HumMer and the three steps of information integration

Modena, 26th November 2005
Felix Naumann
Jens Bleiholder
naumann@informatik.hu-berlin.de
Humboldt-Universität zu Berlin

Humboldt-Universität zu Berlin

Humboldt-Universität zu Berlin

- Wilhelm and Alexander von Humboldt
- Unity of Teaching and Research
- 29 Nobel-prize winners
  - Mommsen, Hertz, Koch, Hahn, Planck, Einstein,...
- 38,000 students, (1100 computer sciences)
- 560 (tenured) professors
  - 11 + 10 in computer sciences

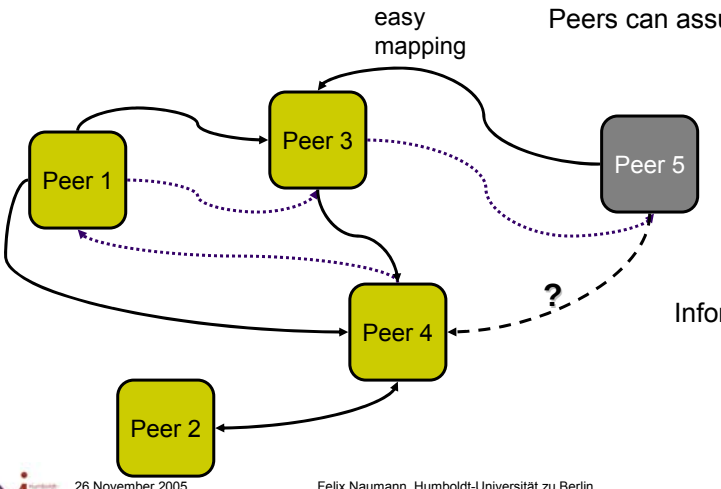
Research Group „Information Integration“

- Head: Felix Naumann (naumann@informatik.hu-berlin.de)
- Researchers
  - Jens Bleiholder (bleiho@informatik.hu-berlin.de)
    - Data fusion in relational data
  - Melanie Weis (mweis@informatik.hu-berlin.de)
    - Object identification in XML Data
- Affiliated
  - Armin Roth (aroth@informatik.hu-berlin.de)
    - Data quality in Peer-Data-Management-Systems
  - Alexander Bilke (bilke@cs.tu-berlin.de)
    - Schema Matching
- Research topics around II
  - Object identification
  - Data fusion
  - Optimization
  - Visualization

Further Research Topics

- System P
  - Peer Data Management (PDMS) [BTW'05]
- Bioinformatics
  - BioFast [SIGMOD-Record'04, WebDB'05, …]
    - Links and Paths through Life Sciences Sources
    - With UMD
  - Aladin [CIDR'05]
    - Almost Automatic Data Integration
    - With Ulf Leser (HU Bioinformatics)
- Information Quality
  - ICIQ community and conference (MIT)
- Cooperations with IBM
  - Clio (IBM Almaden)
  - DB2 Wrapper for Search Engines (IBM SVL)
  - DB2 community (IBM Böblingen)

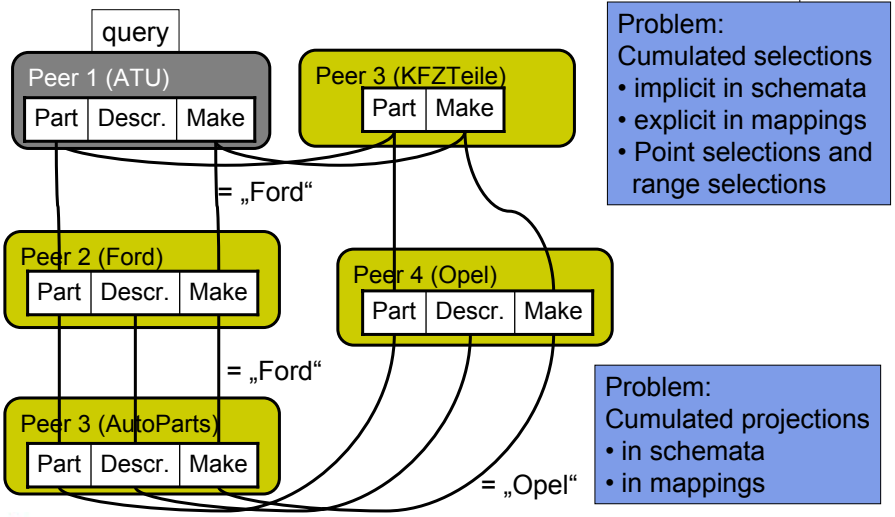
Information Quality for Peer-Data-Management (IQ & PDMS)



Peers can assume many roles:
- data source
- mediator
- wrapper
- querier

Information quality:
- data source
- mapping
- selection
- projection

PDMS: Incomplete and Selective Mappings

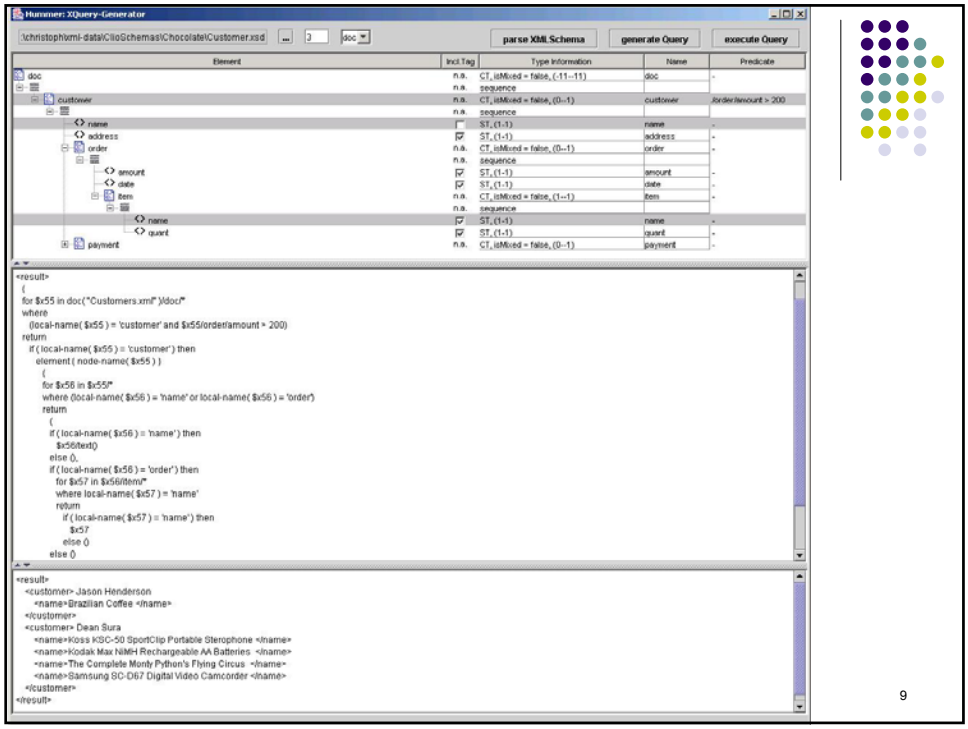


Problem:
Cumulated selections
- implicit in schemata
- explicit in mappings
- Point selections and range selections

Problem:
Cumulated projections
- in schemata
- in mappings

XML related

- XStruct
  - Automatic XML Schema Extraction
    - From many XML files
    - From large XML files
- XQuery Assistance
  - Graphical interface to build XQueries
    - Based on a given Schema
    - Currently: Selection and Projection



9

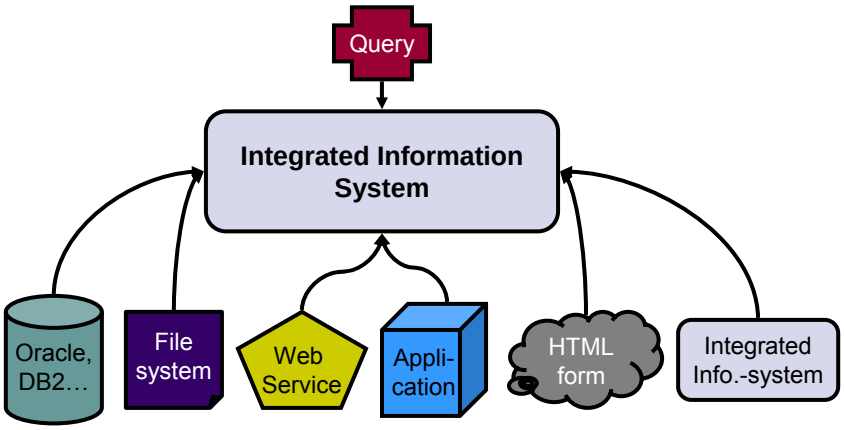
ALmost Automatic Data INtegration – ALADIN (with Ulf Leser)

- Observations
  - Data sources have only one "type" of object
    - Genes, proteins, …
  - Objects have an ID (accession number)
  - Objects have annotations
    - Semi-structured, nested
  - Databases heavily cross-reference each other
    - But cannot keep with the pace of data production
  - Important annotations are often free text
    - which can be compared using text mining
- 5 steps to integration
- Source-specific
  - Download source, parse, import into RDBMS
  - Guess primary objects
  - Guess (hierarchically structured) annotation
- Across data sources
  - Guess cross-references
  - Guess duplicates

