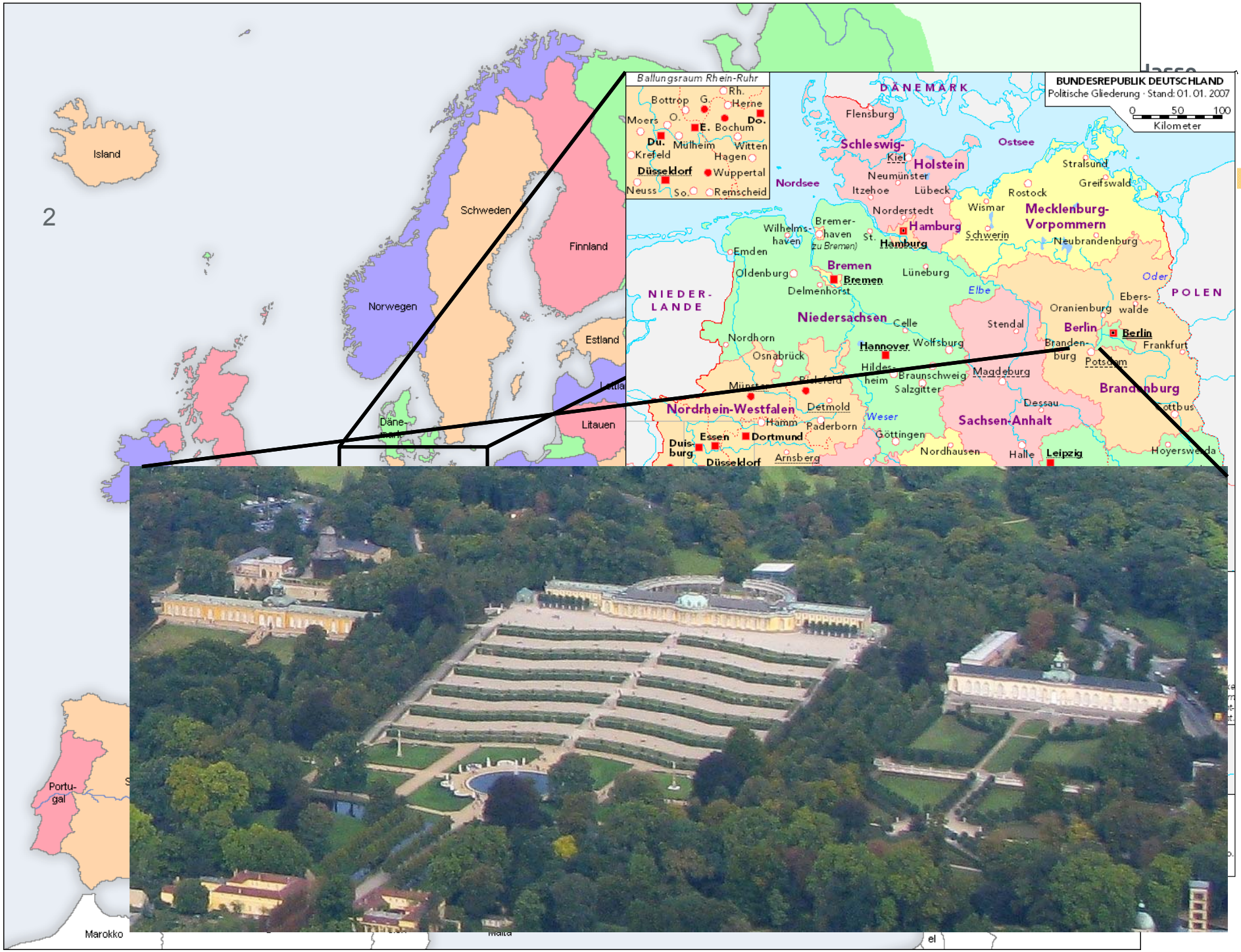
Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Data Integration
in Two Steps and From Above

Palo Alto, February 3, 2009

Felix Naumann


2
Island
Schweden
Finnland
Norwegen
Estland
Litauen
Dänemark
Portugal
Marokko
Ballungsraum Rhein-Ruhr
Bottrop
Rh. Herne
Moers
E. Bochum
Do.
Du. Mülheim
Witten
Krefeld
Hagen
Düsseldorf
Wuppertal
Neuss
So.
Remscheid
DÄNEMARK
BUNDESREPUBLIK DEUTSCHLAND
Politische Gliederung · Stand: 01. 01. 2007
0 50 100
Kilometer
Flensburg
Schleswig-Holstein
Kiel
Ostsee
Nordsee
Neumünster
Itzehoe
Lübeck
Stralsund
Greifswald
Rostock
Wismar
Mecklenburg-Vorpommern
Norderstedt
Hamburg
Schwerin
Neubrandenburg
Wilhelms-haven
Bremer-haven
(zu Bremen)
Emden
Oldenburg
Bremen
Lüneburg
Delmenhorst
Elbe
Oder
POLEN
NIEDER-LANDE
Niedersachsen
Celle
Stendal
Oranienburg
Ebers-walde
Berlin
Frankfurt
Nordhorn
Osnabrück
Hannover
Wolfsburg
Branden-burg
Potsdam
Hildes-heim
Braunschweig
Magdeburg
Salzgitter
Brandenburg
Münster
Bielefeld
Nordrhein-Westfalen
Detmold
Weser
Dessau
Cottbus
Hamm
Paderborn
Sachsen-Anhalt
Essen
Dortmund
Göttingen
Duis-burg
Düsseldorf
Arnsberg
Nordhausen
Halle
Leipzig
Hoyerswerda

The HPI – Hasso Plattner Institut

- Founded in 1998 as a Public Private Partnership
- Hasso Plattner, co-founder of SAP, endowed over € 200 Mio.
- Adjoined with the University of Potsdam
  - Capital of Brandenburg, bordering Berlin
- 450 students – Bachelor, Master, and PhD

Information systems team

Paul **Führing**

project **ViQTOR**

Pat **Hobro**

Prof. Felix **Naumann**

Jens **Bleiholder**

project **fusem**

Karsten **Draba**

Data Fusion

project **HumMer**

DQ Assessment

Information Integration

Information Quality

Christoph Böhm

Data Profiling & Cleaning

Armin **Roth**

project **System P**

Peer Data Management Systems

Matching

Service-Oriented Systems

Ontologies

Data Integration for Life Science Data Sources

project **Aladin**

Mohammed **AbuJarour**

Frank **Kaufer**

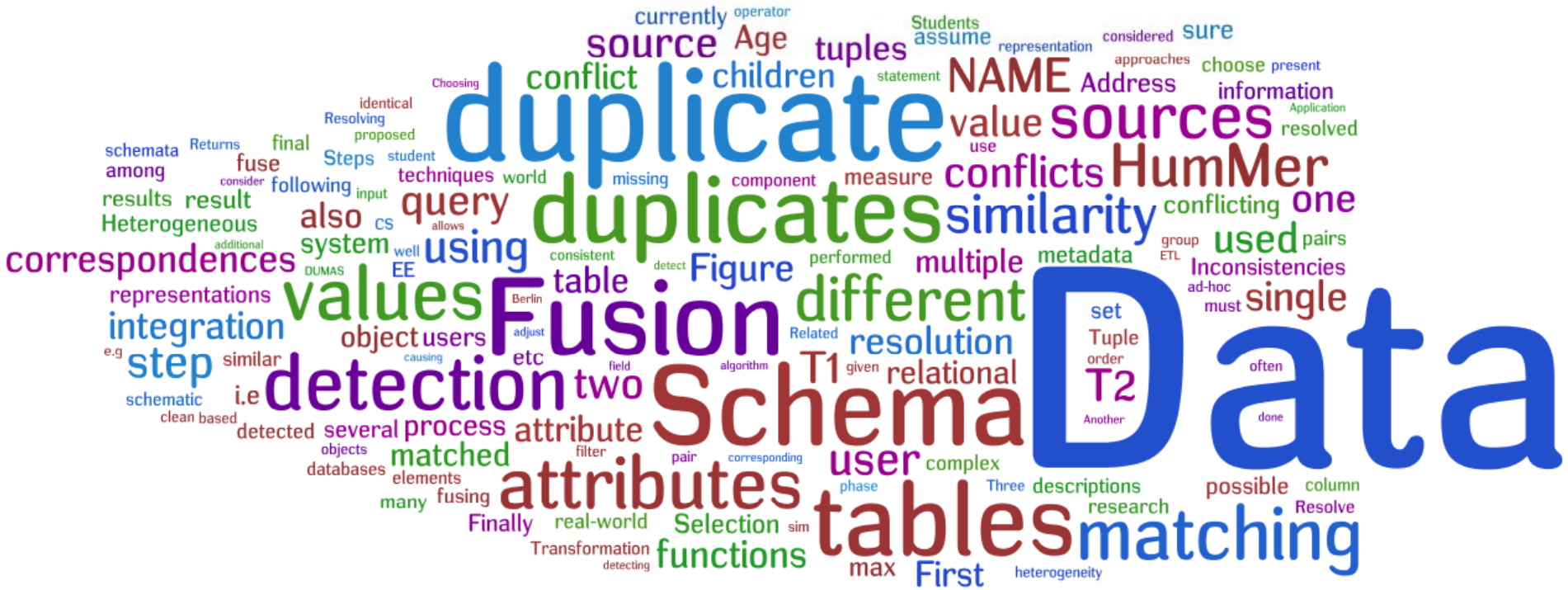
Jana **Bauckmann**

Alexander **Albrecht**

Personal Information Management

Data Profiling for Schema Management

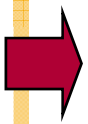
“Data Fusion in Three Steps”
DE Bulletin, 2006



Wordle.net

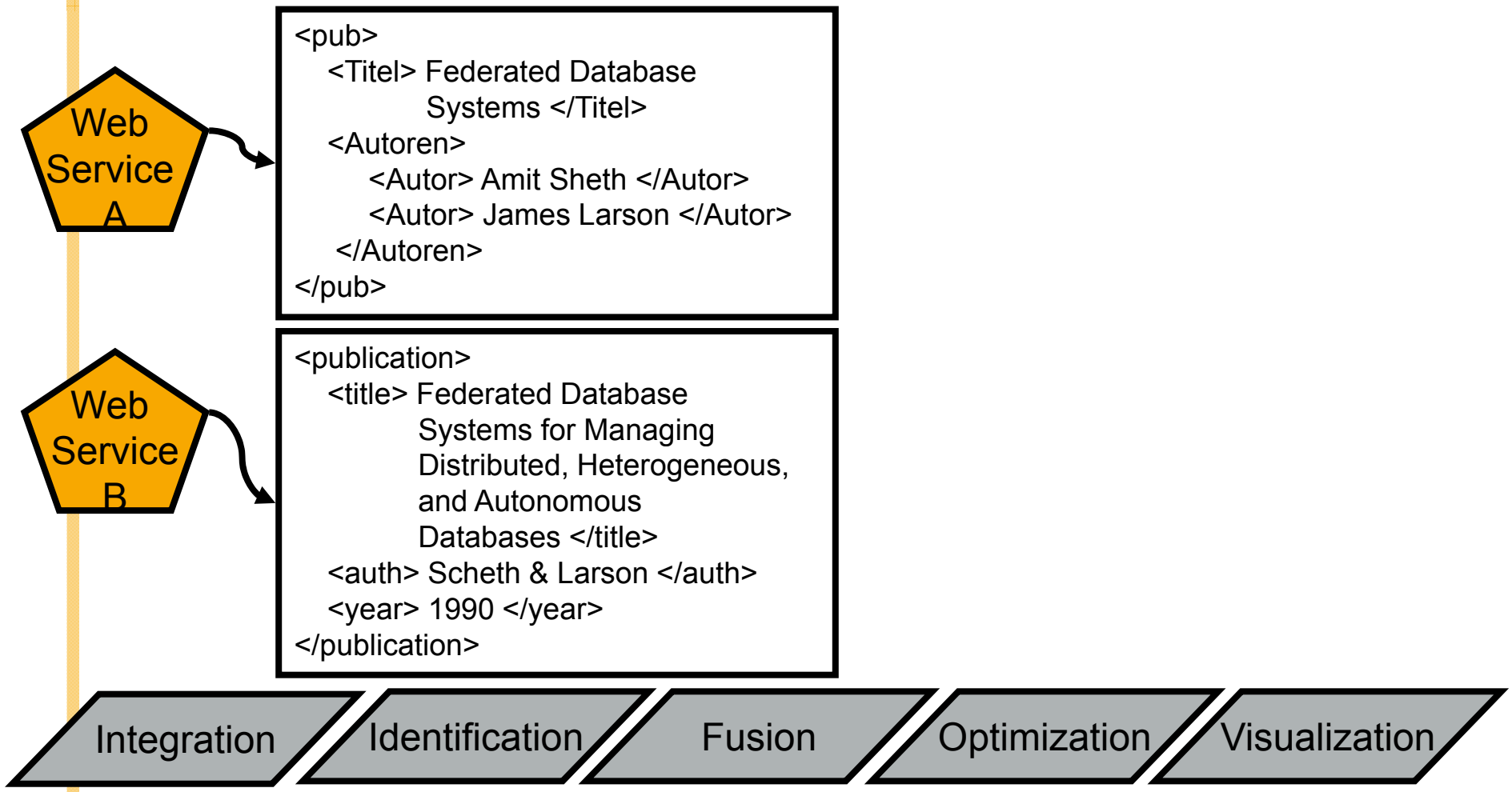
“Data Fusion”
ACM Computing Survey, 2008

Overview

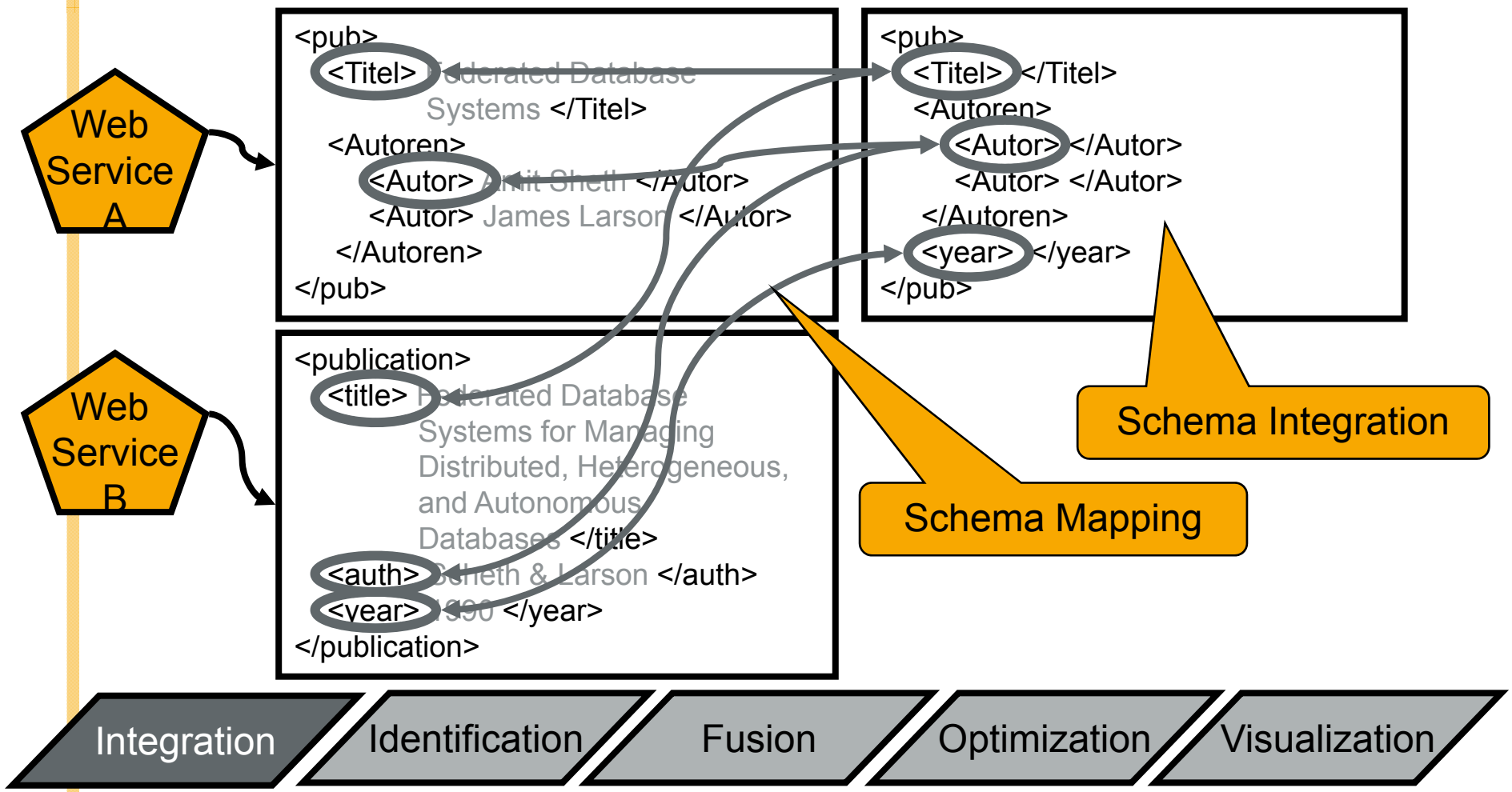


- Introductory example
- Step 1: Schema level integration
  - Schema Mapping
  - Schema Matching
- Step 2: Data level integration
  - Duplicate detection
  - Data fusion
- From above: ETL management

