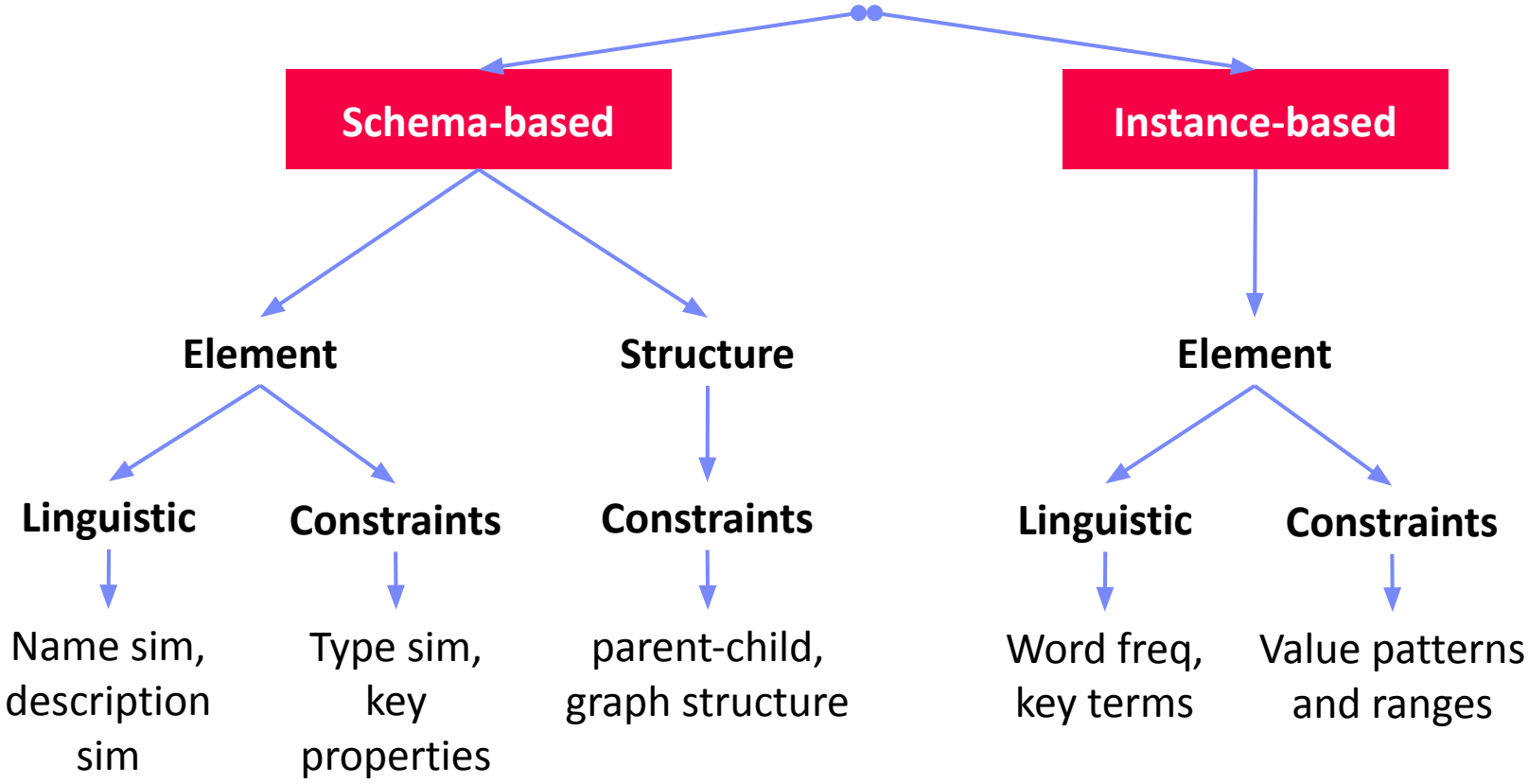
Similarity Combination



Correspon-dences

HCC Institute of HUMAN-CENTRED COMPUTING

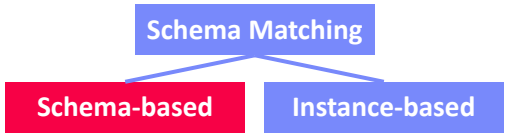
Classification of Matching Techniques



- **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

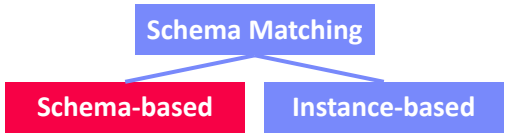


- **Linguistic Approaches**
  - Syntactic
    - Affix (suffix, prefix)
  - Semantic
    - Synonyms
    - Hierarchy → taxonomies
    - Language → dictionaries