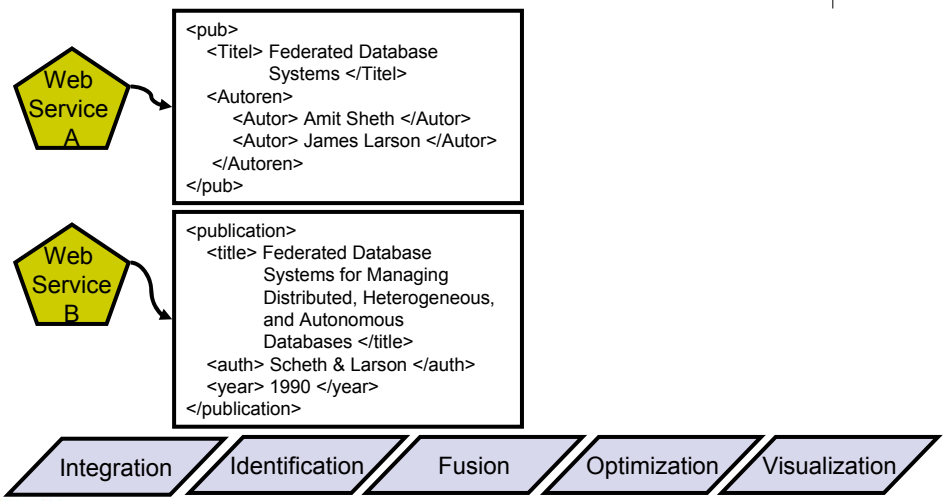
Overview

- Introduction and Motivation
- Information Integration in Three Steps
  1. Schema Mapping
     - And Schema Matching
  2. Duplicate Detection
  3. Data Fusion
- HumMer Architecture
- HumMer Demo