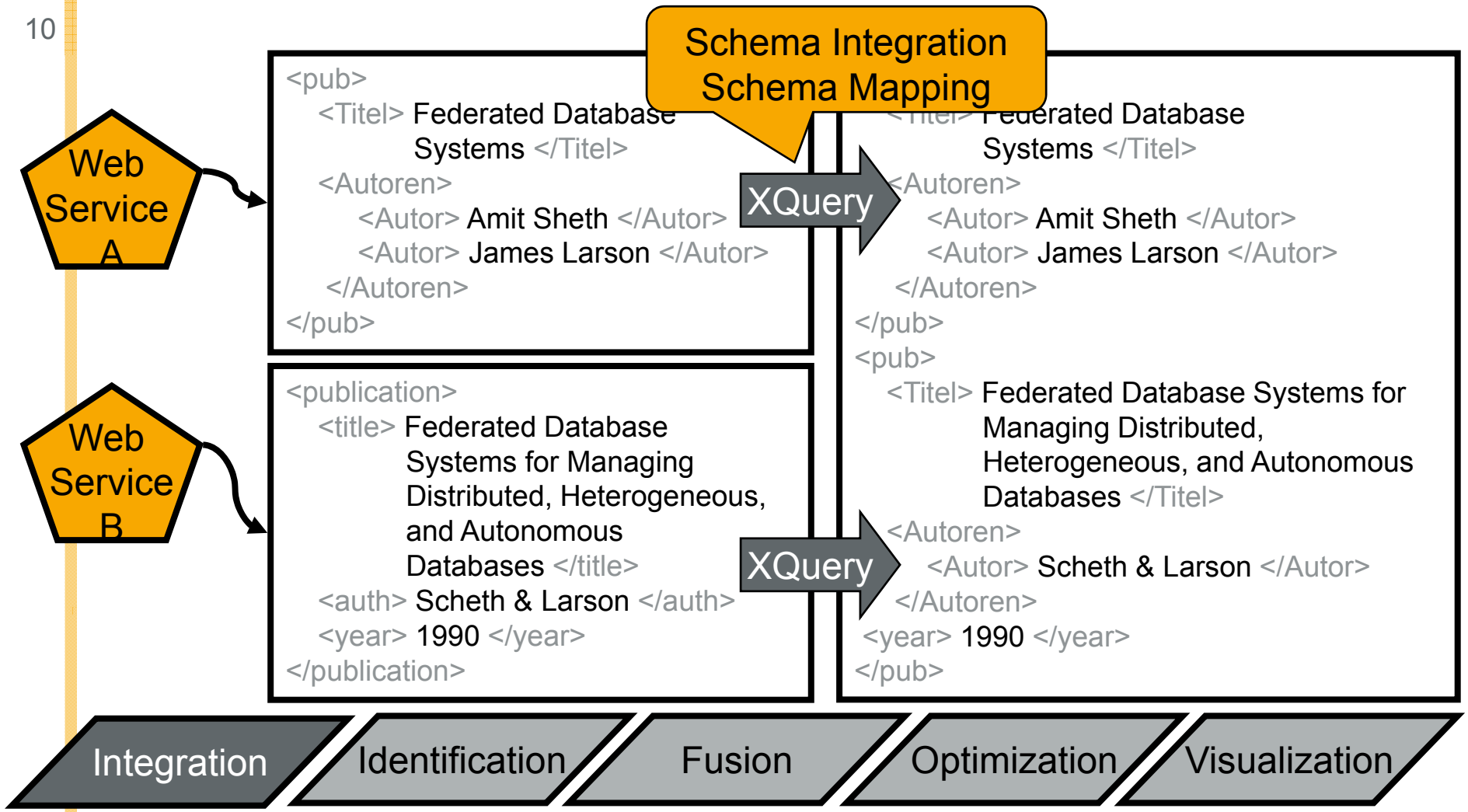
Information Integration



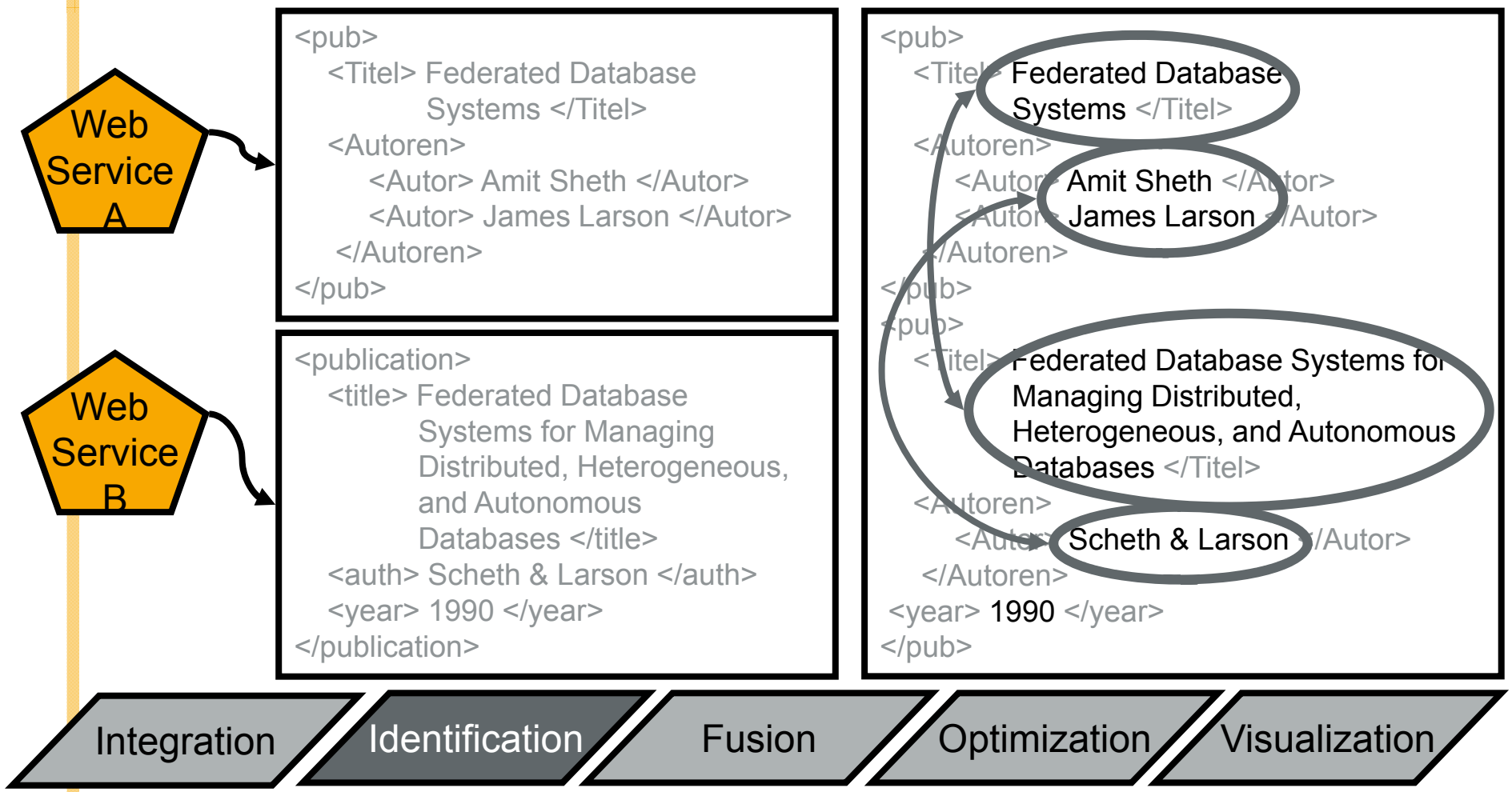
Information Integration



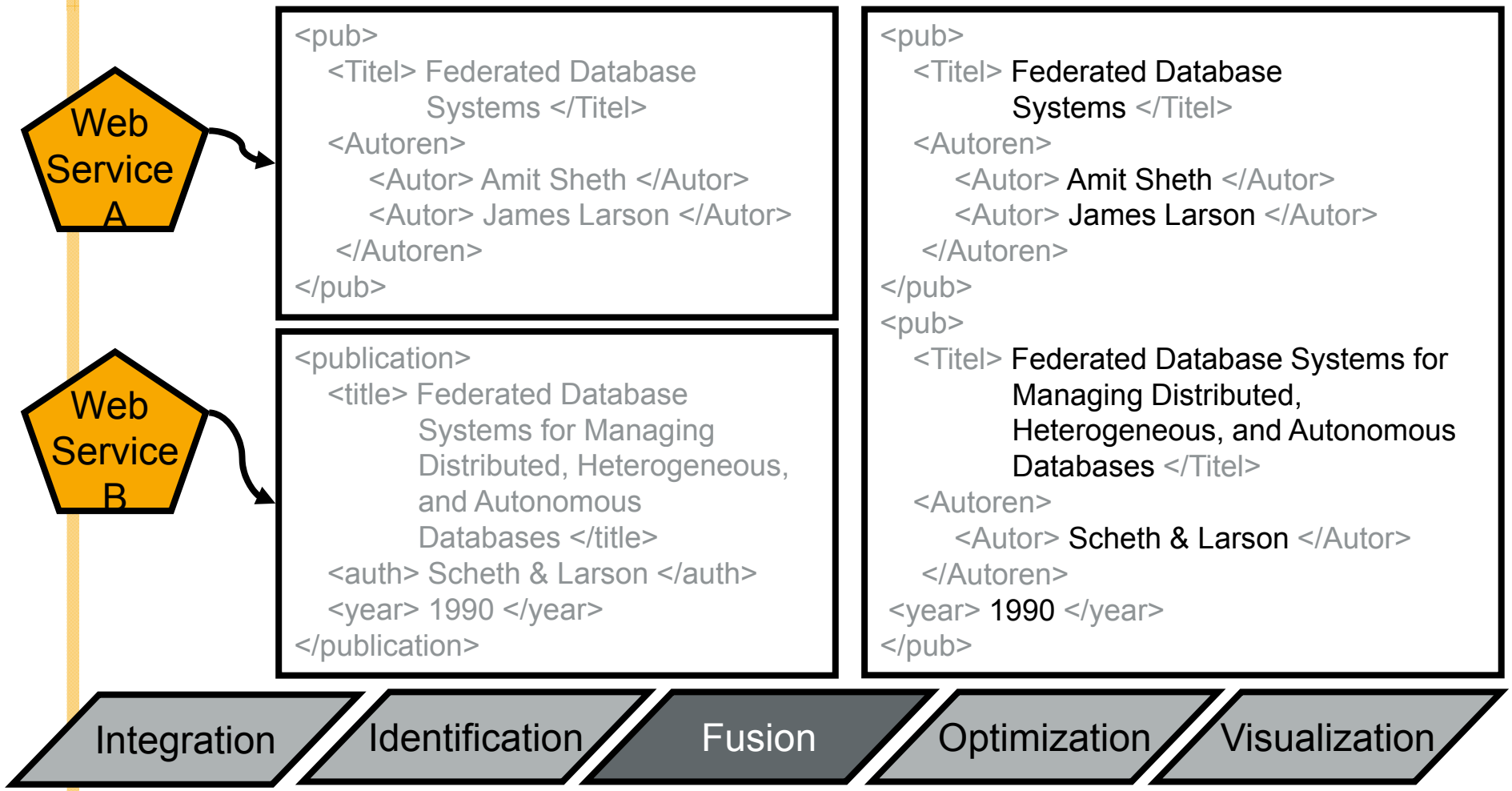
Information Integration



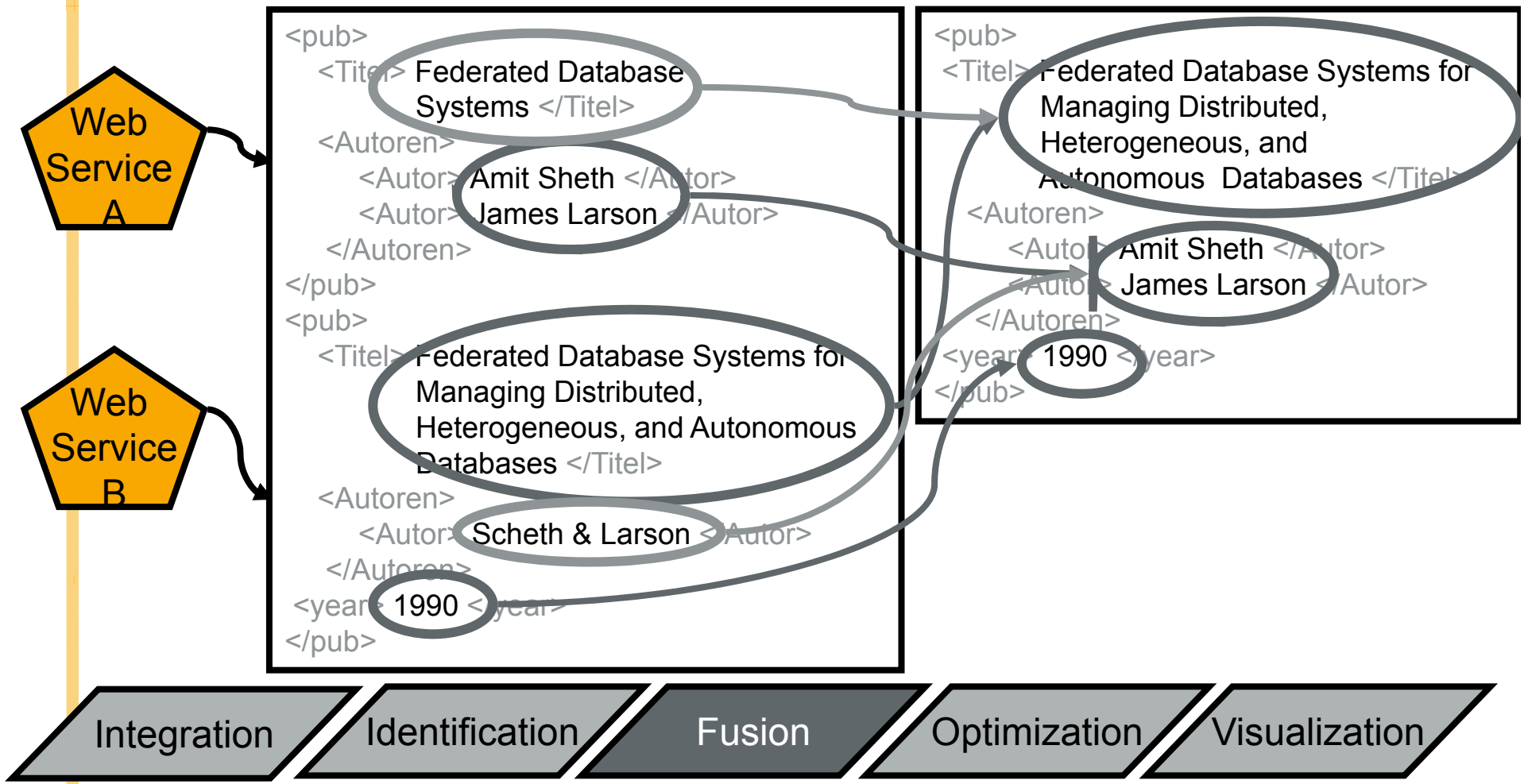
Information Integration



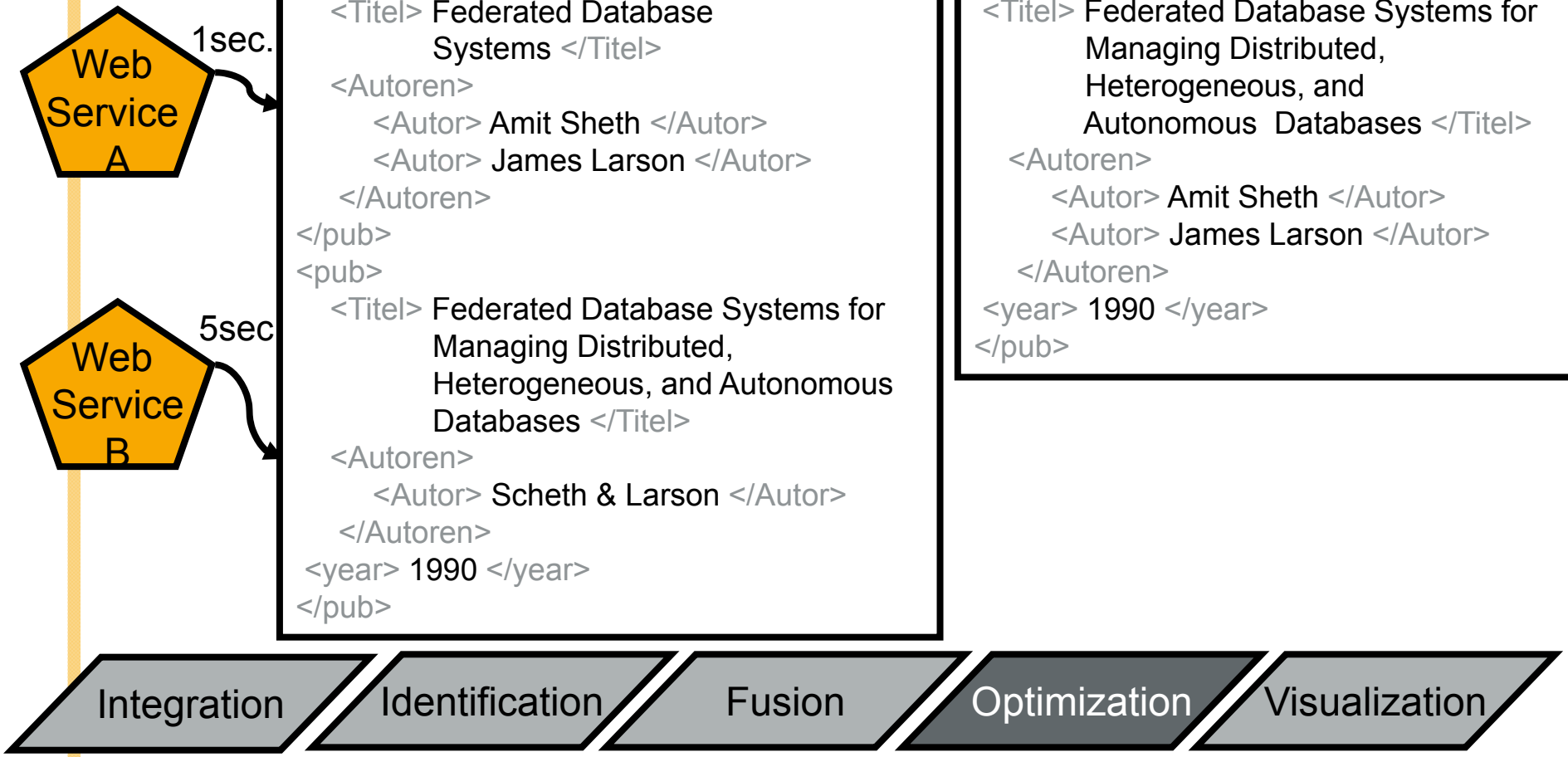
Information Integration



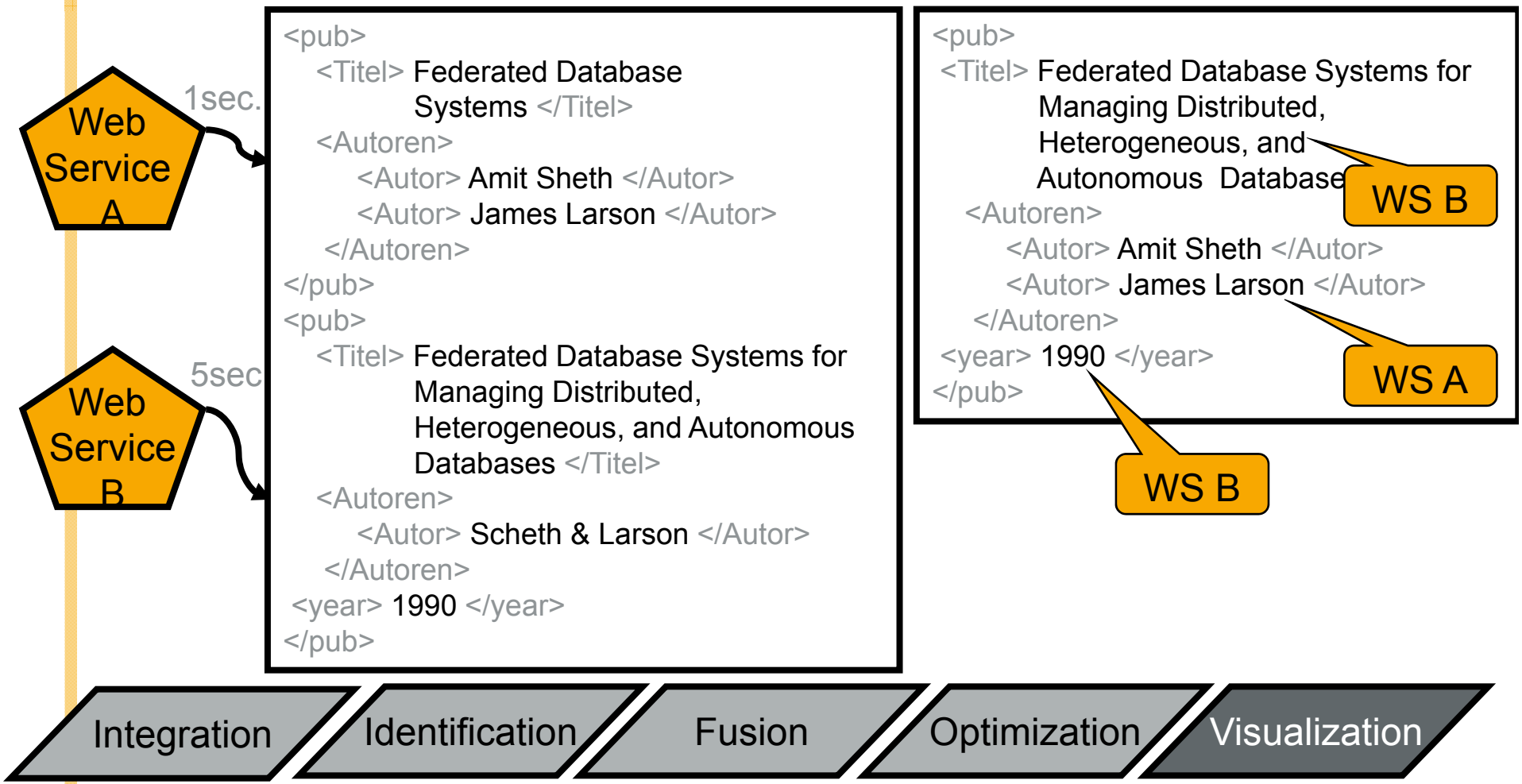