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

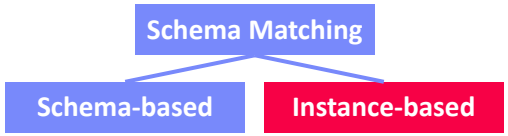
Firstname	Lastname	Street	Num	ZIP	City
Susanne	Froehlich	Weststrasse	2	01187	Dresden
Thomas	Kunze	Dammweg	45	12437	Berlin

Selected Matchers



- **Example Trigram**
  - Similarity = 2|S1 ∩ S2| / (|S1|+|S2|) **[Jaccard]**
  - String 1 → "Name" → " Name " (we add **2 spaces before** and **1 space after** ")
  - String 2 → "Lastname" → " Lastname "
  - " N", " Na", "Nam", "ame", "me " (S1 = # trigrams = 5) **Tokens!**
  - " L", " La", "Las", "ast", "stn", "tna", "nam", "ame", "me " (S2 = # trigrams = 9)
  - Common trigrams = {"nam", "ame", "me "} |S1 ∩ S2| = 3
  - Similarity = 2×3 / 5+9 = 6 / 14 =0.4285

Selected Matchers, cont.



- **Problem Schema-based**
  - Attribute names might differ vastly (CustNa vs Name, CustSt vs Street)

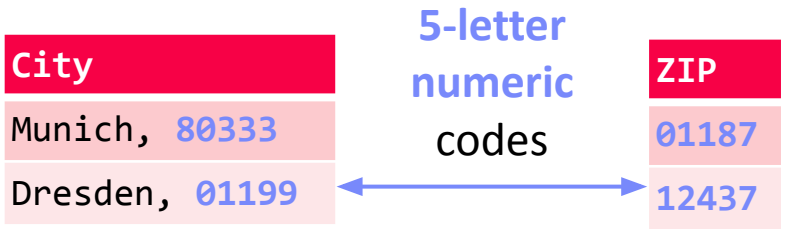
  → **Instance-based matching**, but need for data

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

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

- **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

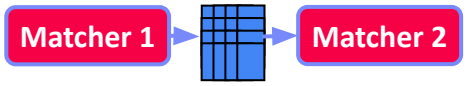


Combination of Matchers

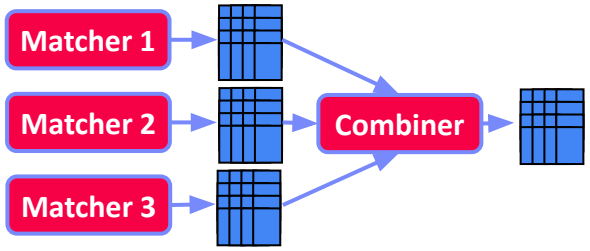
- **#1 Reuse**
  - Exploitation of **transitive similarity**

$S1 \cong S3 \wedge S3 \cong S2 \rightarrow S1 \cong S2$

- **#2 Refinement**
  - **Chaining:** matcher M_{n+1} works on mappings of M_n --» efficiency



- **#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**
  - **Reduced schema sizes** for **expensive matchers**
- **#2 Parallel Matching**
  - Process different steps/fragments **in parallel**

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

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**

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

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

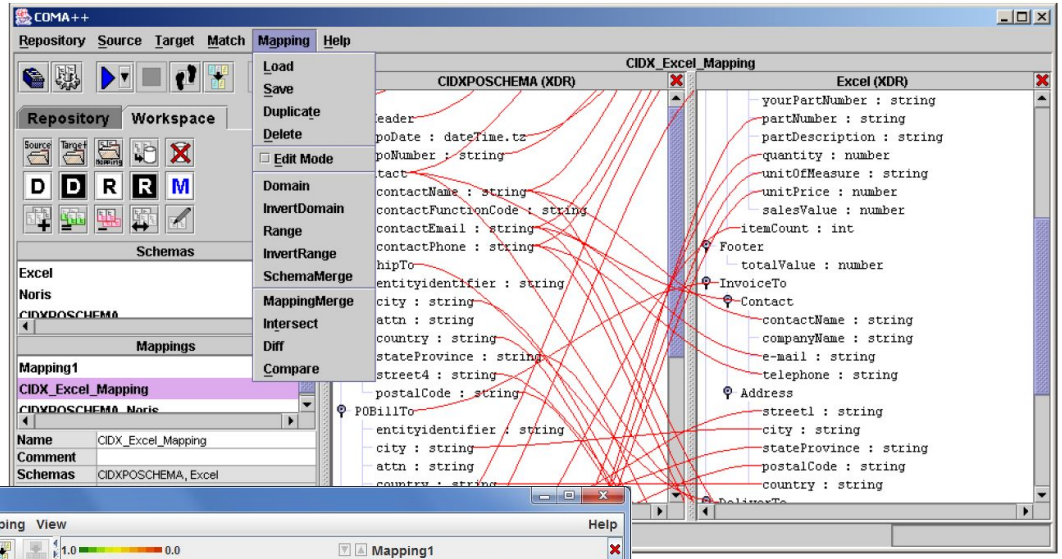
Schema Matching Tools, cont.