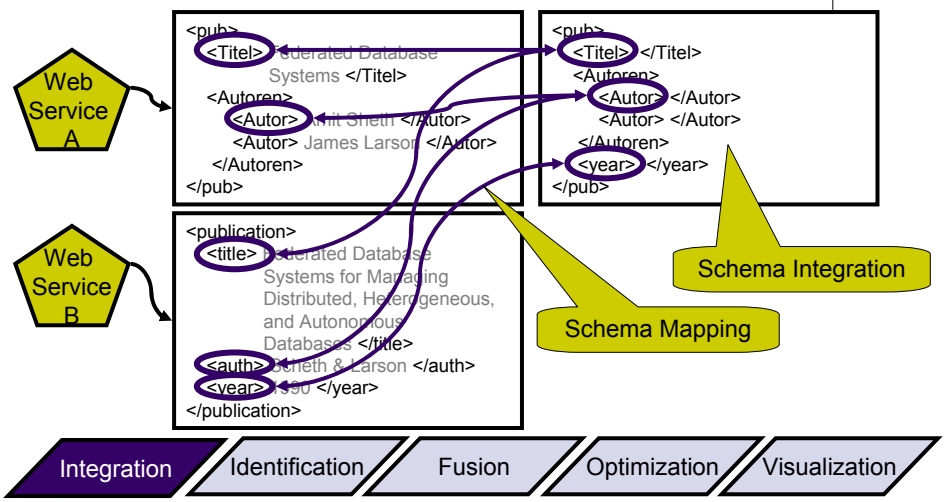
Integrated Information Systems

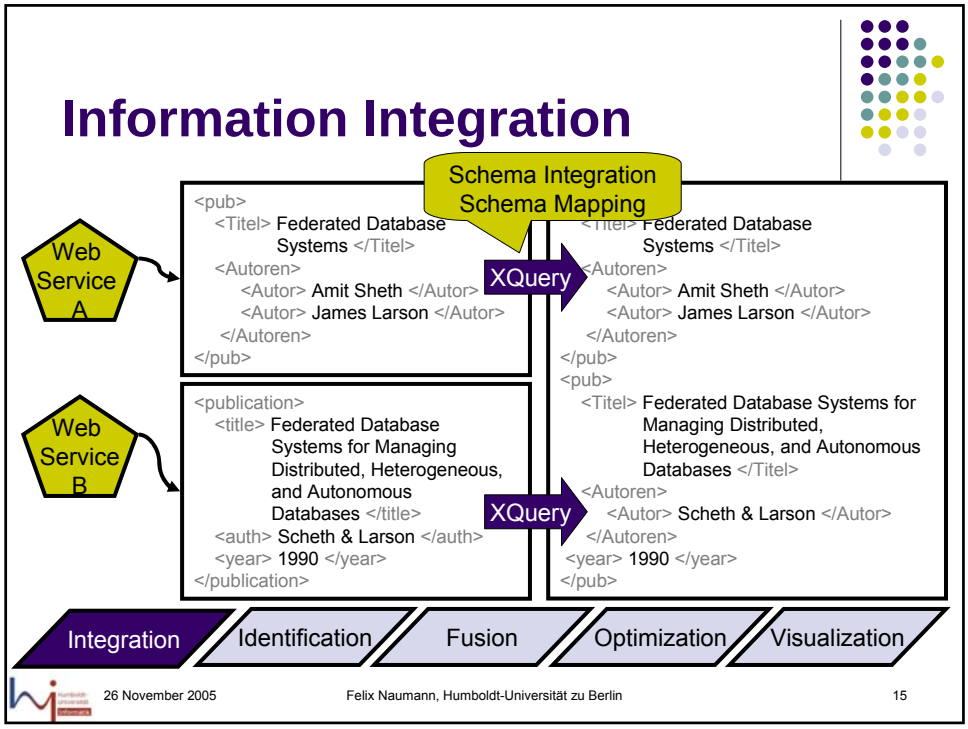


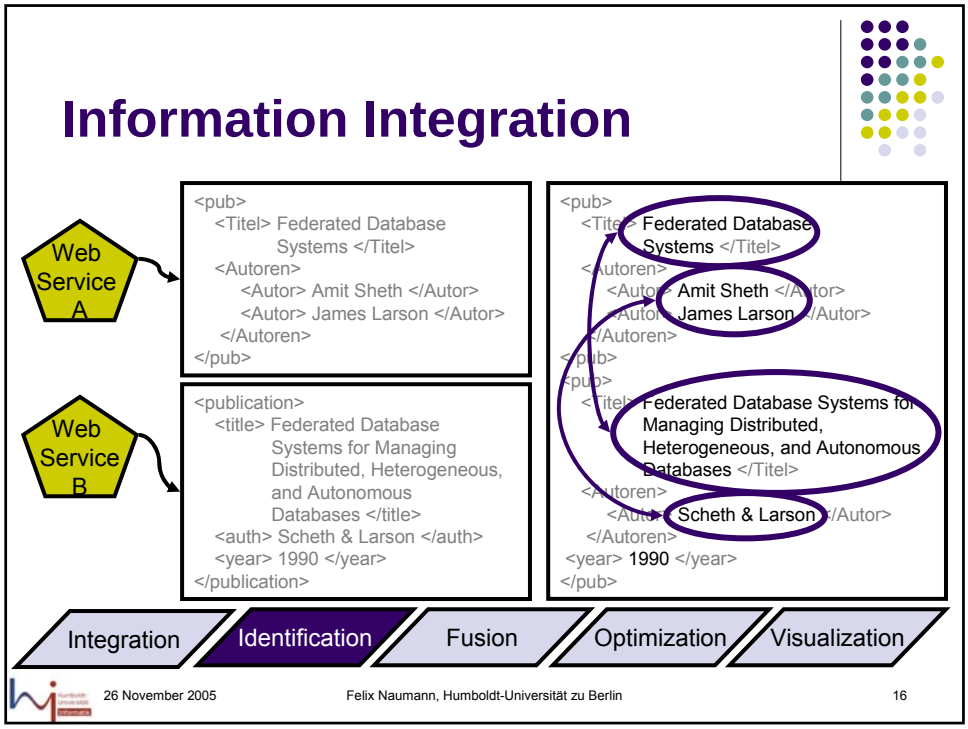
Information Integration

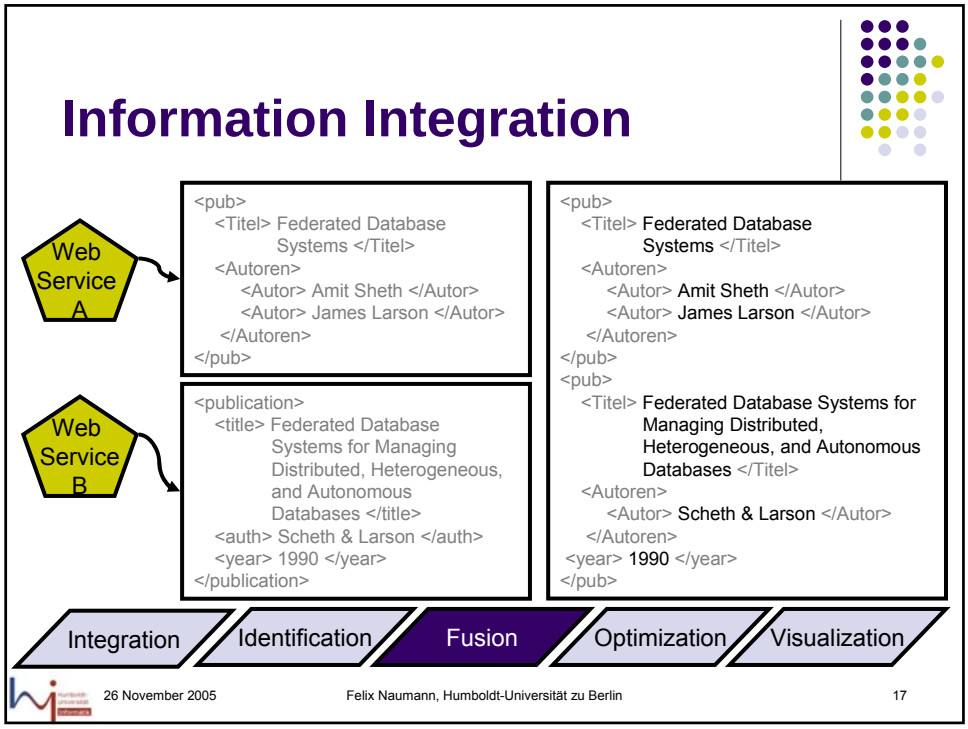


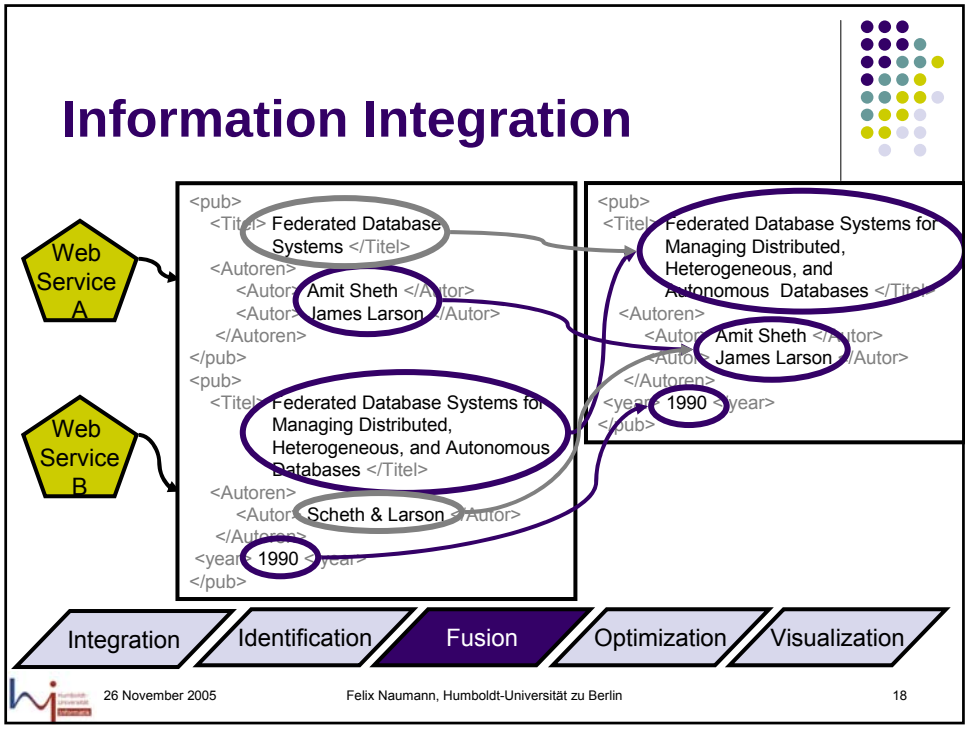
Information Integration



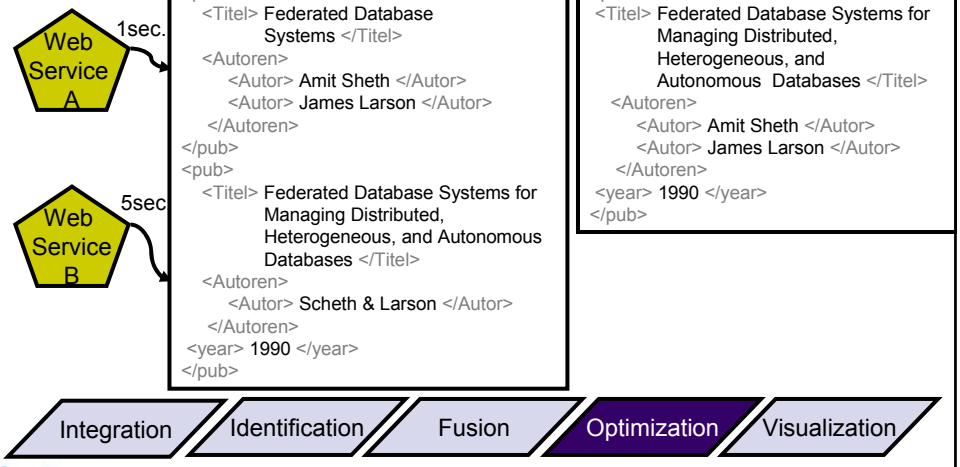

Information Integration
Schema Integration
Schema Mapping
Web Service A
Web Service B
XQuery
XQuery
<pub>
<Titel> Federated Database Systems </Titel>
<Autoren>
<Autor> Amit Sheth </Autor>
<Autor> James Larson </Autor>
</Autoren>
</pub>
<publication>
<title> Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases </title>
<auth> Scheth & Larson </auth>
<year> 1990 </year>
</publication>
<Titel> Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases </Titel>
<Autor> Scheth & Larson </Autor>
Integration
Identification
Fusion
Optimization
Visualization
26 November 2005
Felix Naumann, Humboldt-Universität zu Berlin
15


Information Integration
Web Service A
Web Service B
<pub>
<Titel> Federated Database Systems </Titel>
<Autoren>
<Autor> Amit Sheth </Autor>
<Autor> James Larson </Autor>
</Autoren>
</pub>
<publication>
<title> Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases </title>
<auth> Scheth & Larson </auth>
<year> 1990 </year>
</publication>
Integration
Identification
Fusion
Optimization
Visualization
26 November 2005
Felix Naumann, Humboldt-Universität zu Berlin
16

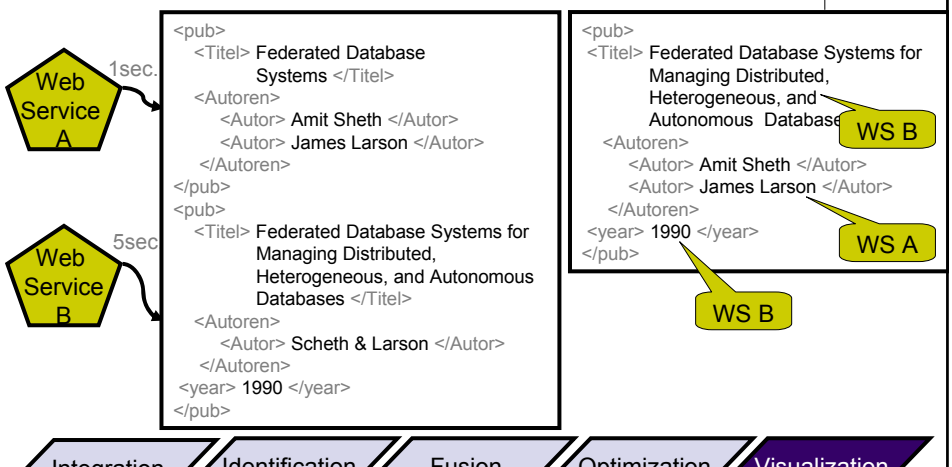

Information Integration
Web Service A
Web Service B
<pub>
<Titel> Federated Database Systems </Titel>
<Autoren>
<Autor> Amit Sheth </Autor>
<Autor> James Larson </Autor>
</Autoren>
</pub>
<publication>
<title> Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases </title>
<auth> Scheth & Larson </auth>
<year> 1990 </year>
</publication>
<pub>
<Titel> Federated Database Systems </Titel>
<Autoren>
<Autor> Amit Sheth </Autor>
<Autor> James Larson </Autor>
</Autoren>
</pub>
<pub>
<Titel> Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases </Titel>
<Autoren>
<Autor> Scheth & Larson </Autor>
</Autoren>
<year> 1990 </year>
</pub>
Integration
Identification
Fusion
Optimization
Visualization
26 November 2005
Felix Naumann, Humboldt-Universität zu Berlin
17

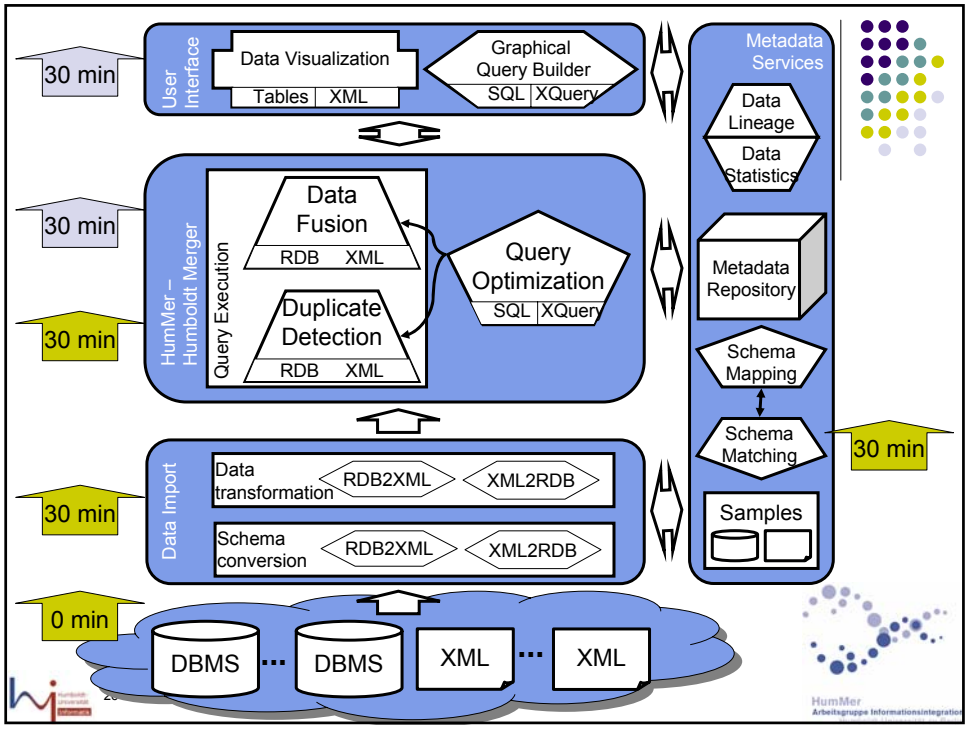

Information Integration
Web Service A
Web Service B
Integration
Identification
Fusion
Optimization
Visualization
26 November 2005
Felix Naumann, Humboldt-Universität zu Berlin
18

Information Integration



Information Integration





The HumboldtMerger Logo

HumMer
The Humboldt Merger







Overview

- Introduction and Motivation
- Information Integration in Three Steps
  1. Schema Mapping
     - And Schema Matching
  2. Duplicate Detection
  3. Data Fusion
- HumMer Architecture
- HumMer Demo

Schematic Heterogeneity

Men(Id, firstname, lastname)
Women(Id, firstname, lastname)

relation vs. attribute

Person(Id, firstname, lastname, male, female)

relation vs. value

attribute vs. value

Person(Id, firstname, lastname, sex)