Information Integration



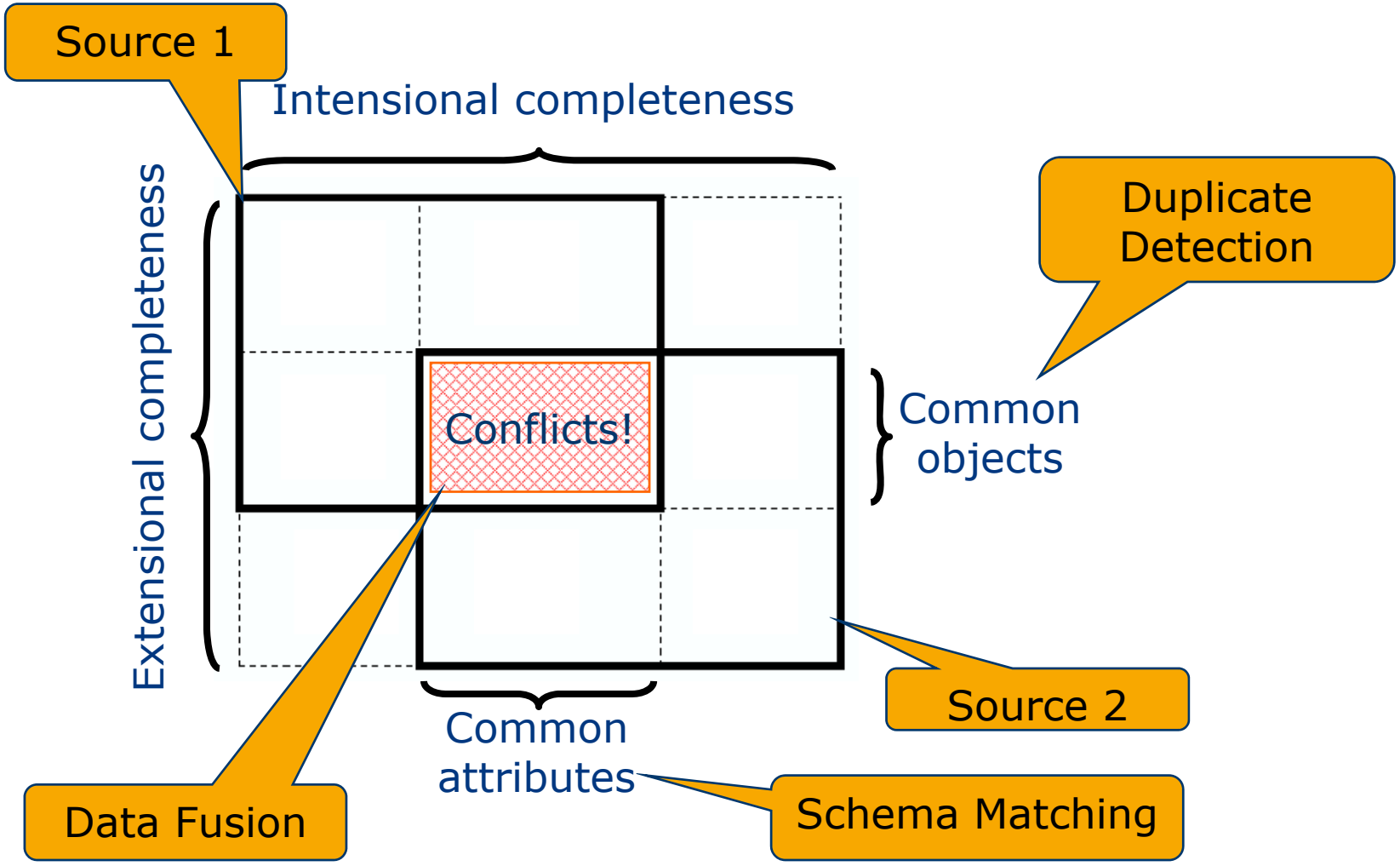
Information Integration



Information Integration



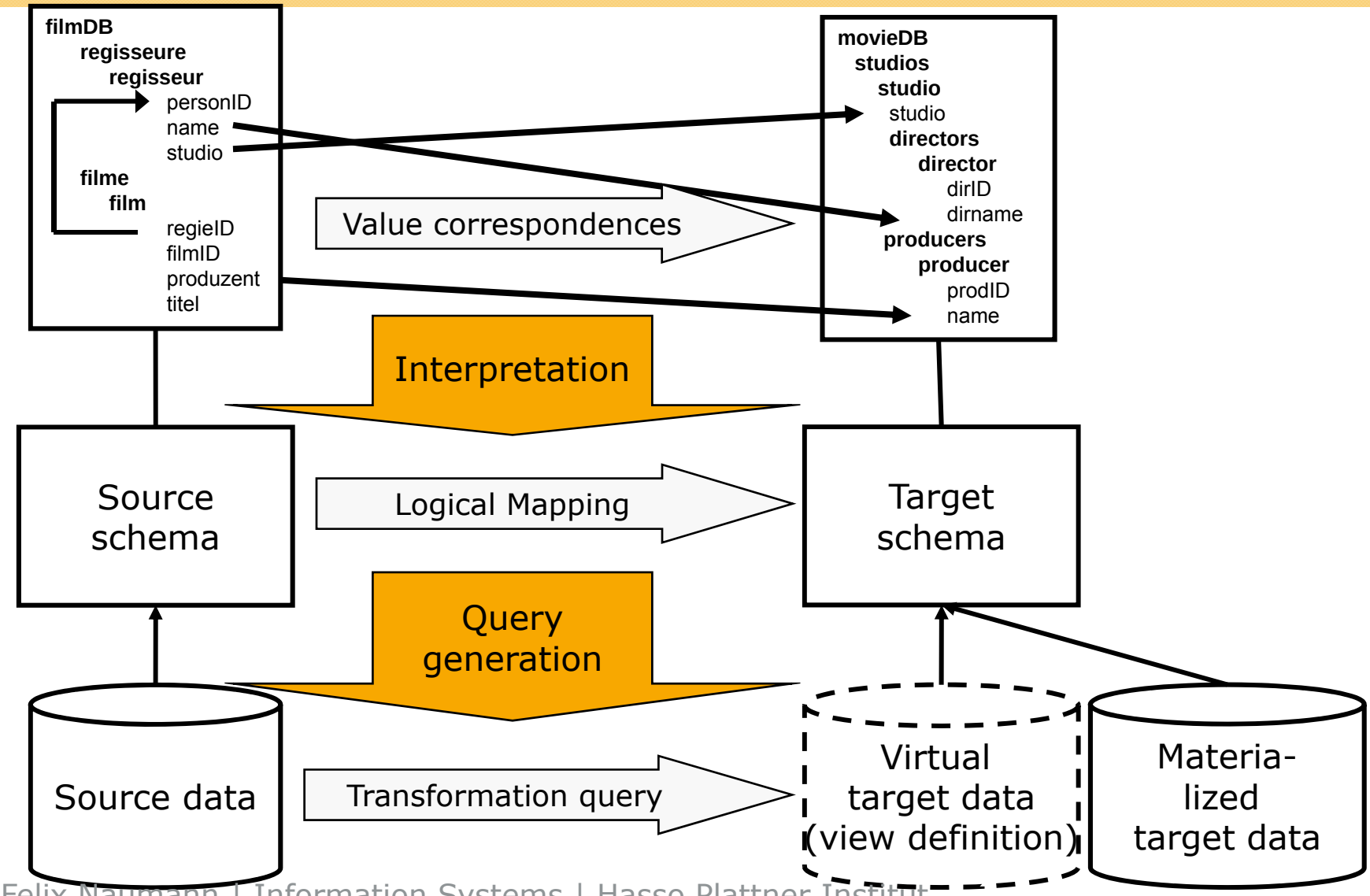
Completeness and Conciseness



Overview

- Introductory example
- Schema level integration
  - Schema Mapping
  - Schema Matching
- Data level integration
  - Duplicate detection
  - Data fusion
- ETL management