COMA system for **co**mbining **ma**tch algorithms in a flexible way)

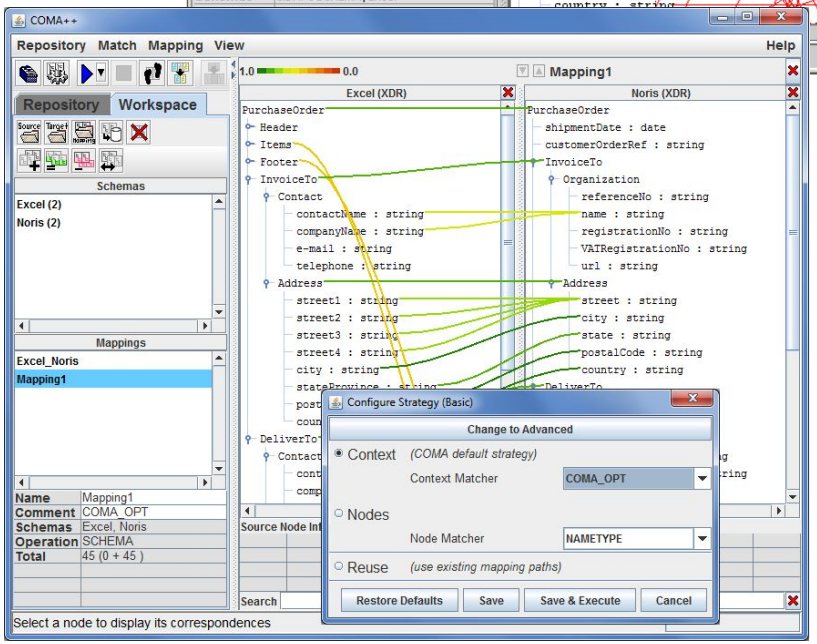
- **COMA ++**

[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**]



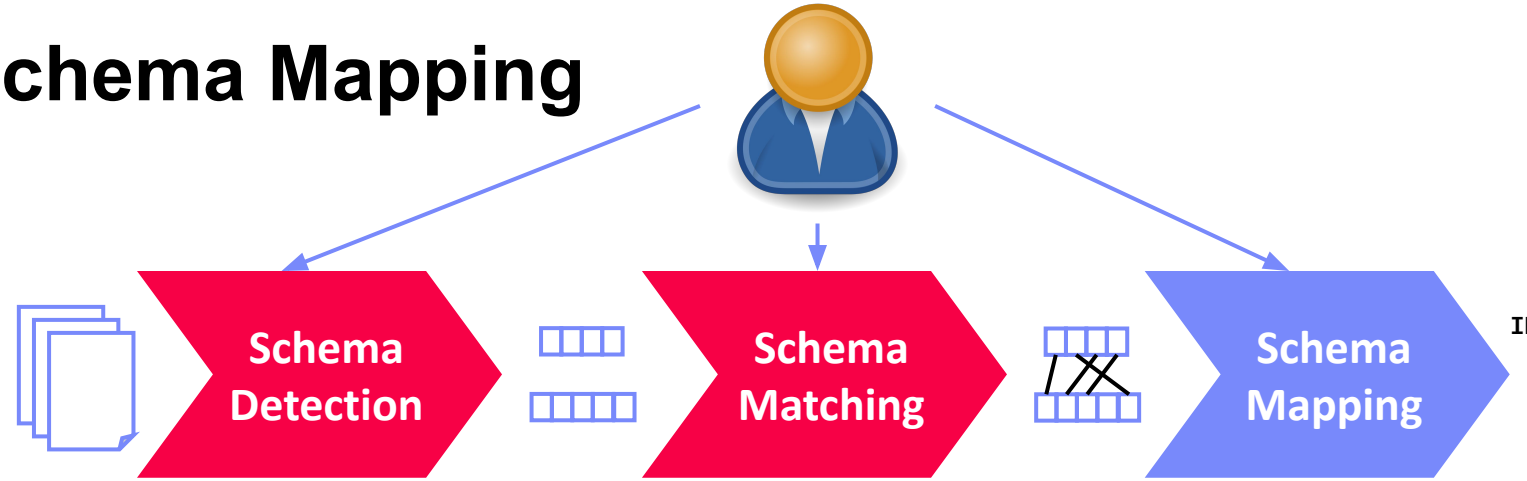
- **COMA 3.0**
  - 2011/2012
  - **Ontology Merging**
  - Workflow Management