Schematic Heterogeneity – Example

- Normalized vs. Denormalized
  - 1:1 associations between values are represented differently
    - In same tuple
    - Key-foreign key relationship

•ARTICLE •artPK •title •pages •AUTHOR •artFK •name	vs.
•PUBLICATION •pubID •title •date •author	

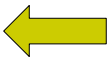
Schematic Heterogeneity – Example

- Nested vs. Flat
  - 1:n associations represented differently
    - As nested elements
    - Key-foreign key relationship
    - Redundant and denormalized

```
•ARTICLE                    •PUBLICATION
   •artPK                      •pubID
   •title        VS.           •title
   •pages                      •author
   •AUTHOR
      •name
```

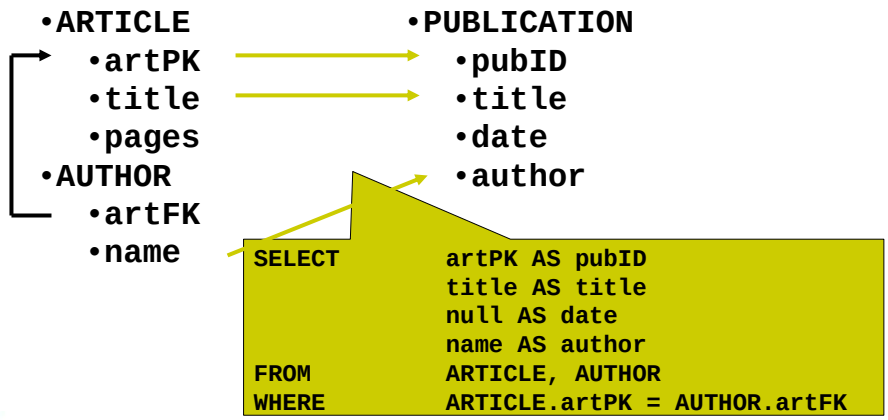

Schematic Heterogeneity - Solutions

- Two alternative problems
  1. Access multiple data sets uniformly
     - At schema level:
       - Schema languages (SchemaSQL, MSQL, FRAQL)
       - Schema mapping (Clio, Rondo, Tools)
     - At data level: virtual integration
  2. Integrate multiple data sets into a new database
     - At schema level: schema integration
     - At data level: materialized integration, ETL, DWH

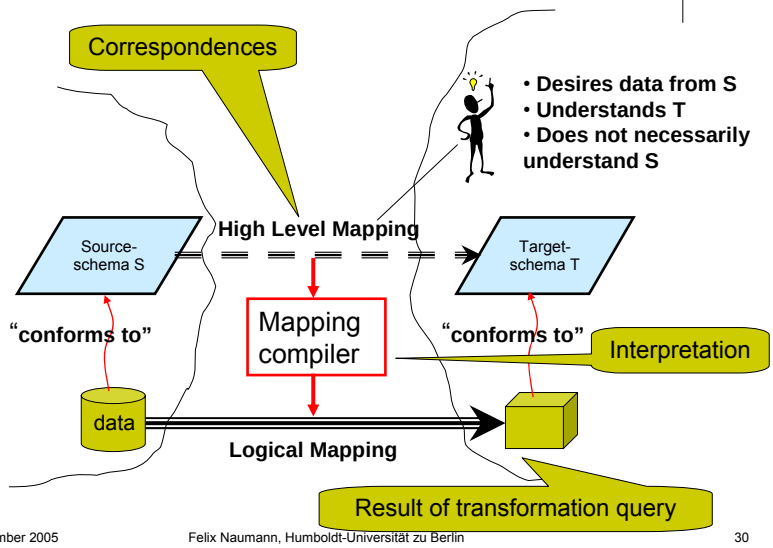


Schematic Heterogeneity – Solutions

- Schema Mapping



Schema Mapping in Context

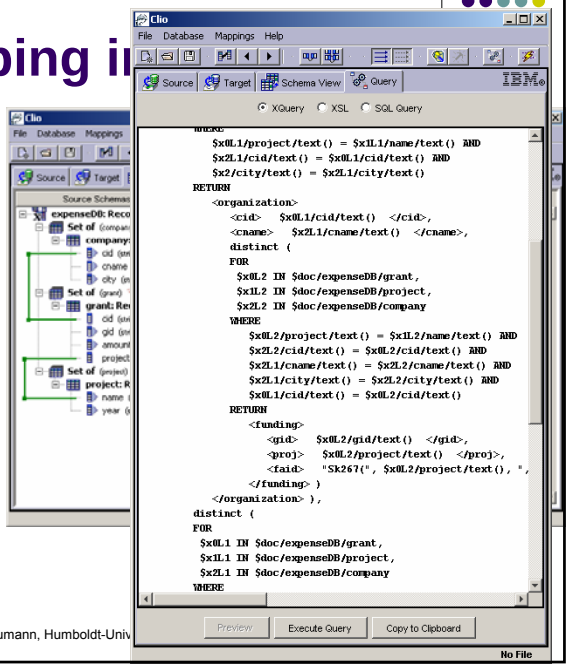


Motivation: Why is Schema Mapping useful?

- Data transformation between heterogeneous schemata
  - Old but omnipresent problem
  - Usually experts formulate complex query or programs
    - Time intensive
    - Expert for domain, for schemata and for query (language)
    - XML makes everything even more difficult
      - XML Schema, XQuery
- Idea: Automation
  - Given: Two schemata and a high-level mapping between them
  - Wanted: Query for data transformation
  - Later: Schema Matching = semi-automatic generation of high-level mapping

Schema Mapping i

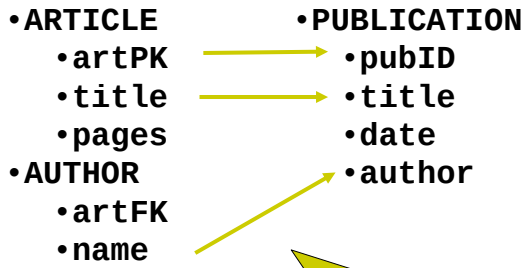
1. Schema Matching & Correspondences
2. Schema Mapping
3. Mapping Interpretation
4. Data Transformation



Motivation: Why is Schema Mapping Difficult?

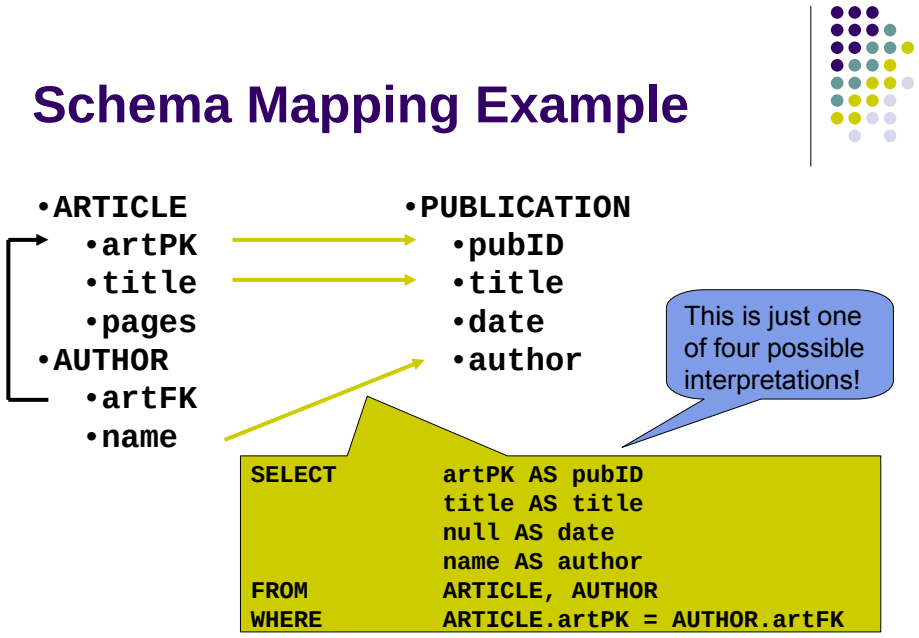
- Generation of „correct“ query under constraints of
  - source and target schemata,
  - the mappings,
  - and the users intention: semantics!
- Guarantee that transformed data adheres to target schema
  - Flat or nested
  - Integrity constraints
- Efficient data transformation
  - For materialized integration
  - For virtual integration

Schema Mapping Example

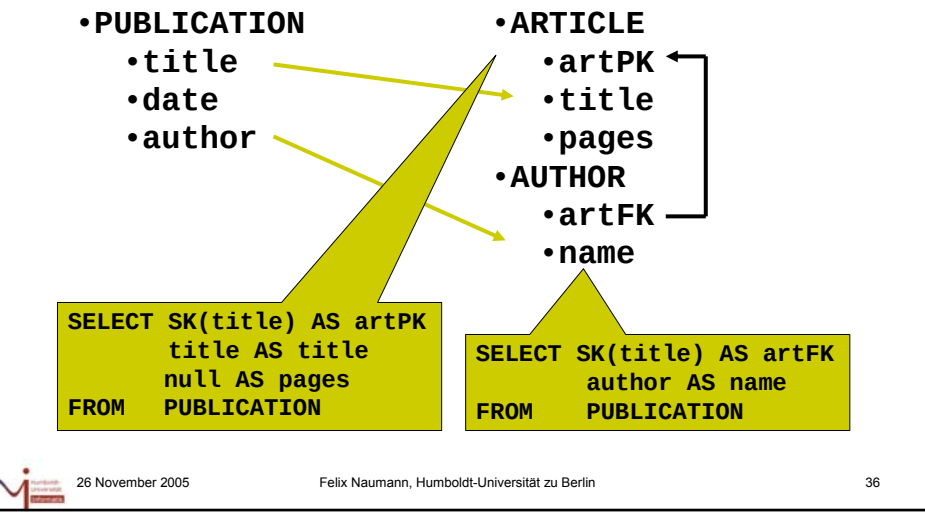



```
SELECT artPK AS pubID   UNION  SELECT null AS pubID
       title AS title                 null AS title
       null AS date                   null AS date
       null AS author                 name AS author
FROM   ARTICLE                FROM   AUTHOR
```