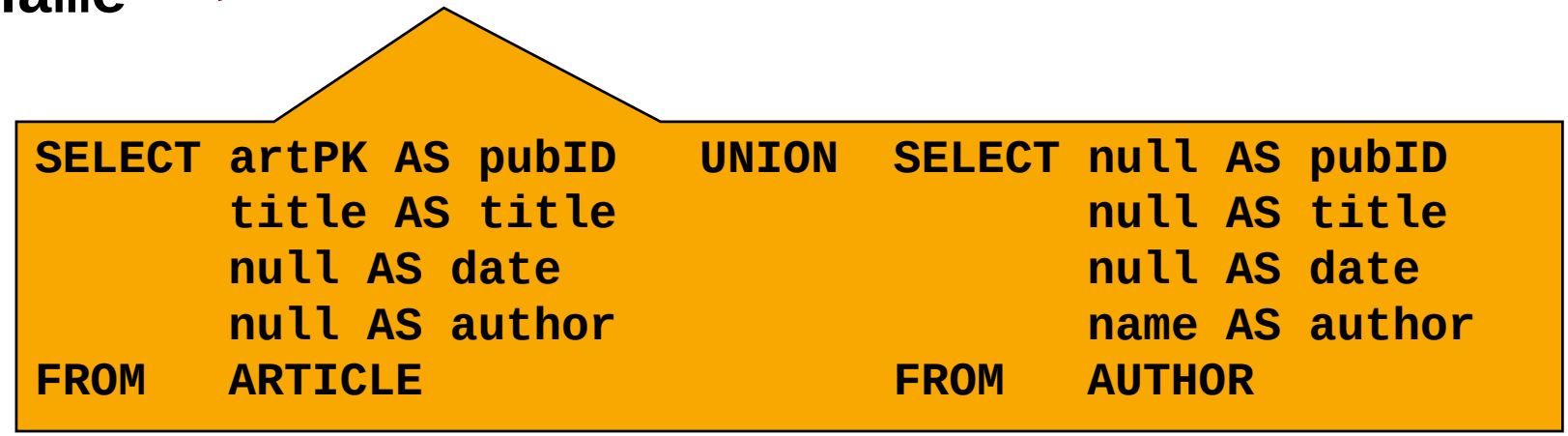
Schema Mapping



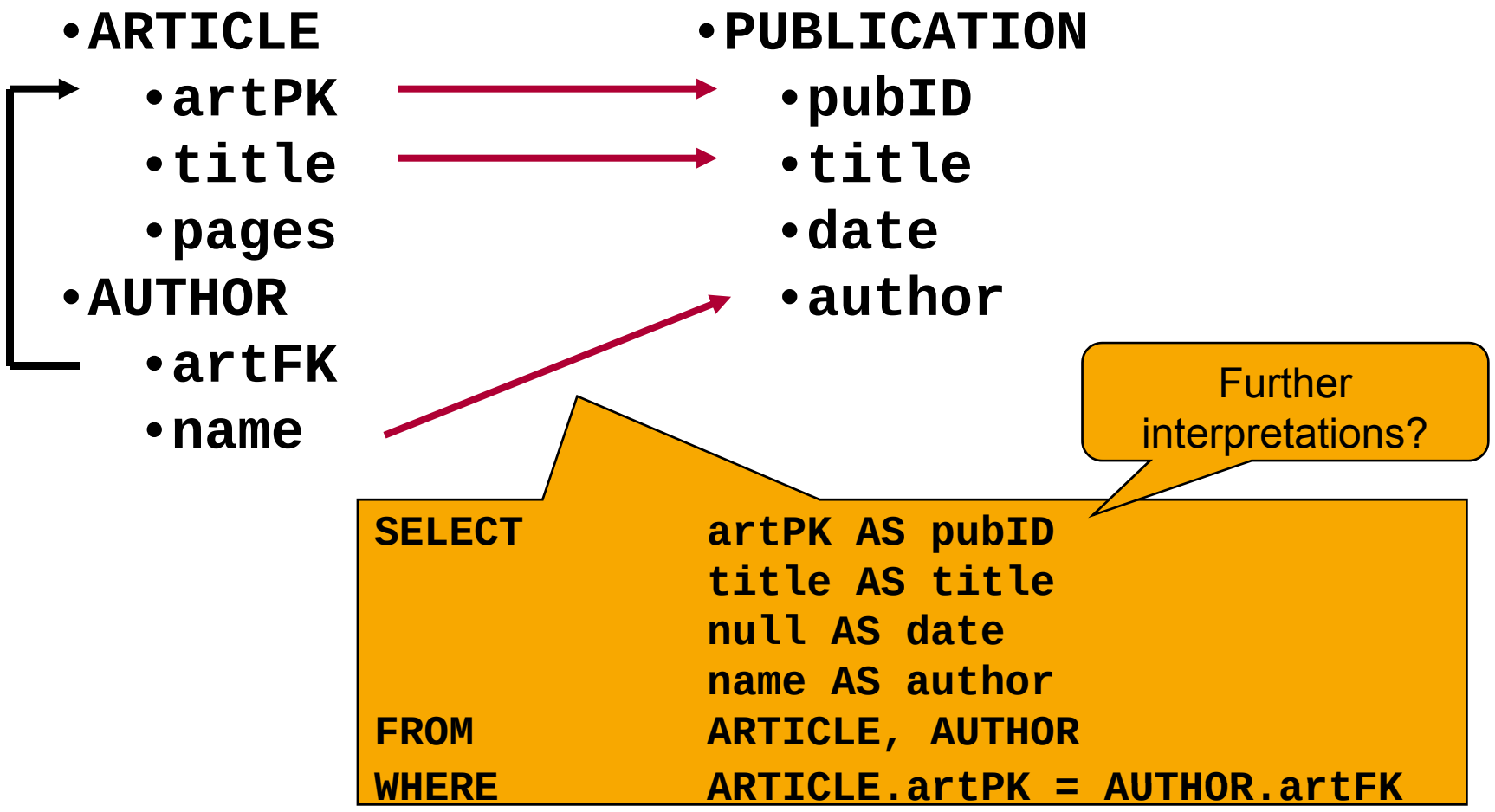
Schema Mapping Example

- **ARTICLE**
  - artPK
  - title
  - pages
- **AUTHOR**
  - artFK
  - name

- **PUBLICATION**
  - pubID
  - title
  - date
  - author



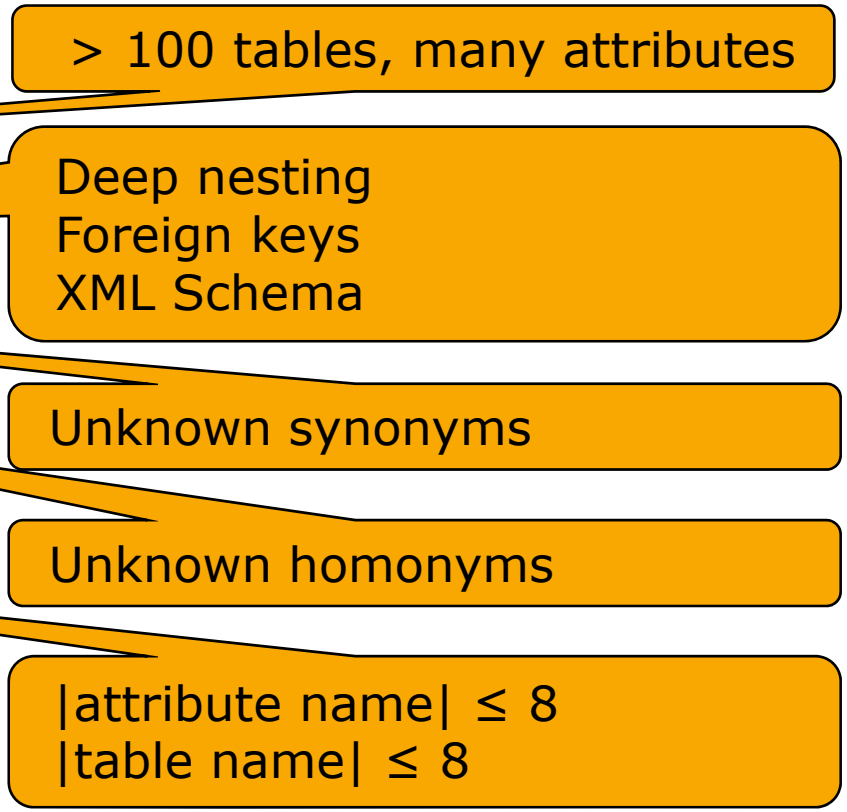
Schematic heterogeneity – solutions

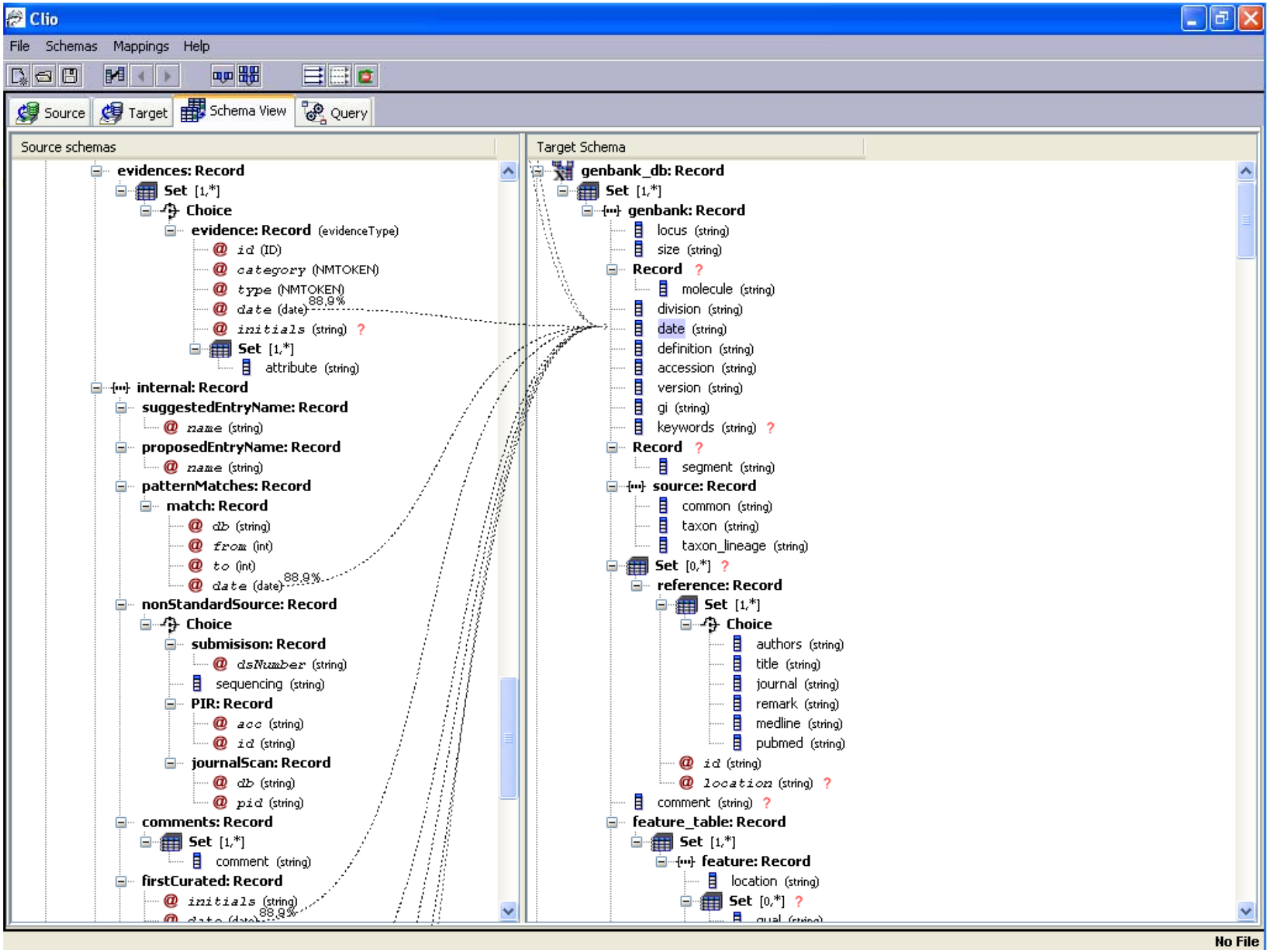


Schema Matching – Motivation

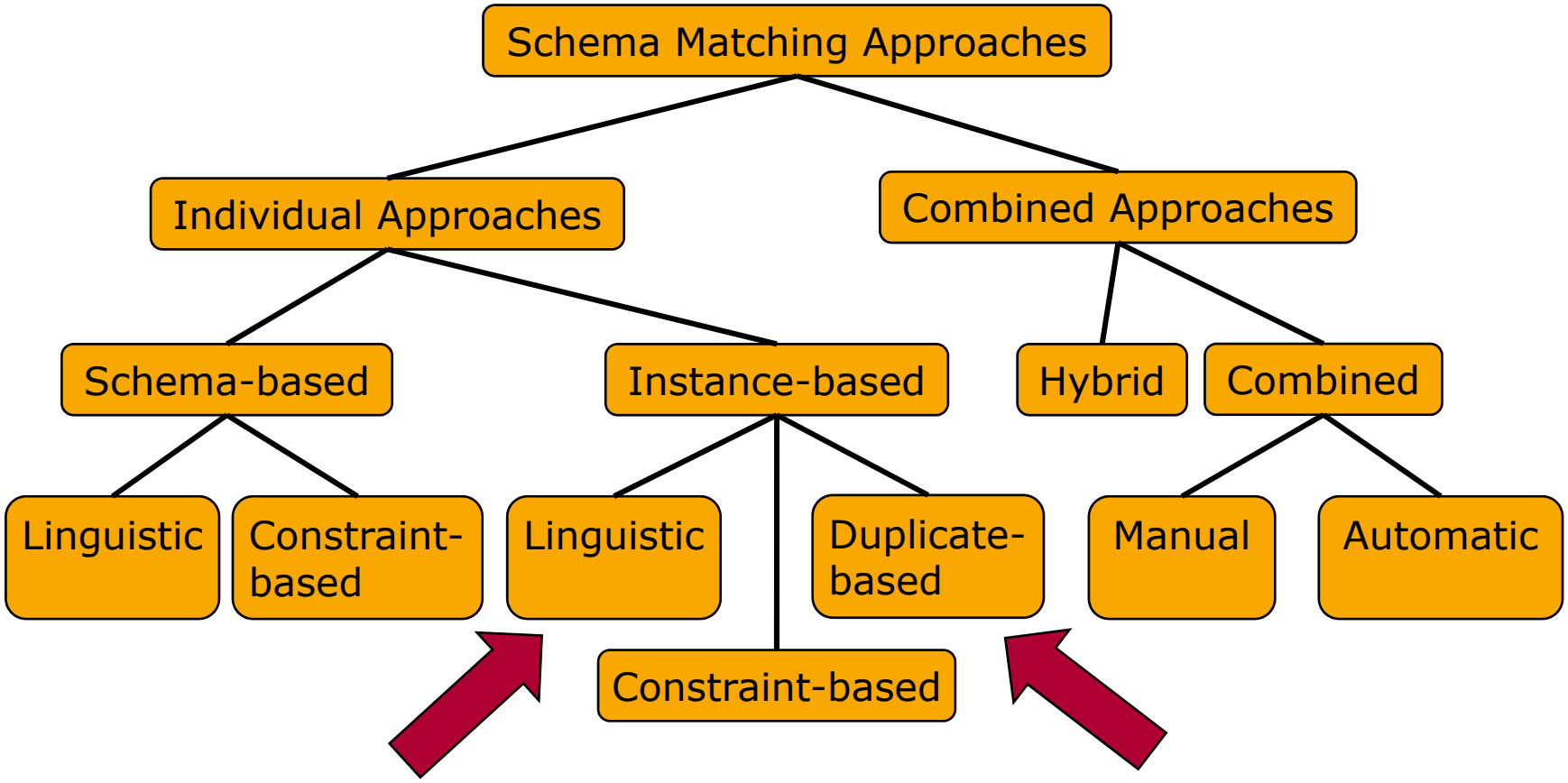
Schemata are

- large
- complex
- foreign
- confusing
- different language
- cryptic




Clio
File
Schemas
Mappings
Help
Source
Target
Schema View
Query
Source schemas
evidences: Record
Set [1,*]
Choice
evidence: Record (evidenceType)
id (ID)
category (NMTOKEN)
type (NMTOKEN)
date (date)
88,9%
initials (string) ?
Set [1,*]
attribute (string)
internal: Record
suggestedEntryName: Record
name (string)
proposedEntryName: Record
name (string)
patternMatches: Record
match: Record
db (string)
from (int)
to (int)
date (date)
88,9%
nonStandardSource: Record
Choice
submisison: Record
dsNumber (string)
sequencing (string)
PIR: Record
acc (string)
id (string)
journalScan: Record
db (string)
pid (string)
comments: Record
Set [1,*]
comment (string)
firstCurated: Record
initials (string)
88,9%
Target Schema
genbank_db: Record
Set [1,*]
genbank: Record
locus (string)
size (string)
Record ?
molecule (string)
division (string)
date (string)
definition (string)
accession (string)
version (string)
gi (string)
keywords (string) ?
Record ?
segment (string)
source: Record
common (string)
taxon (string)
taxon_lineage (string)
Set [0,*] ?
reference: Record
Set [1,*]
Choice
authors (string)
title (string)
journal (string)
remark (string)
medline (string)
pubmed (string)
id (string)
location (string) ?
comment (string) ?
feature_table: Record
Set [1,*]
feature: Record
location (string)
Set [0,*] ?
No File