[**Credit:** https://dbs.uni-leipzig.de/de/Research/coma.html]

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

Schema Mapping

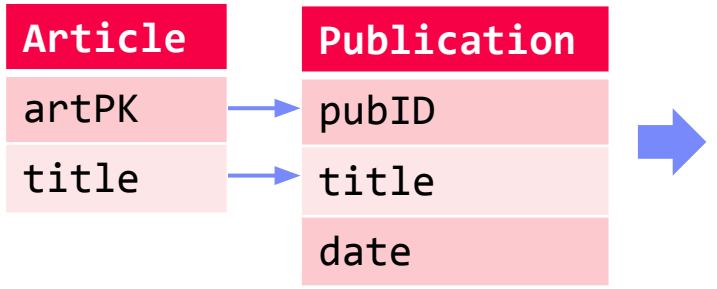



```
INSERT INTO Orders
SELECT BELNR,
  SUMME, DATUM
FROM Bestellungen
```

HCC Institute of HUMAN-CENTRED COMPUTING

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.
  - **Transformation query:** transformation program from one into the other schema, backend-specific (query generation)
- **Example**

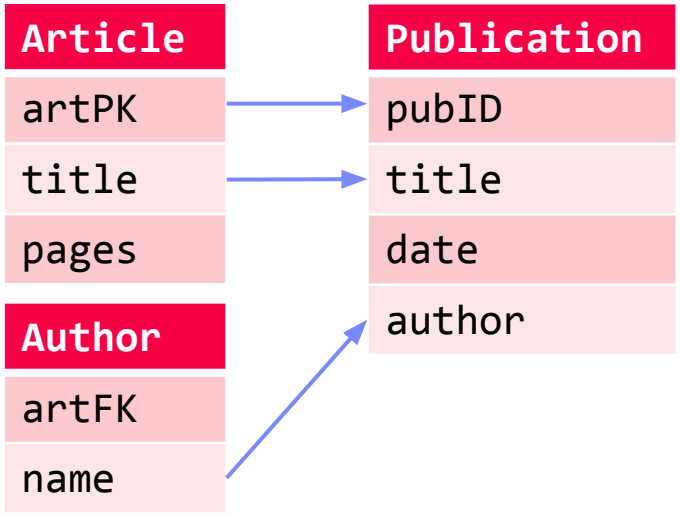


Transformation Query:

```
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

- **Without Intra-Schema Correspondences**

Article

artPK	title
1	AI and Society
2	Data Spaces

Author

artFK	name
1	Lucas Iacono
2	Matthias Müller

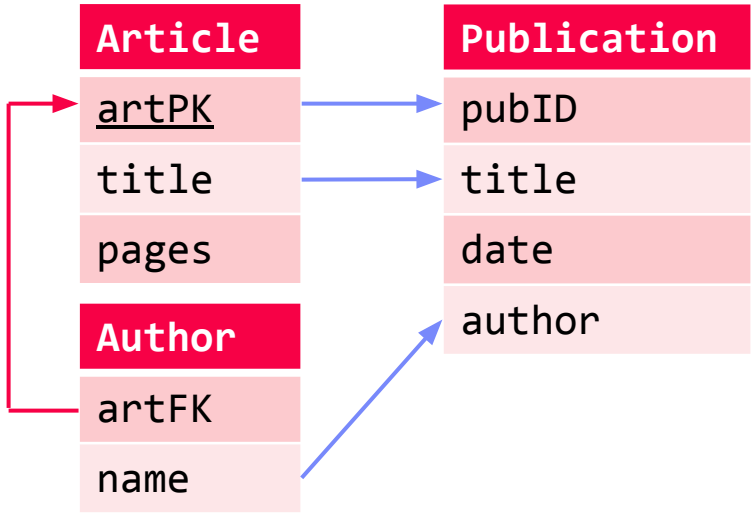
```
INSERT INTO Publication
  (SELECT artPK AS pubID,
          title AS title,
          NULL AS date,
          NULL AS author
     FROM Article)
  UNION ALL
  (SELECT NULL AS pubID,