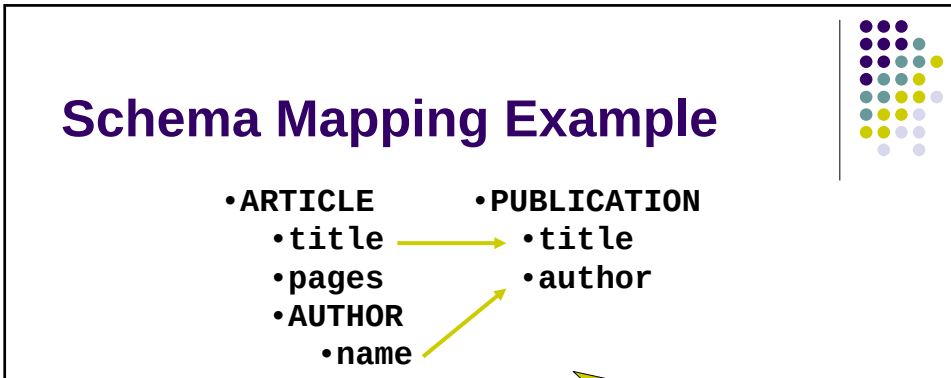
Schema Mapping Example



Schema Mapping Example

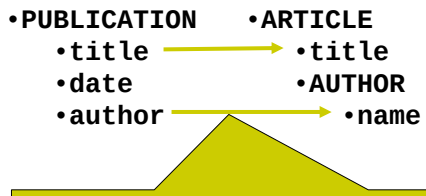


Schema Mapping Example




```
LET $doc0 := document("articls.xml") RETURN
<dblp> { distinct-values (
   FOR $x0 IN $doc0/authorDB/ARTICLE, $x1 IN $x0/AUTHOR
   RETURN
       <publication>
       <title>  {  $x0/title/text() }  </title>
       <author>  {  $x1/name/text() }  </author>
       </publication> )
   } </dblp>
```


Schema Mapping Example



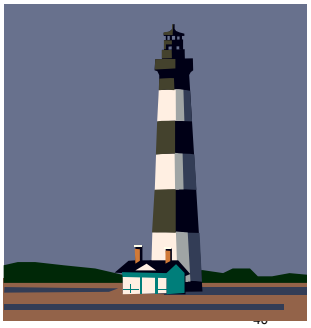

```
LET $doc0 := document("publication.xml")
RETURN
<articles> { distinct-values (
   FOR $x0 IN $doc0/dblp/publication  RETURN
      <ARTICLE>
         <title>  {  $x0/title/text() }  </title>
         { distinct-values (
         FOR $x0L1 IN $doc0/dblp/publications
         WHERE $x0/title/text() = $x0L1/title/text()
         RETURN
            <AUTHOR>
               <name>  {  $x0L1/author/text() }  </name>
            </AUTHOR> )}
      </ARTICLE> ) } </articles>
```

Schema Mapping Tools

- Research
  - Clio
  - Rondo
  - MOMIS
- Industrial
  - WSAD
  - BizTalk
  - MapForce
  - …
  - ETL
- Comparing tools
  - Data Models
  - Query Languages
  - Interface
  - Interpretation!

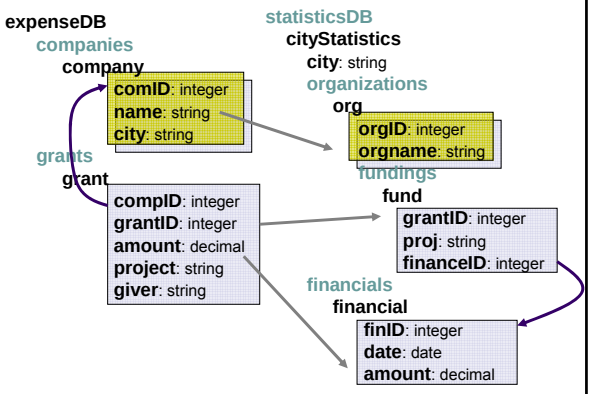
Overview

- Introduction and Motivation
- Information Integration in Three Steps
  1. Schema Mapping
     - Optional: Mapping Interpretation
     - And Schema Matching
  2. Duplicate Detection
  3. Data Fusion
- HumMer Architecture
- HumMer Demo



Mapping – Clio's Algorithm [FHP+02]

- Three steps
  1. Discovery of intra-schema associations
  2. Discovery of inter-Schema logical mappings
  3. Query generation



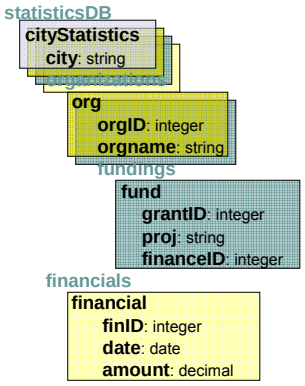
Discovery of Associations

- Step 1
  - Intra-schema associations between schema elements
  - Relational views contain maximal groups of associated elements
  - Each view represents a unique category of data of the data source
  - Independent of mapping (but restricted to mapped elements)

Source: [FHP+02]

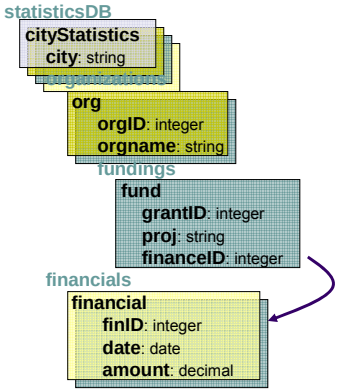
Discovery of Associations

- Start: All „primary“ paths
  - Associations in schema without integrity constraints
- For relational schemas
  - Each relation is one primary path
- For nested schemas
  - Attributes of one level
  - Attributes of nested levels

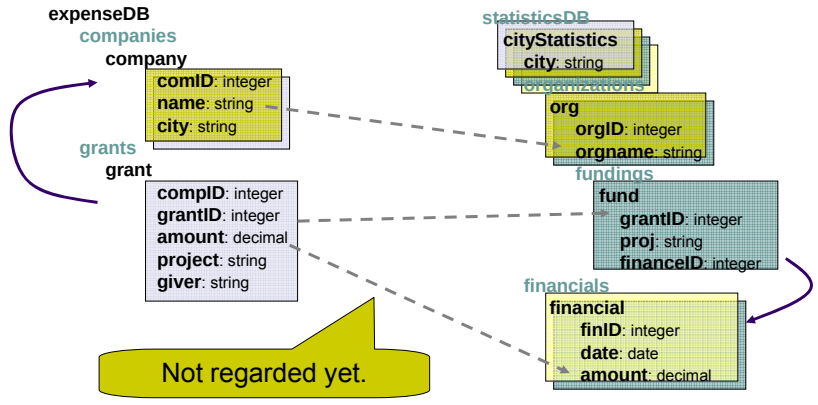


Discovery of Associations

- Include key/foreign key constraints
- Logical relation
  - Extend each primary path by chasing constraints

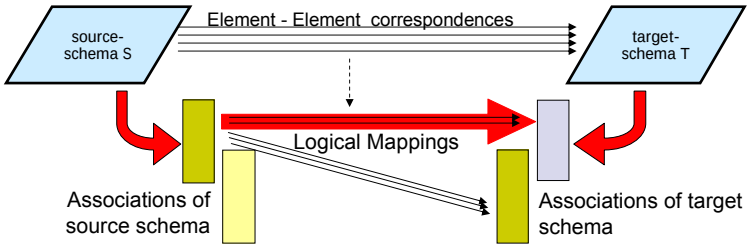


Discovery of Associations

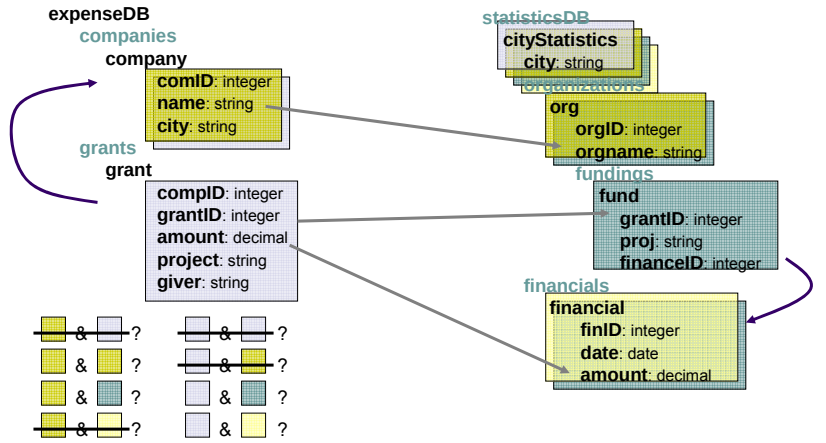


Discovery of Logical Mappings

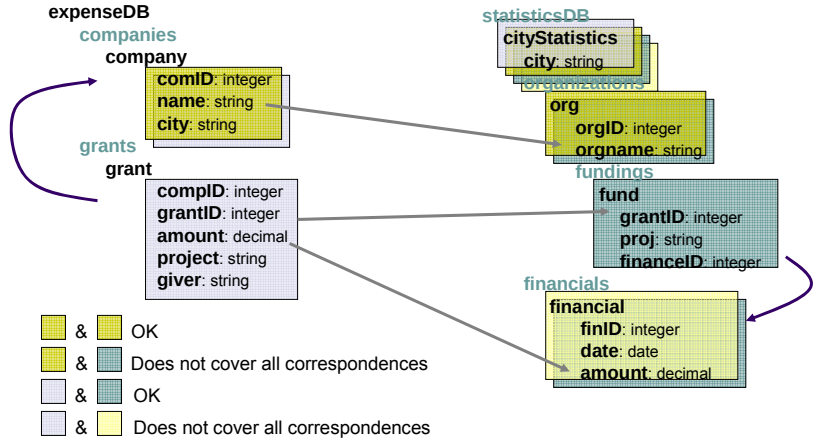
- Step 2
  - Discover logical mappings between source and target schema
  - Regard all combinations of associations of source schema and associations of target schema
    - Under consideration of all correspondences