Schema Matching Classification [RB01]



Instance-based Schema Matching

Instance-based Schema Matching:

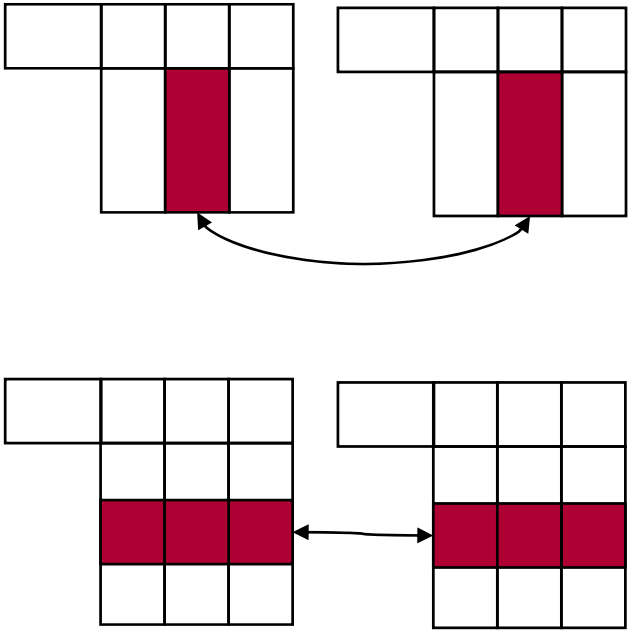
- Correspondences based on similar data values or their properties

Conventional solution: Vertical

- Comparison of columns
- = Attribute classification

Our solution: Horizontal

- Comparison of rows
- = Duplicate detection (despite missing attribute correspondences)



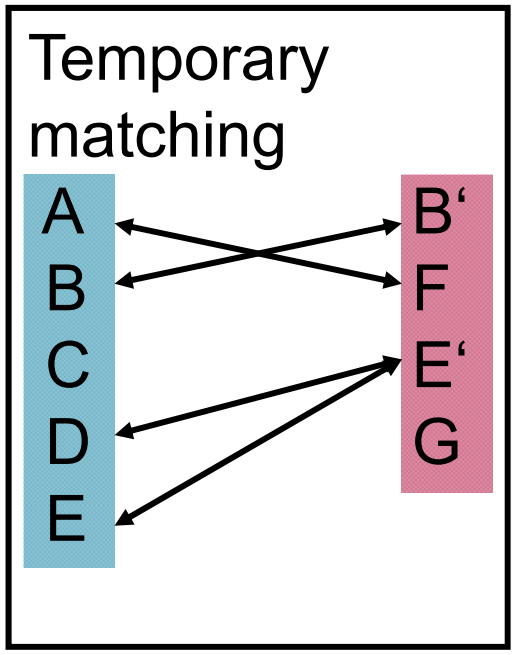
DUMAS Matcher

A	B	C	D	E
Max	Michel	m	601- 4839204	601- 4839204
...	...	...	...	...

?

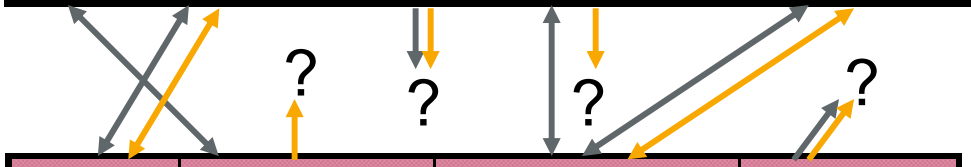
?

B‘	F	E‘	G
Michel	maxm	601- 4839204	UNIX
...	...	...	...

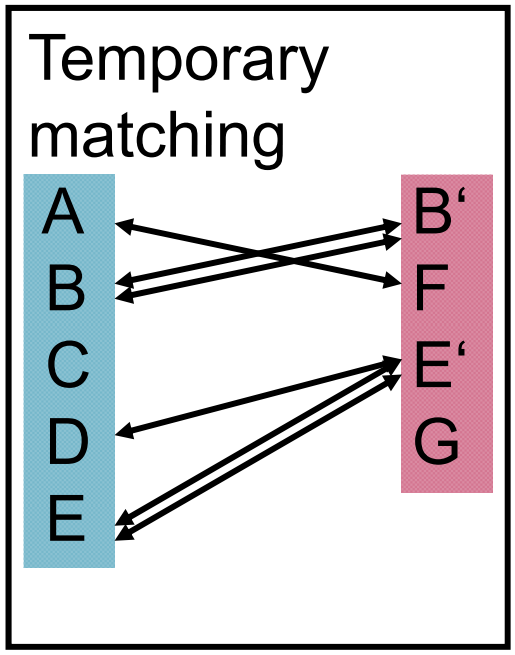


DUMAS Matcher

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



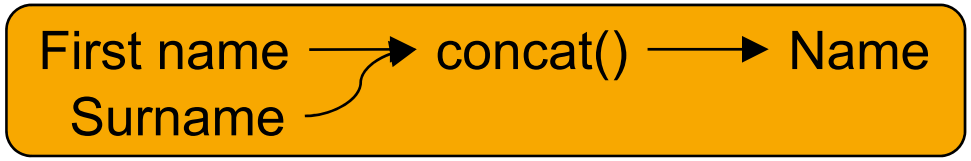
B‘	F	E‘	G
Michel	maxm	601- 4839204	UNIX
Adams	beer	541- 8127164	WinXP



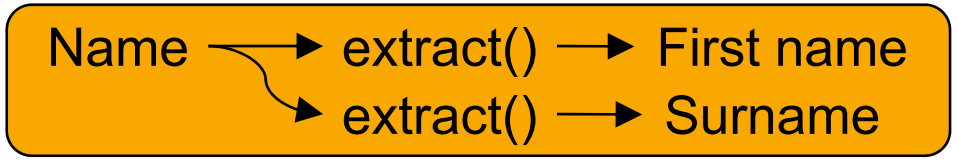
Schema Matching – open problems

- n:1 und 1:n matches
  - Many combinations
  - Many functions
  - Parsing
- Matching in complex schemata
  - Find mapping, not only correspondences
  - Unions and joins

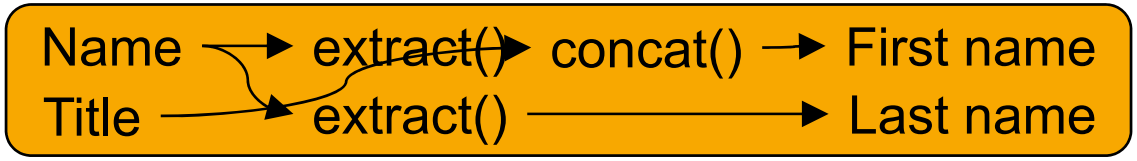
n:1 Matching



1:n Matching



m:n matching



- Tooling: User in the loop!

Overview

- Introductory example
- Schema level integration
  - Schema Mapping
  - Schema Matching
- Data level integration
  - Duplicate detection
  - Data fusion
- ETL management

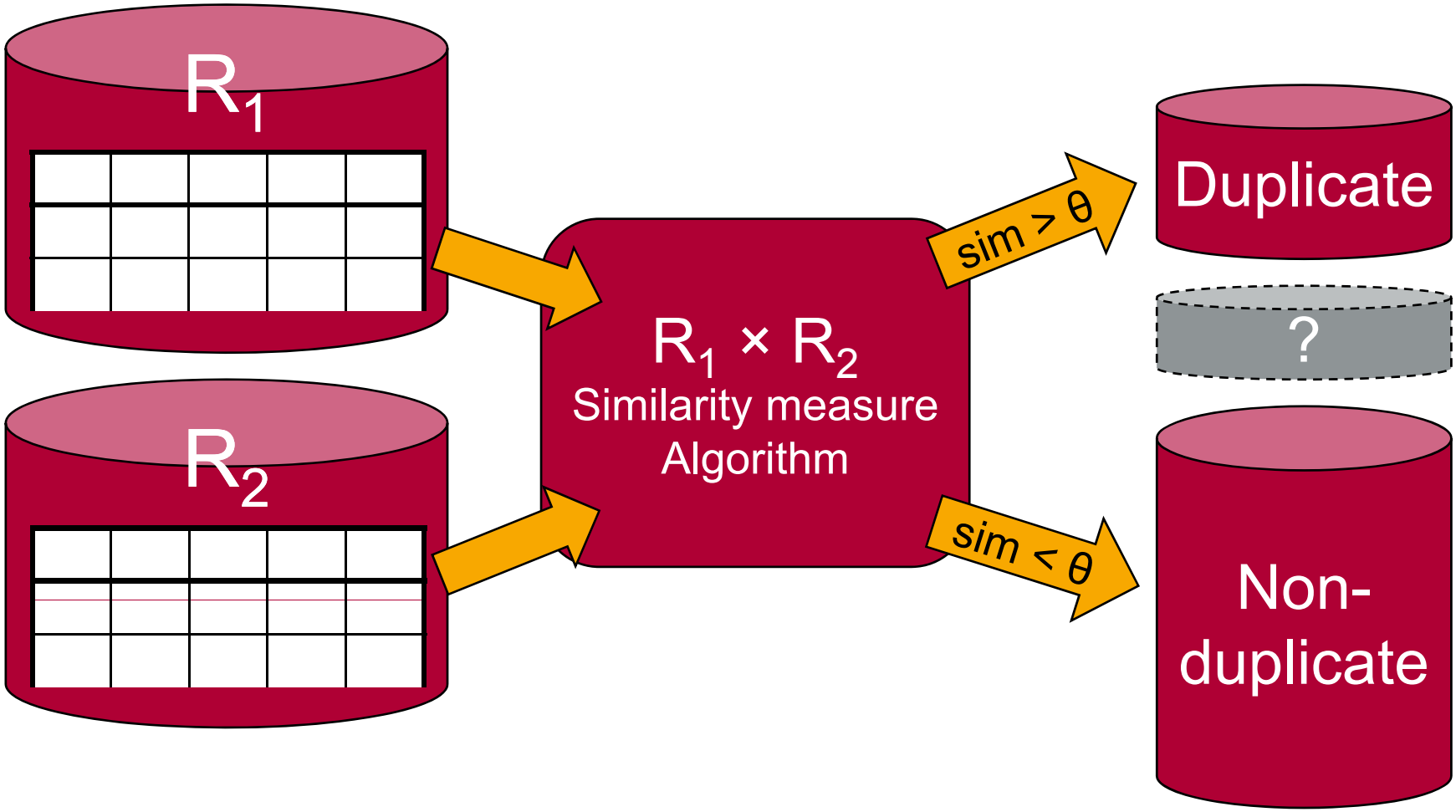
Duplicate Detection

Duplicate detection is the discovery of multiple representations of the same real-world object.

- Problem 1: Representations are not identical.
  - *Fuzzy duplicates*
- Solution: Similarity measures
  - Value- and record-comparisons
  - Domain-dependent or domain-independent

- Problem 2: Data sets are large.
  - Quadratic complexity: Comparison of every pair of records.
- Solution: Algorithms
  - E.g., avoid comparisons by partitioning.

Duplicate Detection



Motivation

- Possible effects
  - Example: Portfolio Management Offers
  - Credit maximum not detected
  - Too low inventory levels
  - No quantity discount for multiple orders
  - Total revenue of preferred customers unknown
  - Multiple mailings of same catalog to same household
- General problems
  - Additional, unnecessary IT expenses
  - Low customer satisfaction
  - Potentials and dangers not detected
  - Poor quality financial data