          NULL AS title,
          NULL AS date,
          name AS author
     FROM Author)
```

Publication

pubID	title	date	author
1	AI and Society	**NULL**	**NULL**
2	Data Spaces	**NULL**	**NULL**
NULL	**NULL**	**NULL**	Lucas Iacono
NULL	**NULL**	**NULL**	Matthias Müller

Mapping Interpretations, cont.

- With Intra-Schema Correspondences




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

Mapping Interpretations

- **Without Intra-Schema Correspondences**

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

Article

artPK	title
1	AI and Society
2	Data Spaces

Author

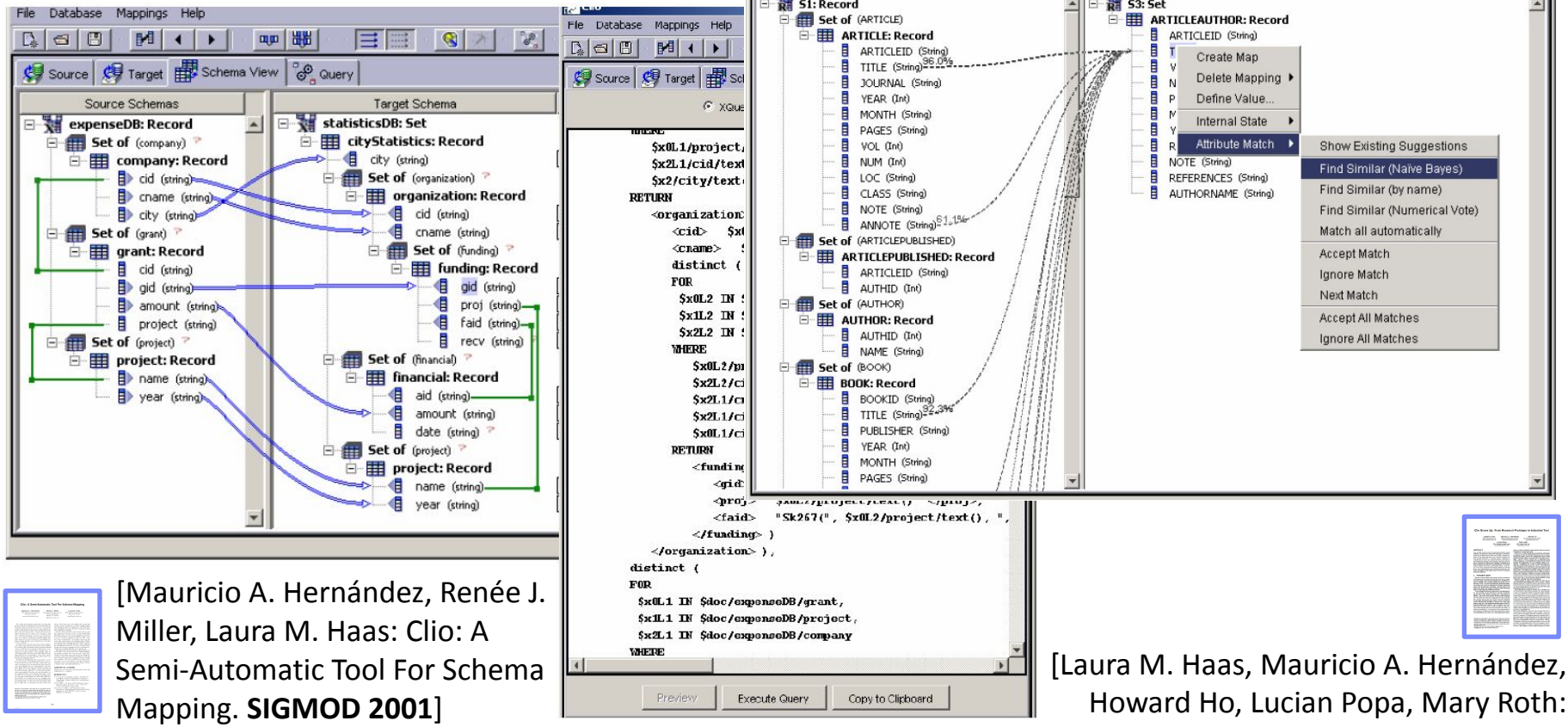
artFK	name
1	Lucas Iacono
2	Matthias Müller

Publication

artPK	title	NULL	name
1	AI and Society	**NULL**	Lucas Iacono
2	Data Spaces	**NULL**	Matthias Müller

Schema Matching Tools

- Clio (IBM Research – Almaden)



[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

- Next Lectures
  - 05 Entity Linking and Deduplication [Oct 31]
  - 06 Data Cleaning and Data Fusion [Nov 14]