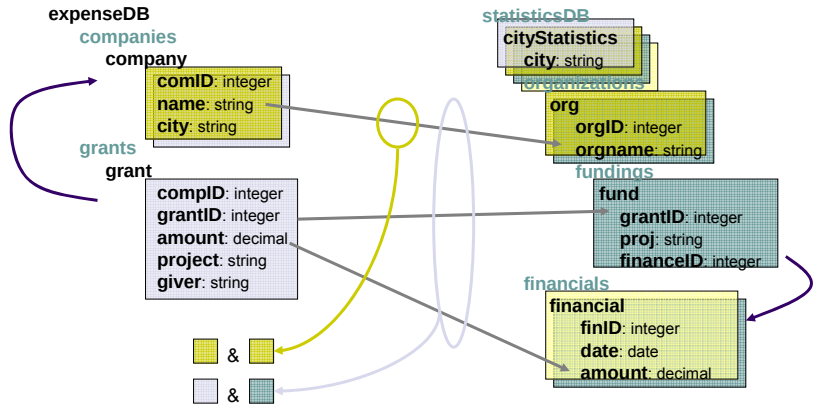
Discovery of Logical Mappings



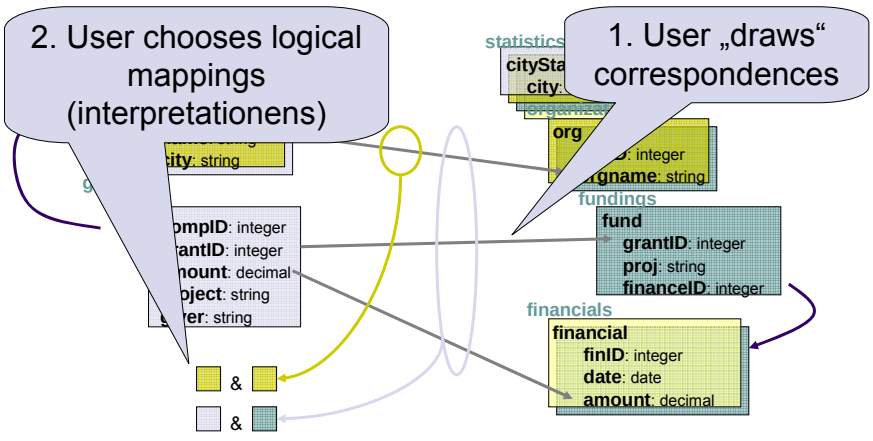
Discovery of Logical Mappings



Discovery of Logical Mappings



User-Input or Expert-Input

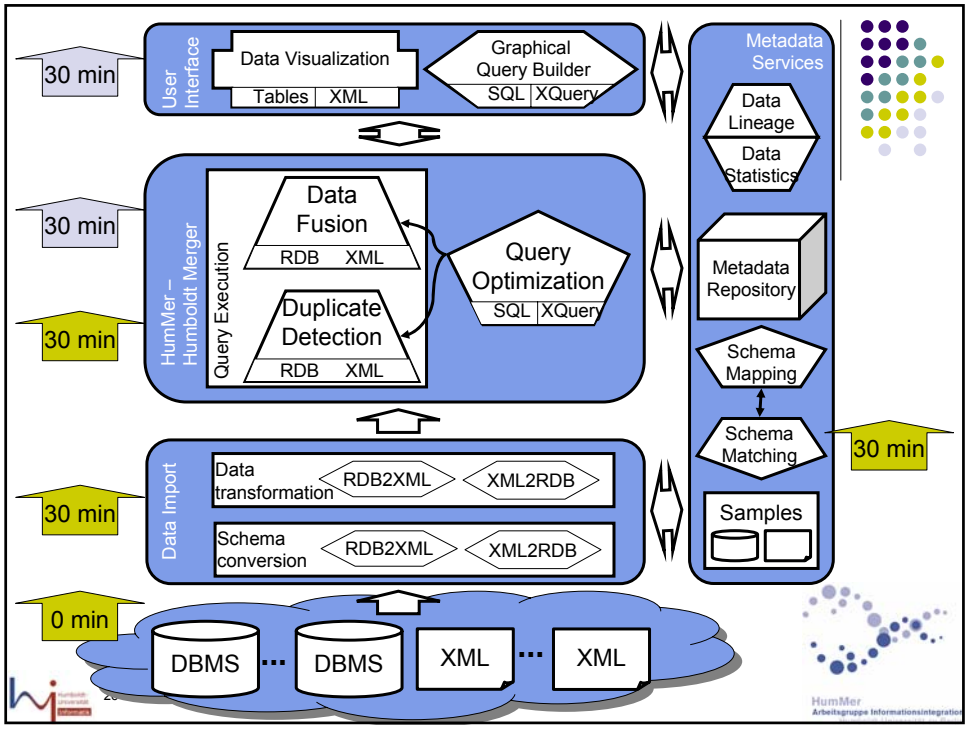


Final Step: Query Generation

- Implementationspecific
- Depending on data models
  - Relational → Relational: SQL
  - Relational → XML: SQLX (nesting and tagging of result)
  - XML → Relational: XQuery or XSLT (remove *tags*)
  - XML → XML: XQuery or XSLT
- In Clio
  - Generate proprietary rules
  - Then: Translate rules into query of specific language
- Not here…

Overview

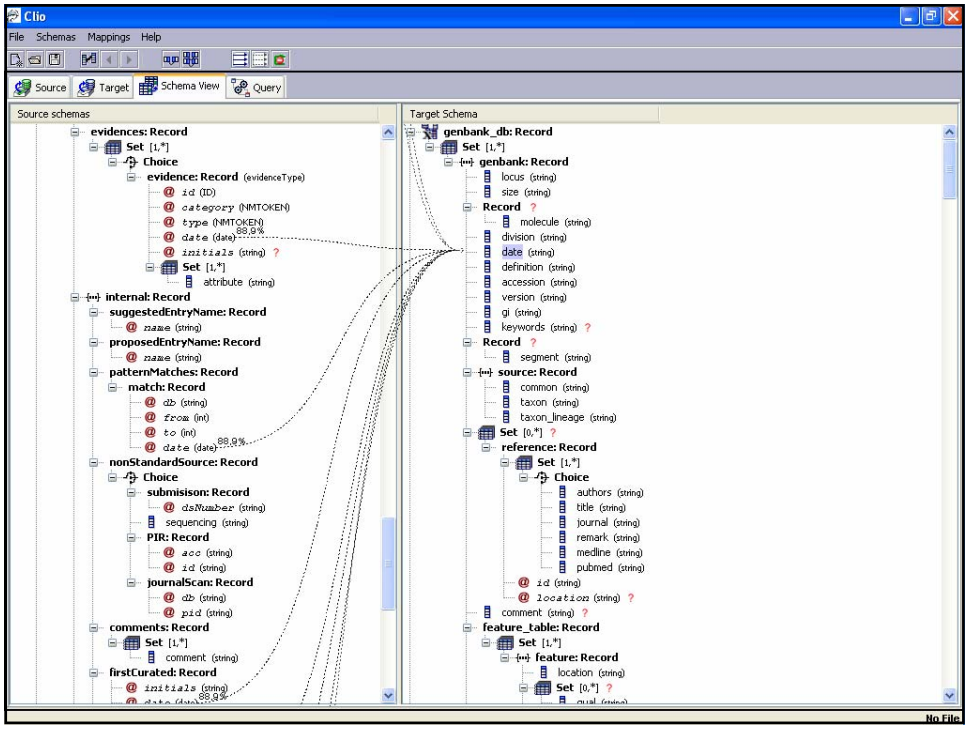
- Introduction and Motivation
- Information Integration in Three Steps
  1. Schema Mapping
     - And Schema Matching
  2. Duplicate Detection
  3. Data Fusion
- HumMer Architecture
- HumMer Demo



Schema Matching – Motivation

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

- > 100 tables, many attributes
- Deep Nesting
- Foreign keys
- XML Schema
- Unknown synonyms
- Unknown homonyms
- |attribute name| ≤ 8
- |table name| ≤ 8

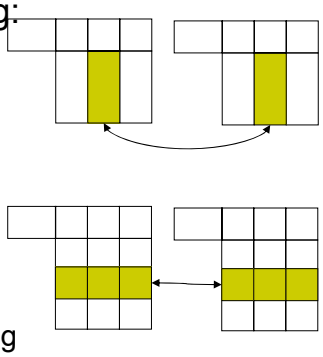


Schema Matching Algorithms

- Core algorithm of Schema Matching:
  - Compare attribute pairs from two schemata.
  - Choose most similar pairs using similarity measure.
  - Semi-automation: Conformation of suggestions by expert
- Similarity measures based on
  - Schema elements (label-based)
    - And other metadata, such as ontologies
  - Data (instance-based)
  - Structure (structure-based)
  - Combination, Meta-Matcher

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
  - [ICDE‘02]
- Our solution: Horizontal
  - Comparison of rows
  - = Duplicate detection (despite missing attribute correspondences)
  - [ICDE‘05]



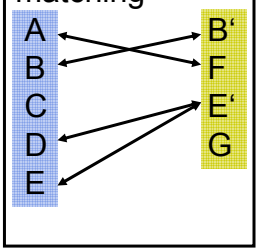
Duplicate-driven Schema Matching

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

? ?

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

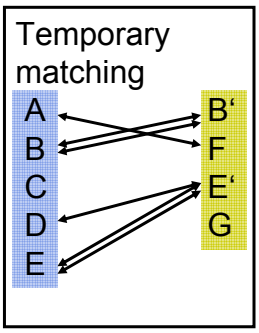
Temporary matching



Duplicate-driven Schema Matching

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



- Assumptions
  - There is data in both DBs.
  - There are (at least a few) duplicates in both DBs.
  - Equal or similar values reflect same semantics of attributes.