Customer	Revenue
BMW	20.000
BaMoWe	5.000.000
Bayerische Motorenwerke	300.000
…	…

Ironically, “Duplicate Detection” has many Duplicates

Household matching

Doubles

Duplicate detection

Mixed and split citation problem

Record linkage

Object identification

Match

Deduplication

Fuzzy match

Object consolidation

Entity resolution

Entity clustering

Approximate match

Identity uncertainty

Reference reconciliation

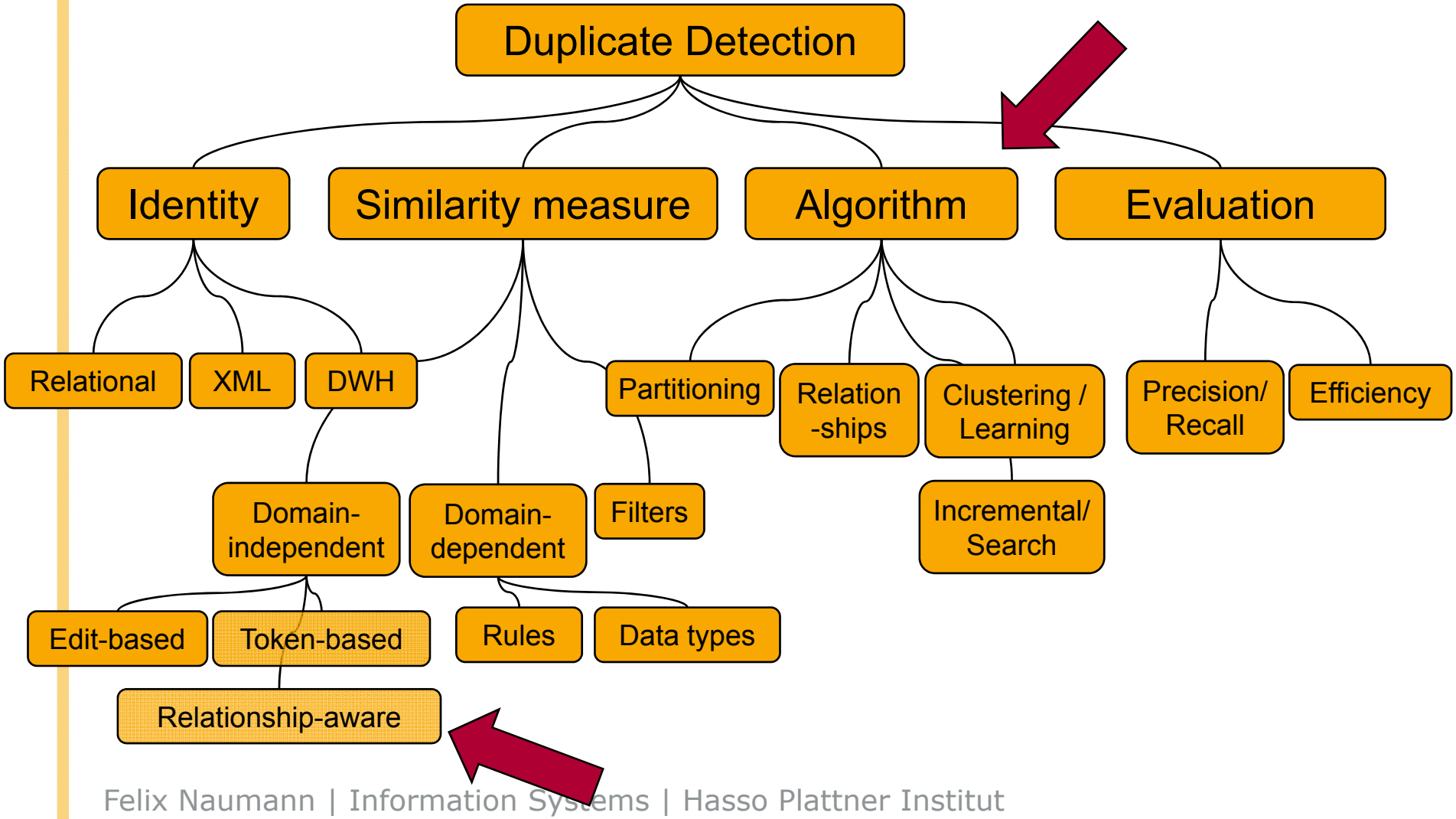
Merge/purge

Hardening soft databases

Householding

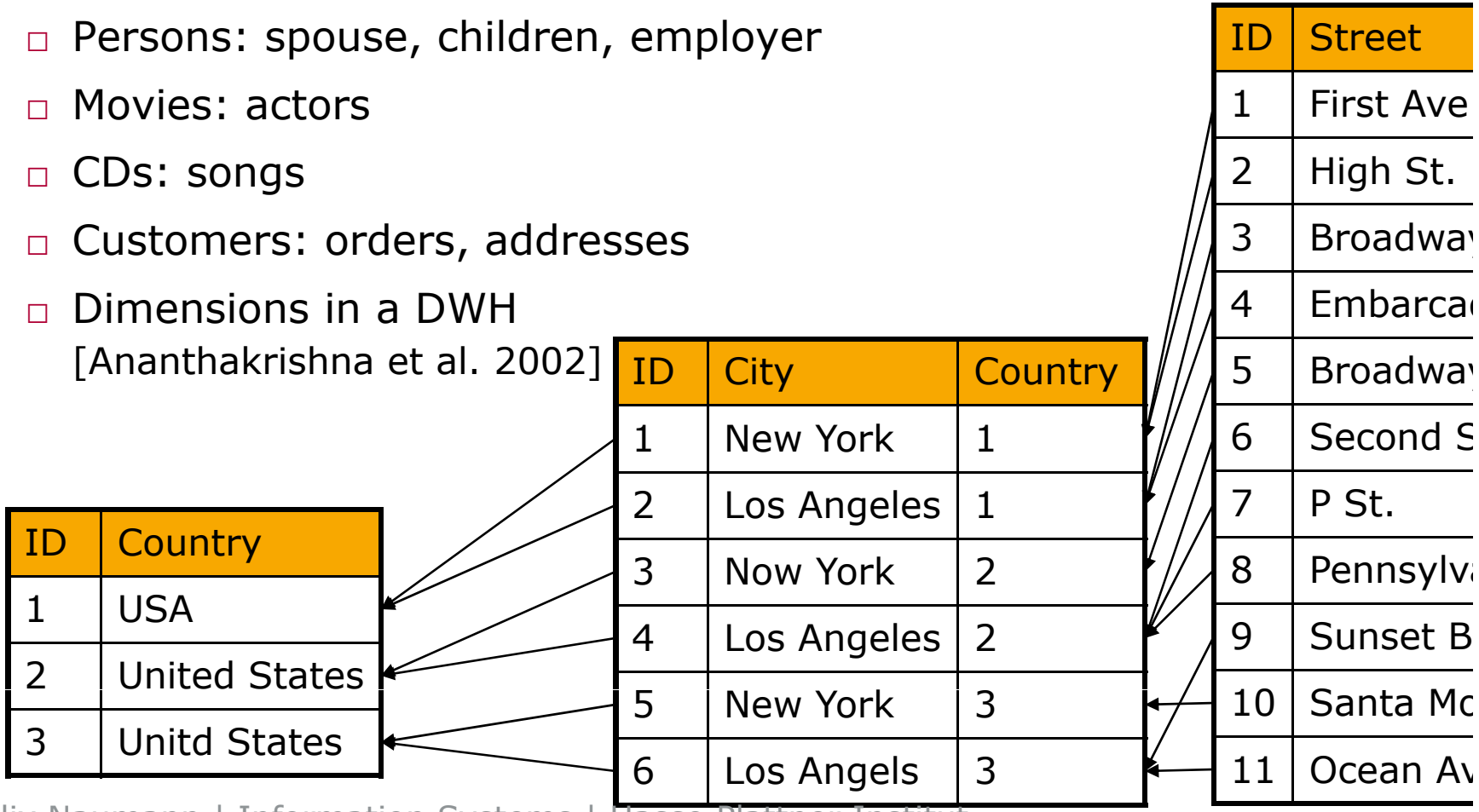
Reference matching

Duplicate Detection – Research



Relationship-aware Similarity Measures

- Idea: Not only values of the records, but values of related records are relevant for similarity.
  - Persons: spouse, children, employer
  - Movies: actors
  - CDs: songs
  - Customers: orders, addresses
  - Dimensions in a DWH
    [Ananthakrishna et al. 2002]

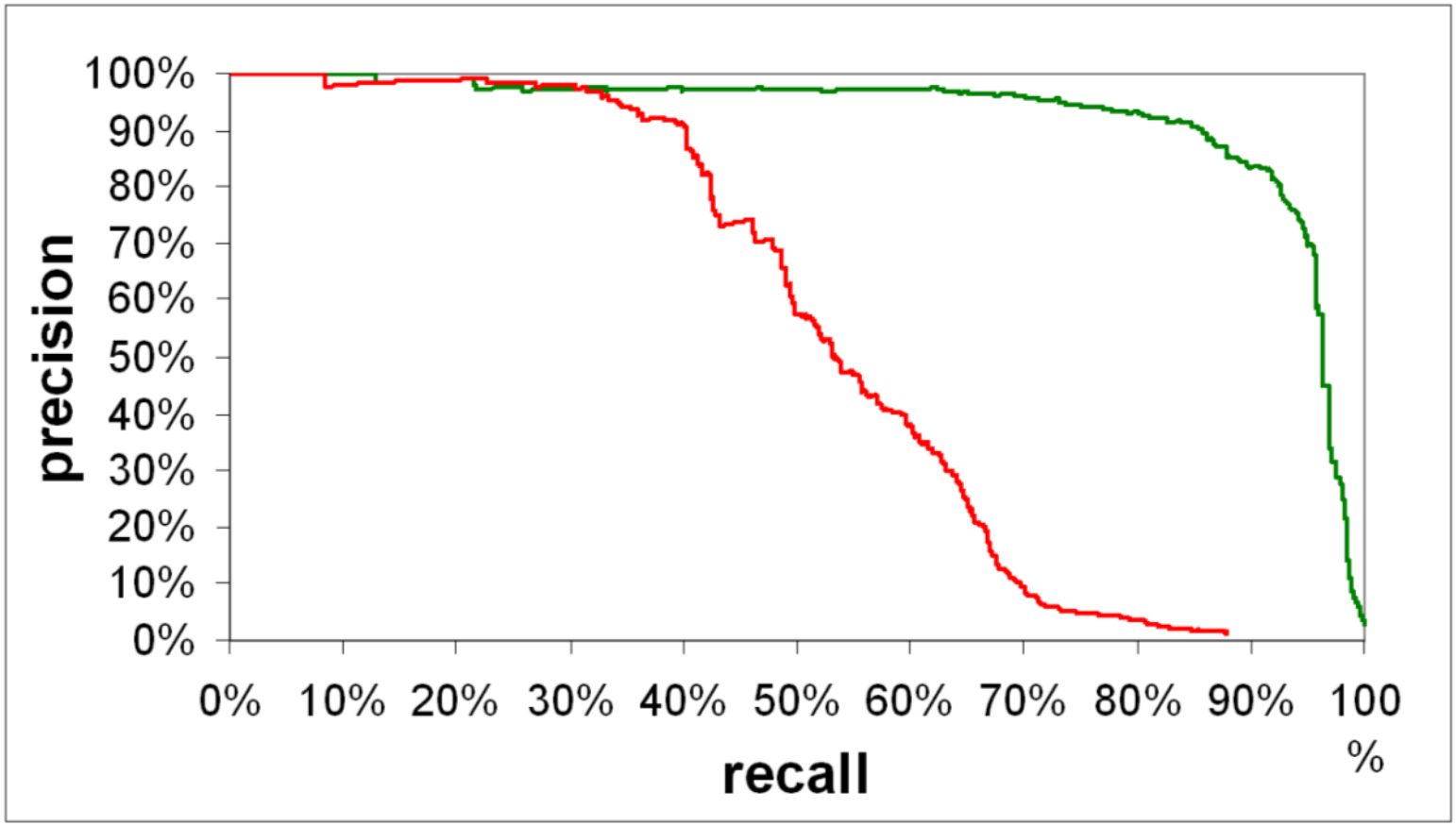


ID	Country
1	USA
2	United States
3	Unitd States

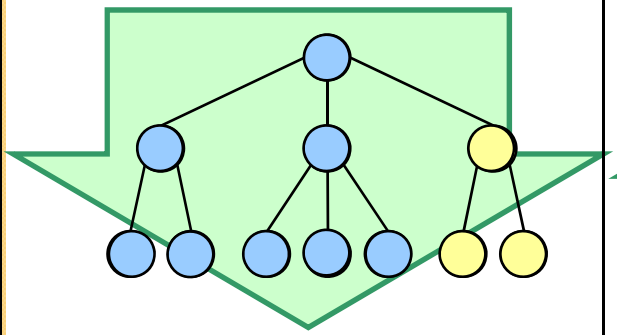
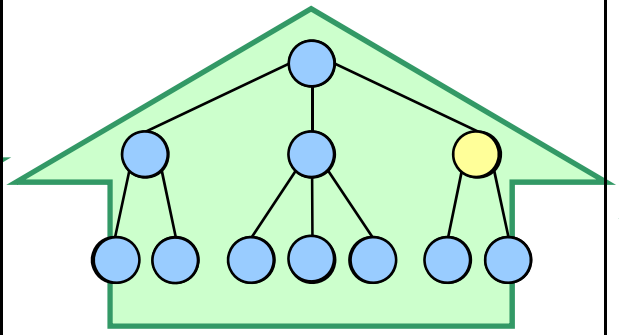
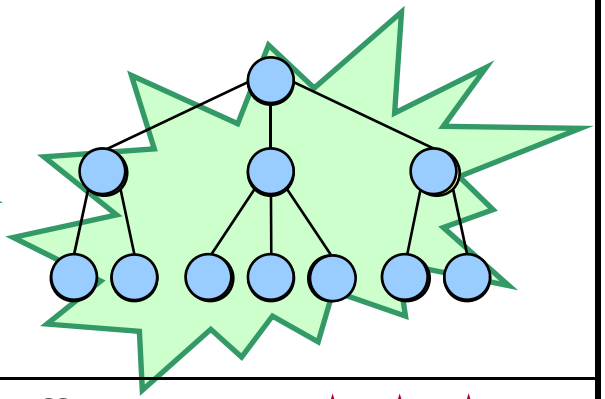
ID	City	Country
1	New York	1
2	Los Angeles	1
3	Now York	2
4	Los Angeles	2
5	New York	3
6	Los Angels	3

ID	Street
1	First Ave
2	High St.
3	Broadwa
4	Embarca
5	Broadwa
6	Second S
7	P St.
8	Pennsylv
9	Sunset B
10	Santa Mo
11	Ocean Av

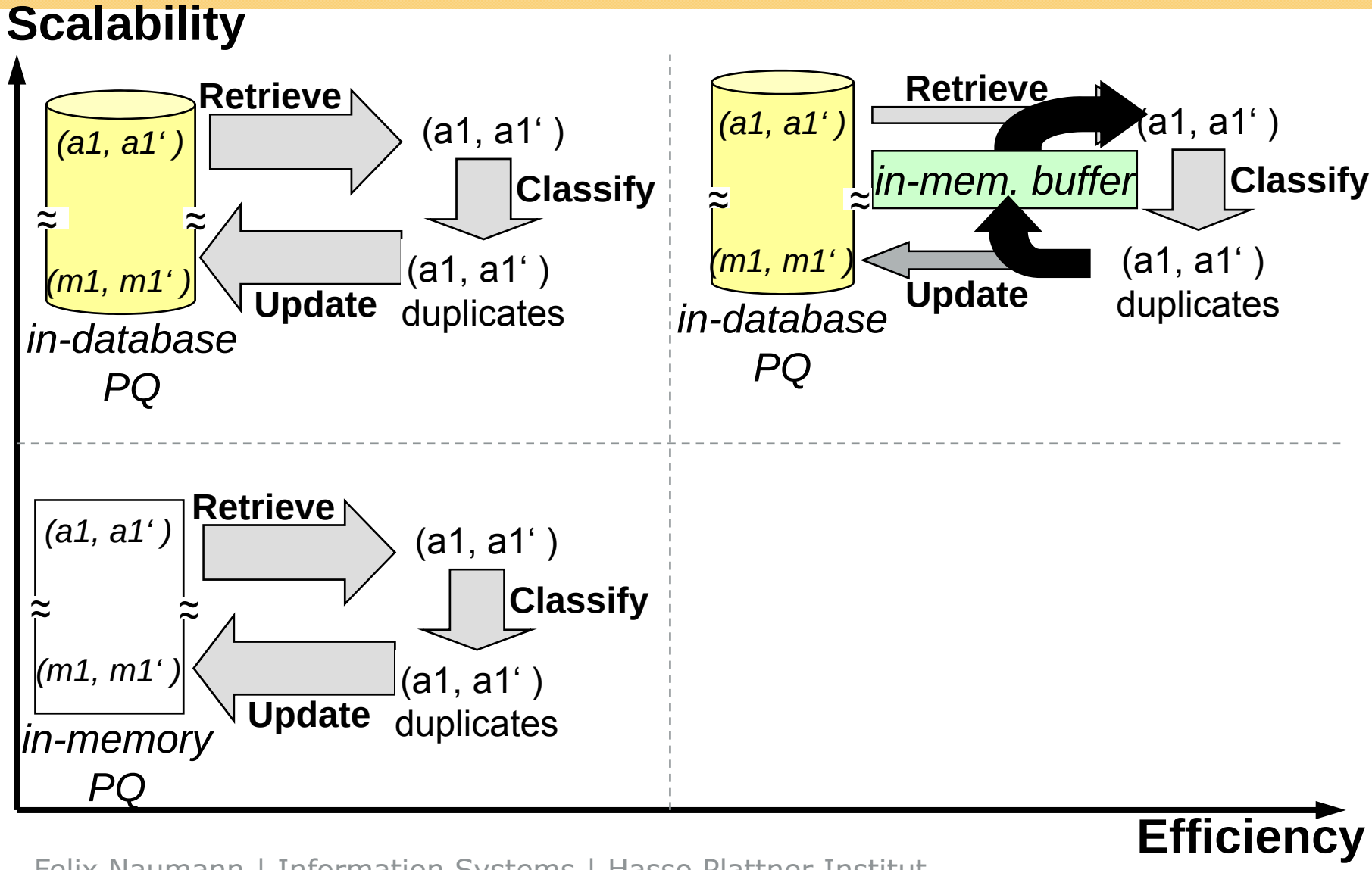
Relationship-aware Similarity Measures – Evaluation



Iterative RADD

Top-down [SIGMOD'05]	Bottom-up [EDBT'06]	From-the-middle [ICDE'06]
		
Effectiveness ★ Efficiency ★★★	Effectiveness ★★ Efficiency ★★	Effectiveness ★★★ Efficiency ★
Further techniques: Object filter; Edit distance filter; Transitivity		

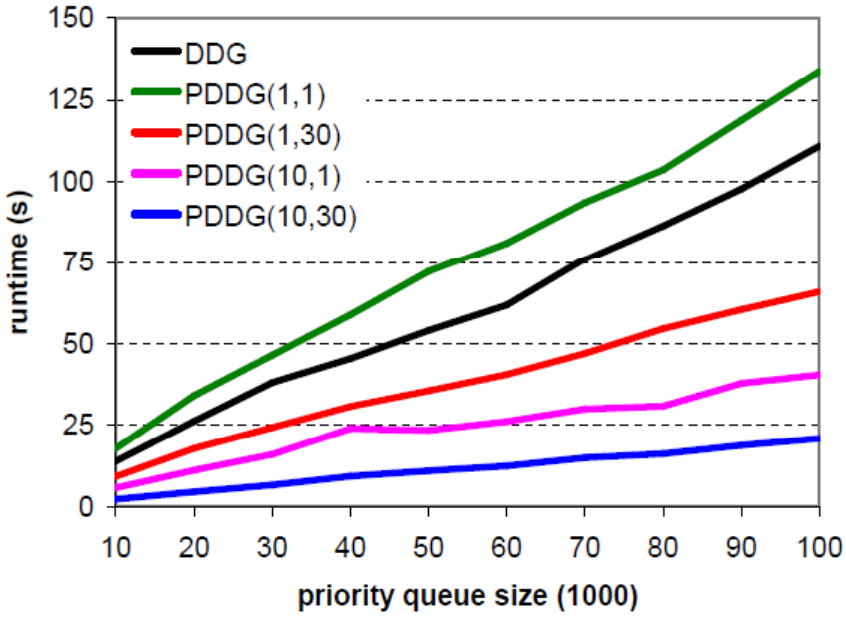
Scalability of the generalized algorithm



Scalability – Comparison

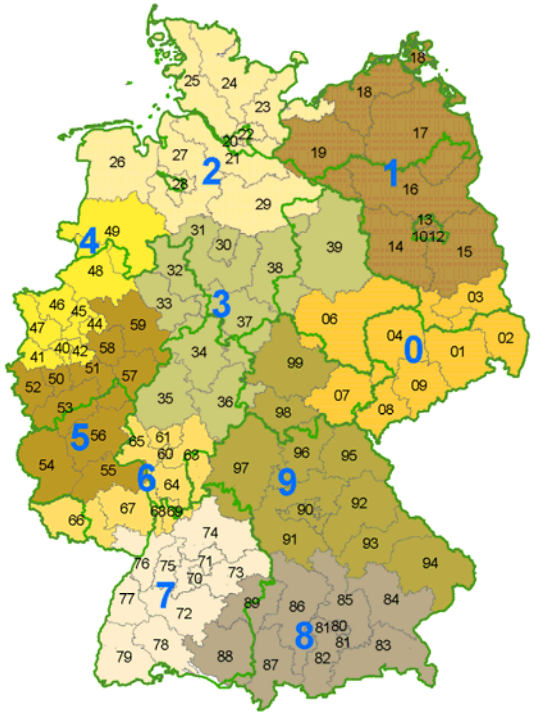
Approach	# candidates	Connectivity	Classif. Time (s)
Singla [27]	1,295	n.a.	n.a.
Dong05 [12]	40,516	13.4	n.a.
RC-ER [2]	45,000/65,000	1.9 / 5.3	100 / 890
RC-ER [4]	97,270	1.9	543 - 690
RelDC [19]	75,000	low - high	180 - 13,000
LinkClus [31]	100,000	10	900
RECUS/BUFF	**1,000,000**	**1.7**	**24,433 (7h)**

Parallelization increases performance:



Partitioning / Blocking

- Partition the records (horizontally) and compare pairs of records only within a partition.
  - Partitioning by first two zip-digits
  - or partition by first letter of surname
  - or …



Source: wikipedia.de

- Idea: Partition multiple times by different criteria.
  - Then apply transitive closure on discovered duplicates.

Sorted Neighborhood
[Hernandez Stolfo 1998]

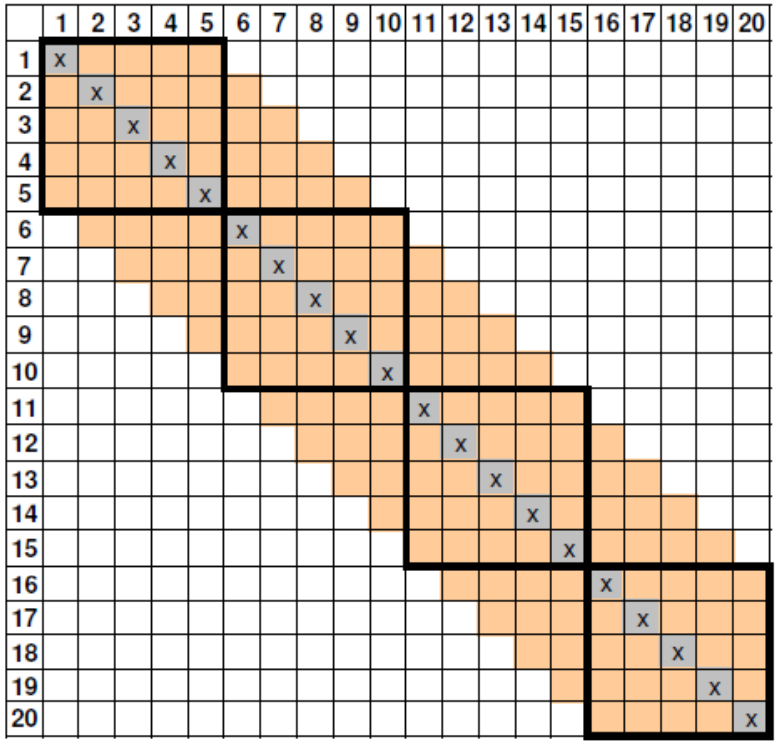
- Idea
  - Sort tuples so that similar tuples are close to each other.
  - Only compare tuples within a small neighborhood (window).

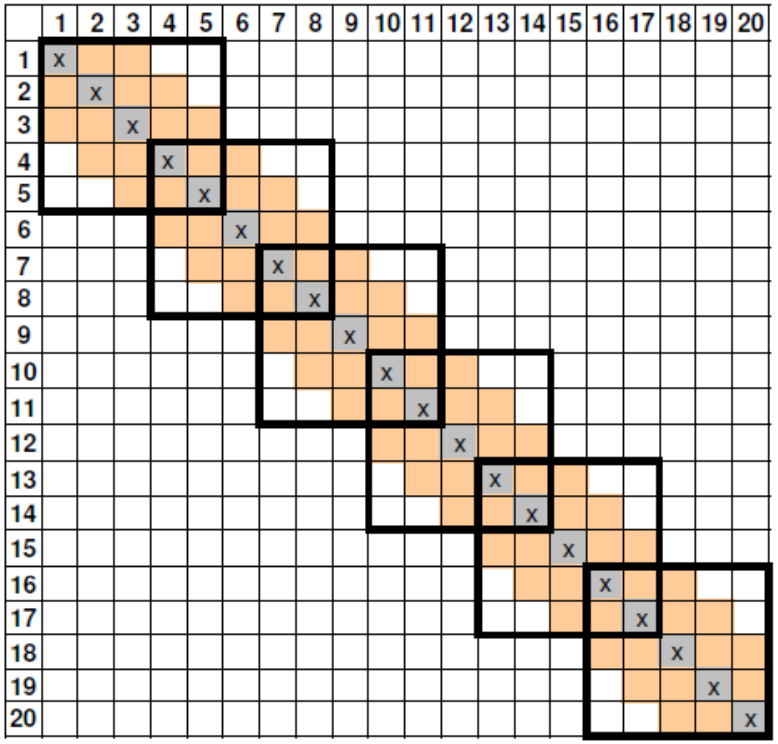
1. Generate key
   - E.g.: SSN+"first 3 letters of name" + ...
2. Sort by key
   - Similar tuples end up close to each other.
3. Slide window over sorted tuples
   - Compare all pairs of tuples within window.

- Problems
  - Choice of key
  - Choice of window size
- Complexity: At least 3 passes over data
  - Sorting!

Current Work on Algorithms

- Dynamic window size adaption for sorted neighborhoods
- Parallelization of duplicate detection
- Scalability of duplicate detection
- Generalization of blocking and sorted neighborhood algorithms

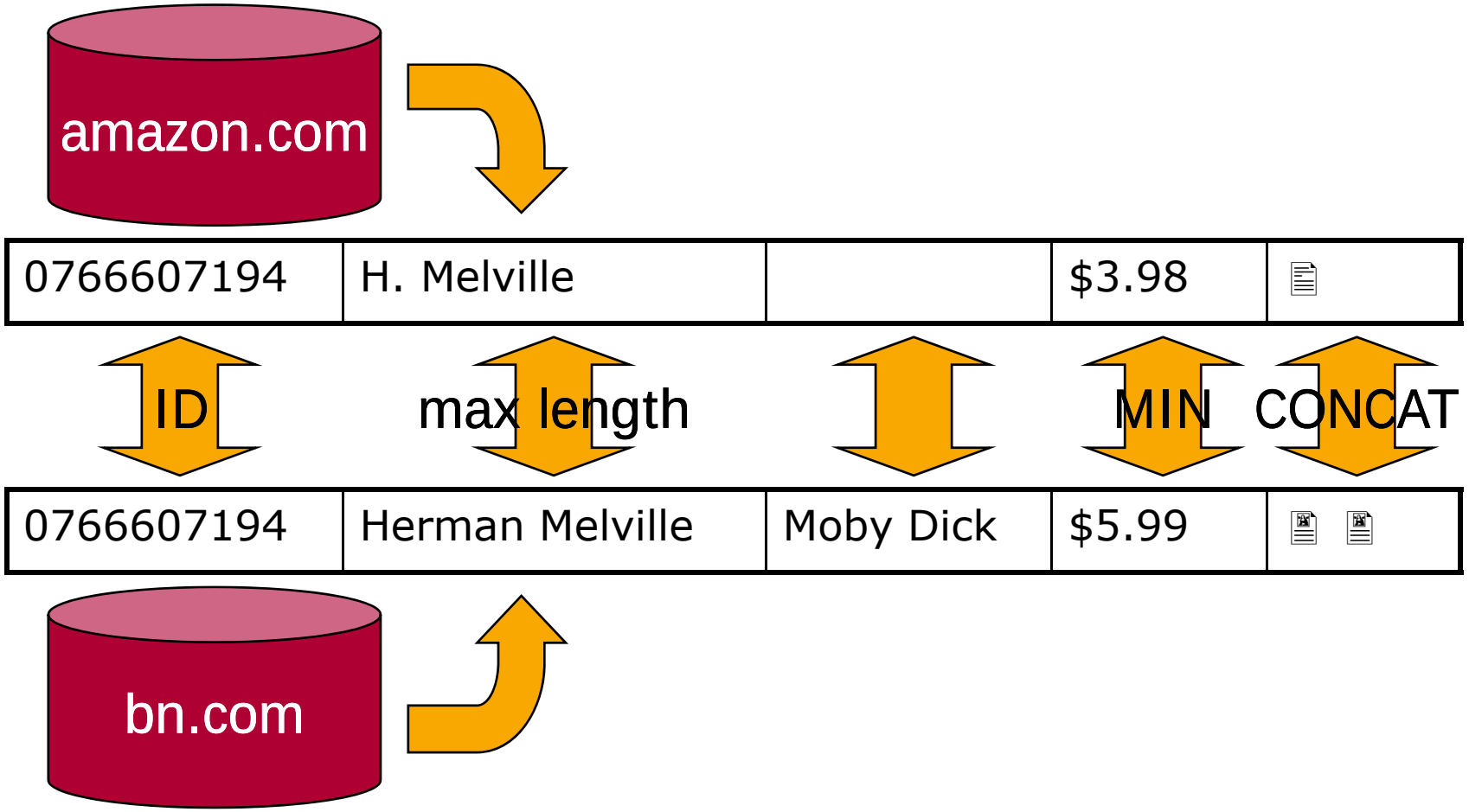




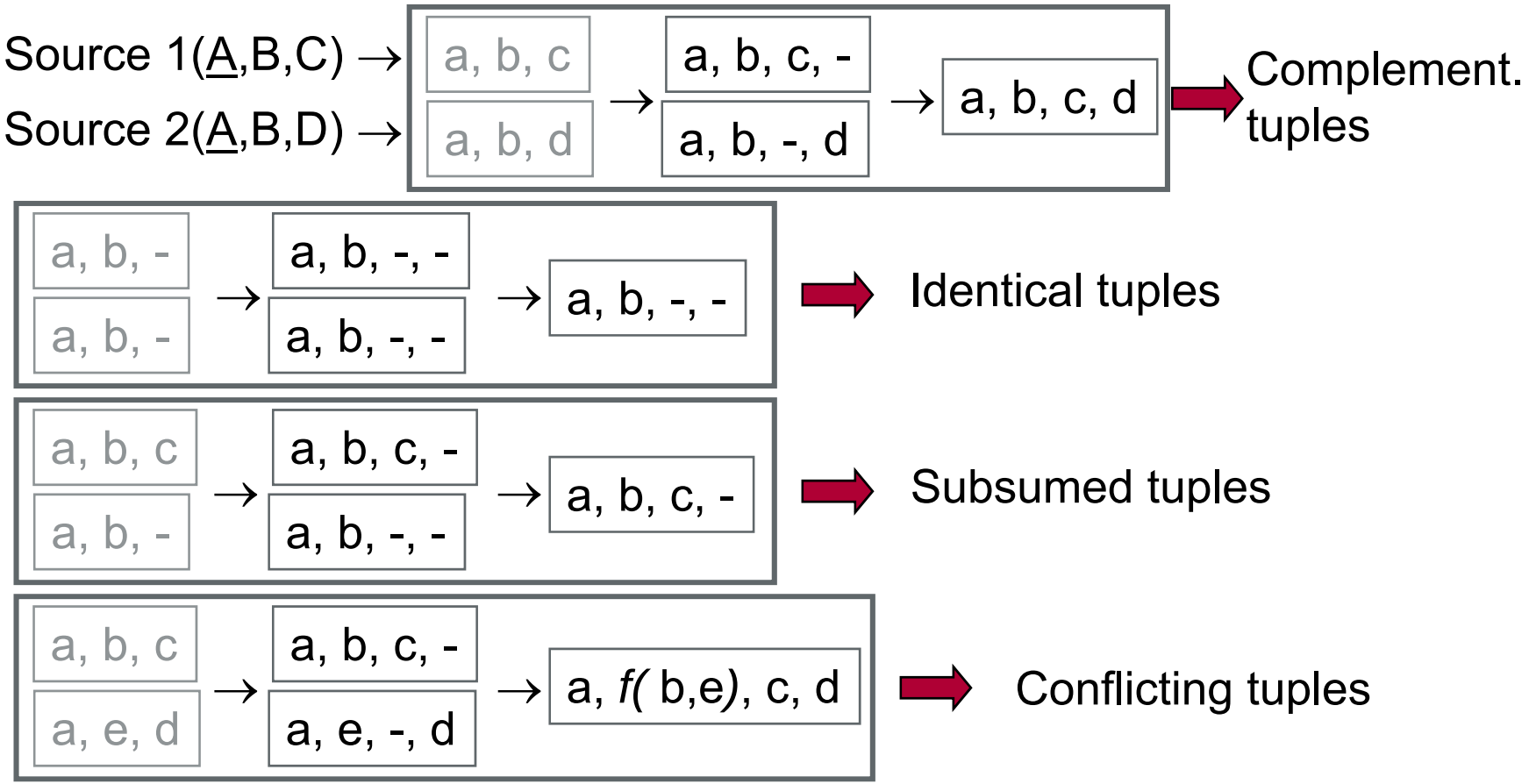
Overview

- Introductory example
- Schema level integration
  - Schema Mapping
  - Schema Matching
- Data level integration
  - Duplicate detection
  - Data fusion
- ETL management

Data Fusion


amazon.com
0766607194
H. Melville
$3.98
ID
max length
MIN
CONCAT
0766607194
Herman Melville
Moby Dick
$5.99
bn.com