Duplicate-driven Schema Matching

1. Duplicate detection
   - Goal: Find the top-k duplicates.
   - Problems
     - Correspondences unknown
     - Possibly small intensional overlap
2. Schema Matching
   - Goal: Derive attribute correspondences from attribute values.
   - Problems
     - Attribute values only similar, not equal
     - Synonyms and homonyms in values

Duplicate Detection in Unaligned Tables

- Cosine measure with TFIDF weights
  - Tuple as vector of term weights
  - Term weights is high if term appears often in tuple (TF) and it appears in only few tuples (IDF).
  - Tuple similarity is the cosine of the angle of both (normalized) vectors.

$$w'(r,t) = \log(tf_{r,t} + 1) \cdot \log(\frac{N}{df_t} + 1)$$

$$tsim(r,s) = \sum_{t \in r \cap s} w(r,t) \cdot w(s,t)$$

123	Max Michel	max@michel.com	601- 4839204	601- 4839204

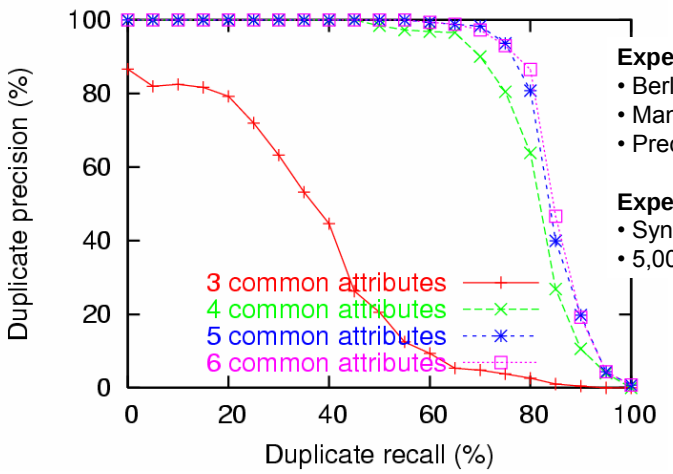
1	Max Michel	601- 4839204	max@michel.com

„1“	0	0.06
„123“	0.21	0
„Max“	0.3	0.28
„601“	0.14	0.16
:	:	:

Duplicate Detection

- Algorithm
  - Difference to regular duplicate detection:
    - Only top K duplicates necessary ⇒ no threshold
  - Comparisons for top 10 duplicates in 5,000 tuples
    - Naive: 25,000,000
    - Inverted index: 340.333
    - WHIRL (A*): 8.833
- Synthetic person data
  - Different intensional overlaps
  - Different degree of duplication
- Berlin real estate offers

Duplicate detection

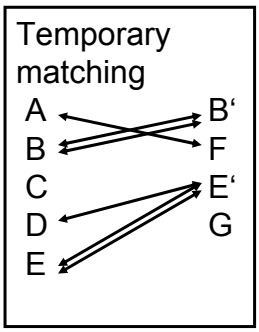


Experiment 1 (no Fig.):
- Berlin real estate
- Many duplicates
- Precision for top 10: 100%

Experiment 2:
- Synthetic person data
- 5,000 tuples, 50 duplicates

Schema Matching

- Given the top K duplicates.
- Find a (global) matching
  - I.e., each attribute has 0 or 1 correspondence.



Formalized by similarity matrix

Average SoftTFIDF

	A	B	C	D	E
B‘	0.22	0.92	0.07	0	0
F	0.60	0.60	0.07	0	0
E‘	0	0	0	0.58	0.64
G	0	0.07	0	0.07	0.02

2. Schema Matching – Similarity Measure

- Fieldwise Comparison
  - Compare attribute values of record pairs. Similarity measure: Edit-distance vs. SoftTFIDF
- Edit distance
  - Minimal number of edit operations (substitute, insert, delete); several variations exist
- SoftTFIDF
  - 'Soft' variation of TFIDF that also considers similar terms

$$tfidf(r,s) = \sum_{t \in r \cap s} w(r,t) \cdot w(s,t)$$

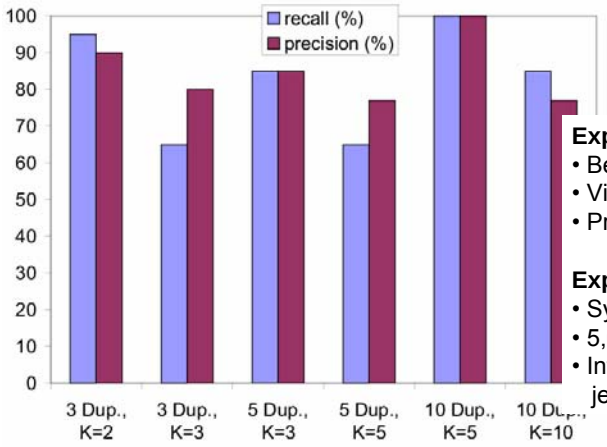
$$soft - tfidf(r,s) = \sum_{t \in CLOSURE(\theta, r, s)} w(r,t) \cdot w(s,t') \cdot sim(t,t')$$

Graph Matching

- Given:
  - Similarity matrix
  - = weighted bipartite graph
- Find:
  - Maximal weight matching
  - Alternative: Matching with *stable marriage* property
- Outlook
  - Produce mappings and not just correspondences

	A	B	C	D	E
B'	0.22	0.92	0.07	0	0
F	0.60	0.60	0.07	0	0
E'	0	0	0	0.58	0.64
G	0	0.07	0	0.07	0.02

Schema Matching

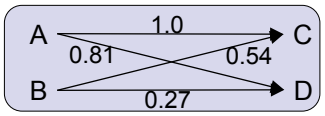


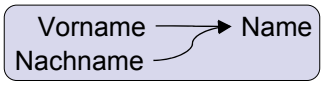
Experiment 1 (ohne Abb.):
- Berliner Wohnungsmarkt
- Viele Duplikate
- Precision und recall immer 100%

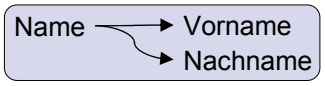
Experiment 2:
- Synthetische Personendaten
- 5,000 Tupel
- Intensionale Überlappung: jeweils 4 von 8

Schema Matching – Extensions

- Globales Matching
  - Matche Tabellen und Schemata, nicht nur Attribute
  - Stable Marriage bzw. Maximum Weighted Matching
- n:1 und 1:n Matches
  - Viele Kombinationsmöglichkeiten
  - Viele Funktionen denkbar
- Matching in komplexen Schemata
  - Ziel: Finde Mapping, nicht Korrespondenzen







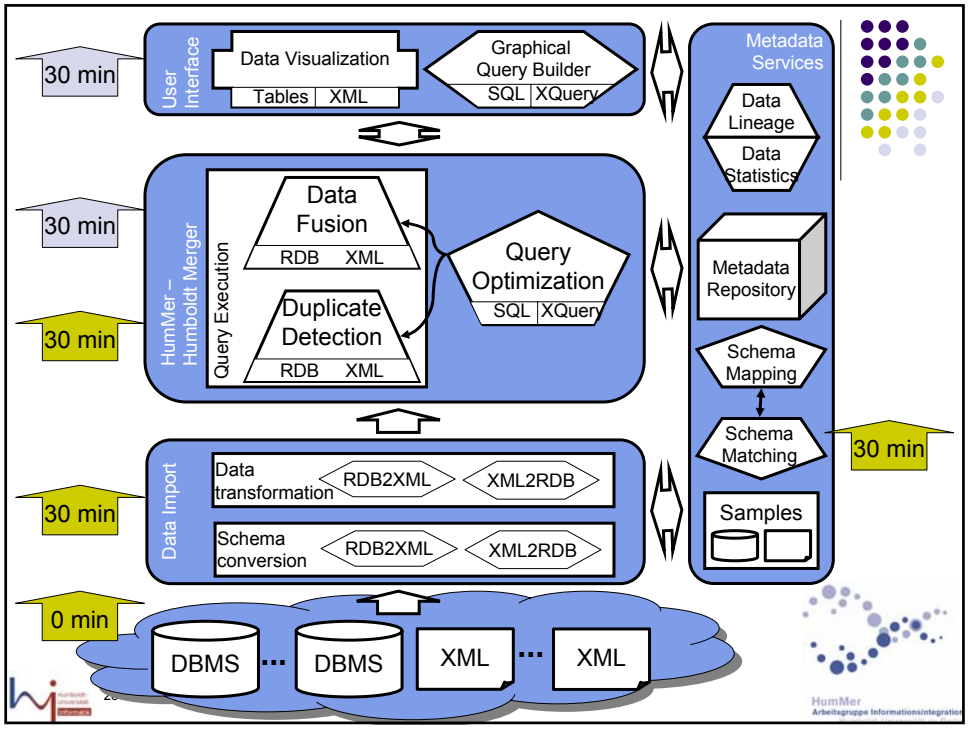
Coffee break?