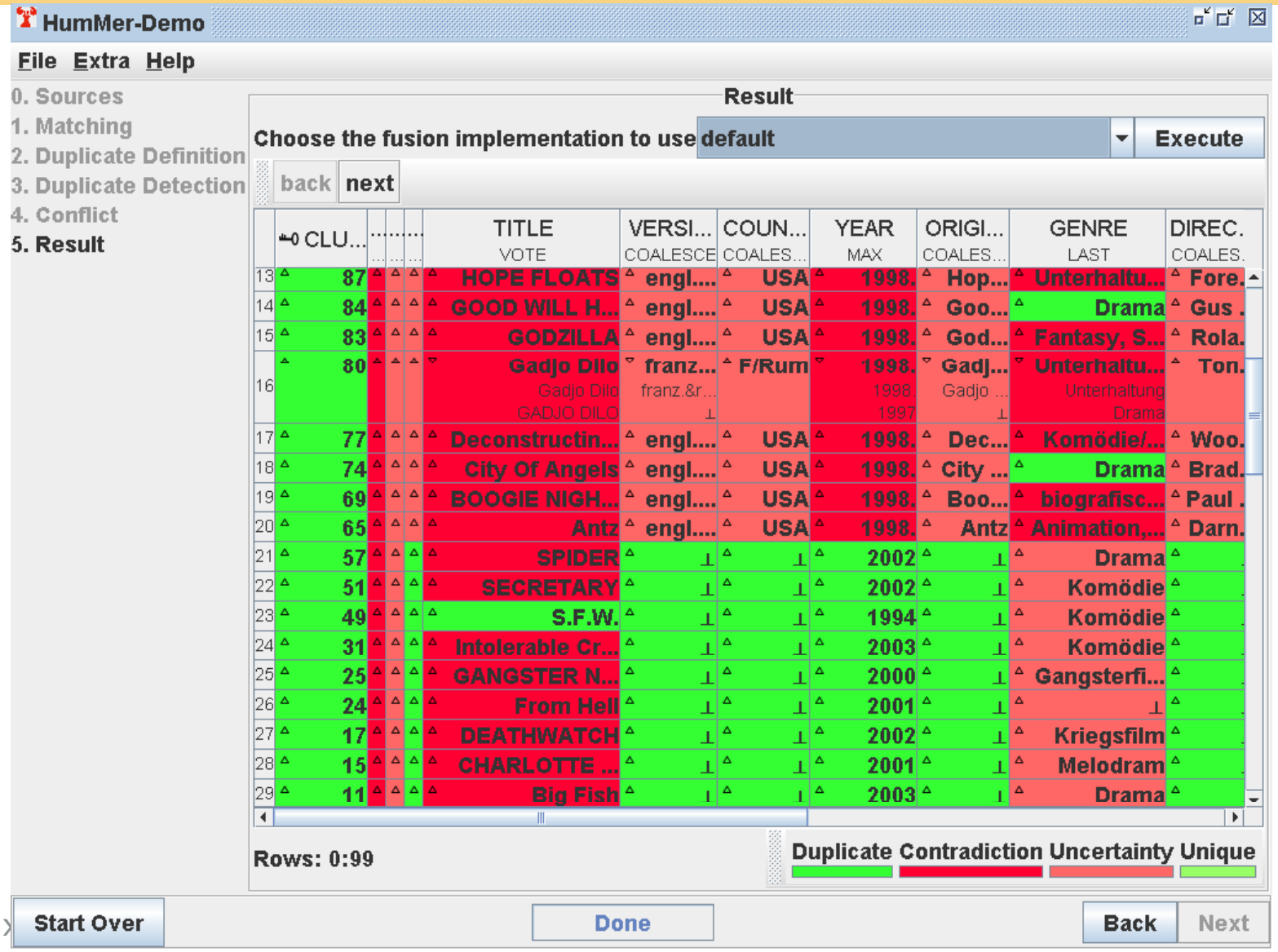
“Proper” Data Fusion



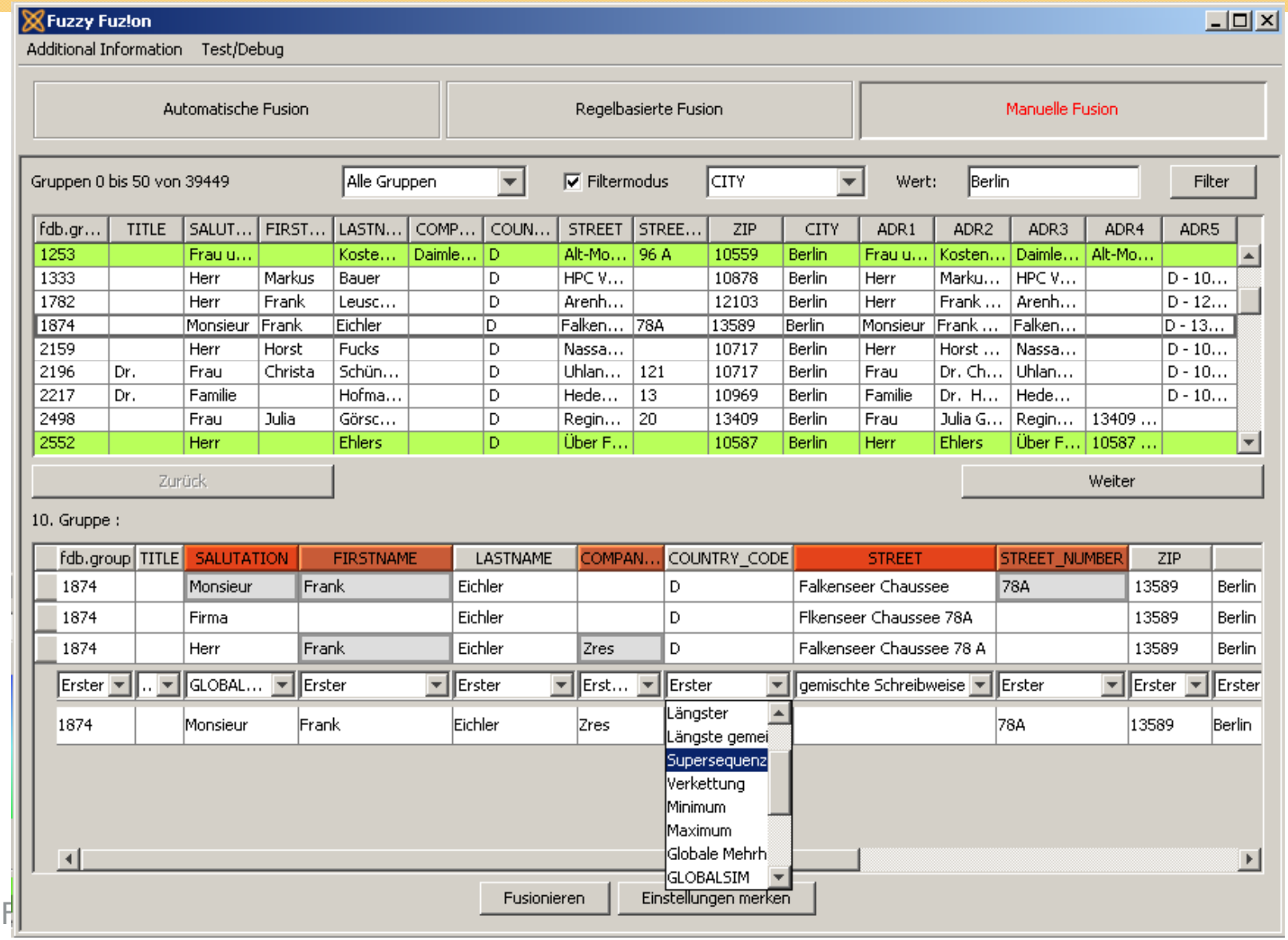
Conflict Resolution Functions

Min, Max, Sum, Count, Avg, StdDev	Standard aggregation
Random	Random choice
First, Last	Choose first/last value; depends on order
Longest, Shortest	Choose longest/shortest value
Choose(*source*)	Choose value froma particular source
ChooseDepending(*col*, *val*)	Choose depending on *val* in other column *col*
Vote	Majority decision
Coalesce	Choose first non-null value
Group, Concat	Group or concatenate all values
MostRecent	Choose most recent (up-to-date) value
MostAbstract, MostSpecific	Use a taxonomy / ontology
….	….

Visualization of Integrated Data



Tool-based Data Fusion



Overview

- Introductory example
- Schema level integration
  - Schema Mapping
  - Schema Matching
- Data level integration
  - Duplicate detection
  - Data fusion
- ETL management (with Alexander Albrecht)

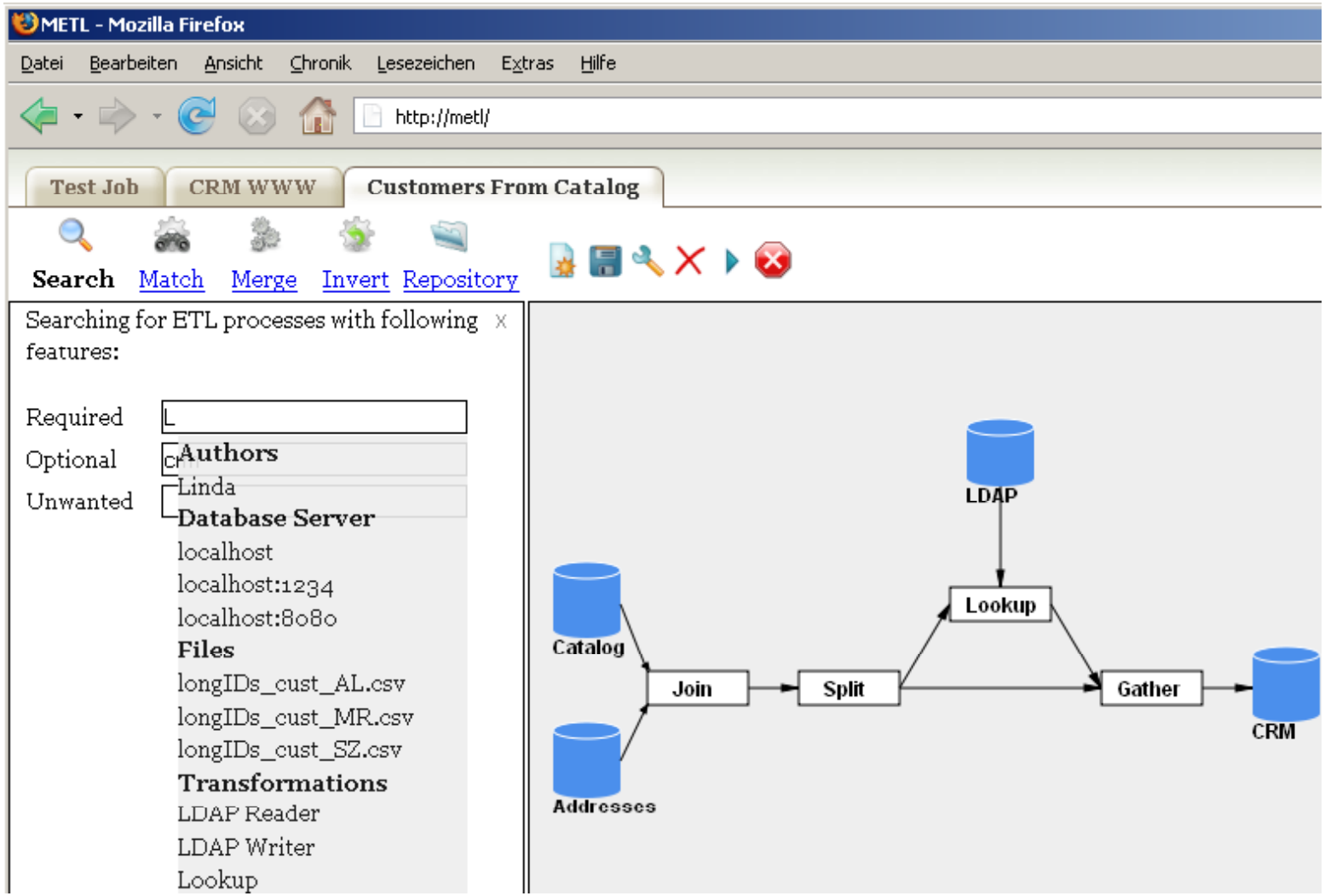
ETL Process Management

- State of the art
  - Many data sources from different organizational areas are involved in a variety of data integration projects using ETL.
  - ETL processes may encompass shared data sources, same data targets, common sub-processes, and transformations
- Diagnosis
  - No common method, approach, or framework to uniformly manage entire ETL processes
- With METL (Managing ETL) we are implementing a next generation ETL tool that supports high-level ETL management.

Management Operators in METL

- **Search** – retrieves all ETL processes that satisfy the specified search query.
- **Match** – given an ETL (sub-)process it finds all corresponding ETL (sub-)processes that extract, transform, or load common data in a similar way.
- **Merge** – takes one or more ETL processes as input and returns a merged ETL process.
- **Invert** – feeds the output of one ETL process back to its sources
- **Create** – populates system from a variety of sources (mappings, SQL, scripts, mappings, ...)
- **Import/Deploy** – interface for existing ETL tools, based on a common representation of ETL processes

METL Prototype

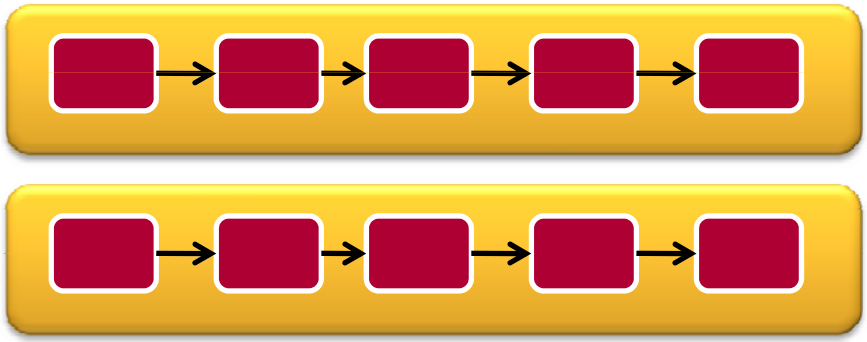


Search & Match

- **Search** – similar to search engines
  - Search predicates
  - Optionally prepended by a context-label
- **`phone +table:addresses +stage:lookup -author:Alice`**
- Ranking based on relevance to specified search terms using TF/IDF-like term weighting

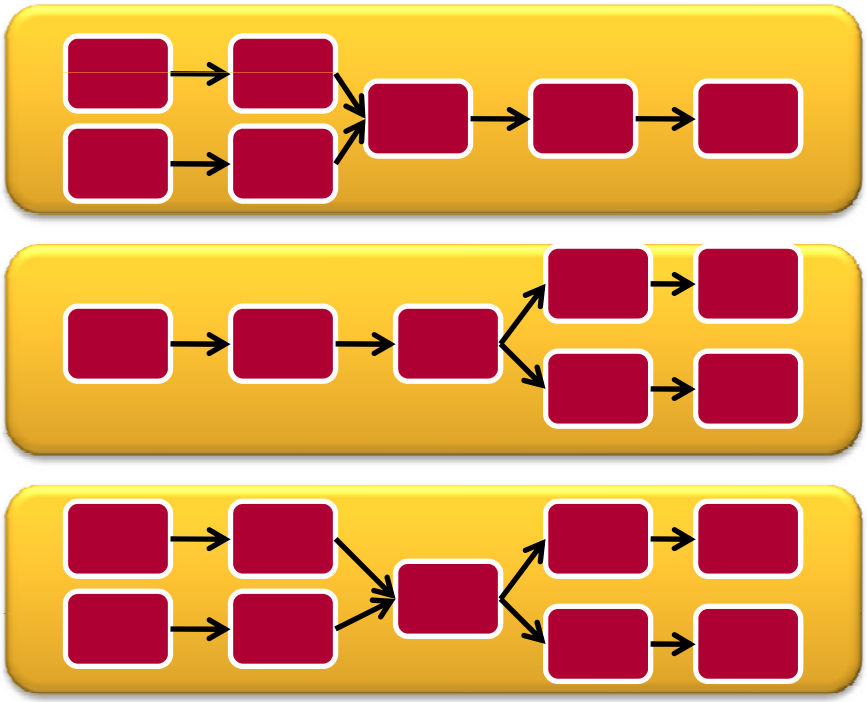
- **Match** – easy-to-use access to ETL processes in repository
- Problem: Suitable similarity measure for ETL processes
  - Variety of ETL features
  - Semantic or syntactic heterogeneity
- Structure-aware Match operator
  - Position in ETL process
  - Data-Schema
  - Type of operator

Merge



Benefits

- Better utilization of shared resources
- Latency improvement and reduced amount of data transmission
- Enhancement of performance compared to performing all processes in a separate run
- Single view of all information that was originally processed separately



Invert

- Invert output at a given transformation within the ETL process
- Generates for source table S a corresponding source table S' with cleansed data.
  - ETL-equivalence
- Benefits
  - Consolidated and cleaned data is fed back to the sources to ensure data quality for applications on top of original sources.
  - Propagation of corrections to sources improves future ETL projects
    - Avoid multiple corrections of same error
  - Application area: MDM

Summary

- Introductory example
- Schema level integration
  - Schema Mapping
  - Schema Matching
- Data level integration
  - Duplicate detection
  - Data fusion
- ETL management

- Other topics of Interest
  - Data Profiling
  - Bulk Annotation
  - Peer data management
  - Service Depot

naumann@hpi.uni-potsdam.de