- After the break
  - Duplicate Detection
  - Data Fusion
  - Demo

Overview

- Introduction and Motivation
- Information Integration in Three Steps
  1. Schema Mapping
     - And Schema Matching
  2. Duplicate Detection
  3. Data Fusion
- HumMer Architecture
- HumMer Demo



Duplicate detection

- Detection of multiple same or similar representations of a real-world object
- Also
  - Object Identification
  - Data Cleansing
  - Record Linkage
  - Reference Reconciliation
  - …
- Domain-specific algorithms
  - Address data („mailman similarity“)
  - Genome data
- Sometimes there are IDs
  - ISBN, SSN, URL

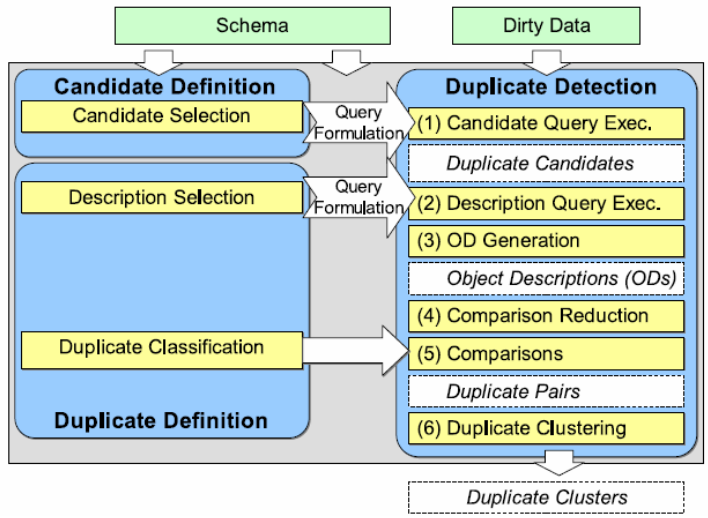
Duplicate detection

- Definition of an Object
  - Tuple
  - XML element

- Similarity measure
  - Domain independent
    - Edit-distance
    - TFIDF-based
    - Tree-distance
  - Domain dependent
    - Rules

- Algorithm
  - Acceleration of similarity comparisons
    - Filter
  - Avoidance of similarity comparisons
    - Partitioning

The General Framework

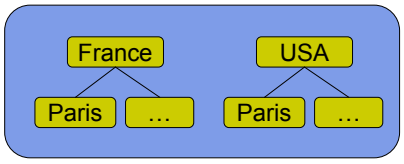


Definition of an XML object

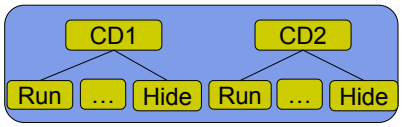
- Compare <author>
  - Include sub-elements (<publication>)?
  - How deep?
- Compare <publication>
  - Only in the scope of the parent element?
  - Schema, or Data?
- In short: What is an object?

Top-down, bottom-up, and through the middle

Top-down [SIGMOD'05]
- Only compare if parents are equal or duplicates
- Improved efficiency



Bottom-up [EDBT'06]
- Similar Children ⇒ Duplicate
- Improved effectiveness



Through the middle [ICDE'06]
- Begin with most promising pairs
- Best of both worlds

Further techniques
- Object filter
- Edit distance filter
- Transitivity

DELPHI – Data Warehouse Duplicates

ID	country
1	USA
2	United States
3	Unitd States

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

String similarity
→ 2 ≈ 3

Common children
→ 1 ≈ 2 ≈ 3

Similarity measure

- Token-Set *TS(e)*
  - Data of XML element *e* as tokens
- Children-Set *CS(e)*
  - Data of children elements of *e* as strings
- *D(e) = TS(e) ∪ CS(e)*

$$softdf(s) = \left|\{e \mid d_{edit}(s,s') \le t_{edit}, s' \in e\}\right|$$

Soft document frequency

$$softidf_D(S) = \sum_{s \in S} \log \frac{|D|}{softdf_D(s)}$$

Soft inverse doc. frequency

$$sim(e_1,e_2) = \frac{softidf(D(e_1) \cap D(e_2))}{softidf((D(e_1) \cup D(e_2)) \setminus D(e_1) \cap D(e_2))}$$

Importance of common tokens

Importance of non-common tokens

Algorithm

1. Create data structure
   - XQueries to extract relevant elements
     - Elements of one type at one level
     - plus descendants
2. Acceleration of similarity comparisons
   - Edit-distance filter
3. Avoidance of similarity comparisons
   - Element-similarity-filter
   - Connected components
4. Similarity comparisons
   - Among remaining elements
   - $sim(e_1,e_2) \geq t_{dup}$

Supported by graph-based data structure:

Similarity of tokens are edges between token-nodes

Similarity of elements are edges between element-nodes

Edit-distance Filter

- Goal:
  - Accelerate calculation of $d_{edit}(s_1,s_2) \leq t_{edit}$
  - $d_{edit}(s_1,s_2)$: Minimum number of edit-operations $s_1 \leftrightarrow s_2$
1. Length filter
   - $|l(s_1) - l(s_2)| \leq d_{edit}(s_1,s_2)$
2. Triangular inequation property
   - $|d_{edit}(s_1,s_2) + d_{edit}(s_2,s_3)| \geq d_{edit}(s_1,s_3)$
   - $|d_{edit}(s_1,s_2) - d_{edit}(s_2,s_3)| \leq d_{edit}(s_1,s_3)$
3. Bag filter
   - $d_{bag}(s_1,s_2) = \max\{|S_1 - S_2|, |S_2 - S_1|\} \leq d_{edit}(s_1,s_2)$

Element-similarity Filter

- Goal
  - Avoid calculation of $sim(e_1,e_2) \geq t_{dup}$
- Element-similarity filter

$$sim(e_1,e_2) = \frac{softidf\,(D(e_1) \cap D(e_2))}{softidf\,\big((D(e_1) \cup D(e_2)) \setminus D(e_1) \cap D(e_2)\big)}$$

$$\leq \frac{softidf\,(D(e_1) \cap (Dok - D(e_1)))}{softidf\,\big(D(e_1) \setminus \big(D(e_1) \cap (Dok - D(e_1))\big)\big)}$$

- Intuition
  - Similarity to an element ≤ similarity to all elements
- Question
  - What do I save?

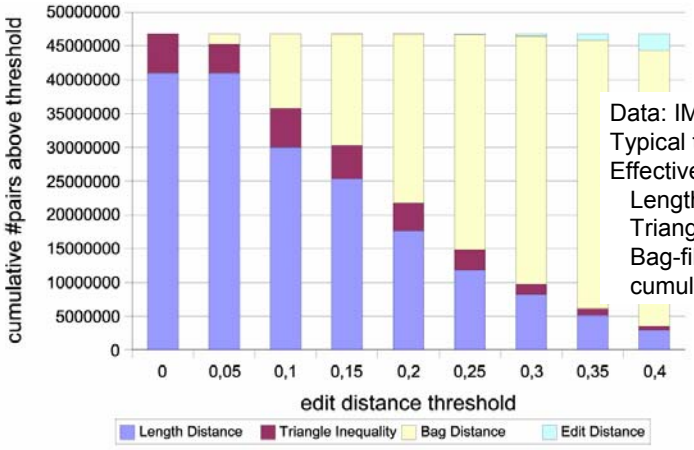
Similarity comparisons

1. $sim(e_1,e_2) \leq t_{dup}$ for remaining element pairs
2. Transitive closure of all duplicates
   - Duplicate-cluster
3. Repeat for next level of hierarchy
   - But: Only within clusters

Experiments

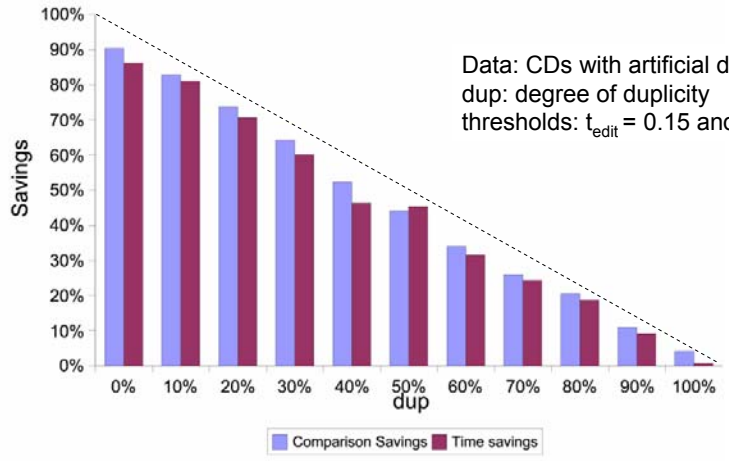
- Data
  - IMDB: 54.000 persons
  - freedb.org: 10.000 CDs
  - mondialDB: 260 countries
    - Synthetic duplicates
- Effectiveness of edit-distance filter
- Effectiveness of element-similarity filter
- Precision & recall of duplicate detection

Edit-distance Filter



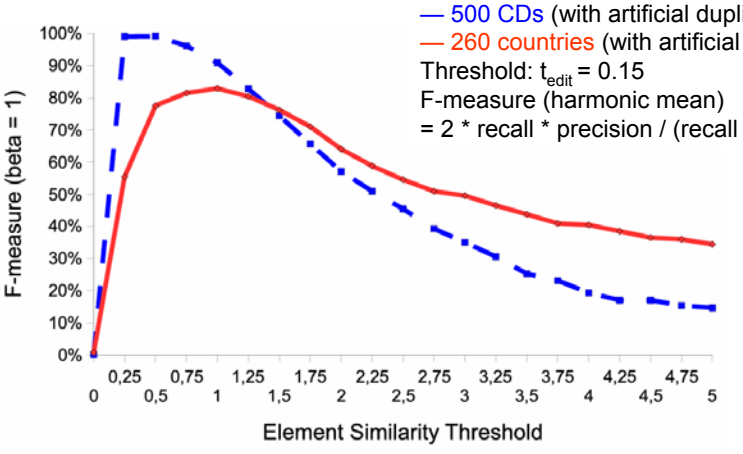
Data: IMDB Persons
Typical threshold: t_{edit}0.15
Effectiveness at t_{edit}=0.15:
Length-filter: 83%
Triangle: 23%
Bag-filter: 97%
cumulated: ca. 99.6 %

Element-similarity Filter



Data: CDs with artificial duplicates
dup: degree of duplicity
thresholds: t_{edit} = 0.15 and t_{dup} = 1

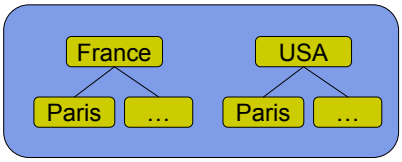
Precision & Recall



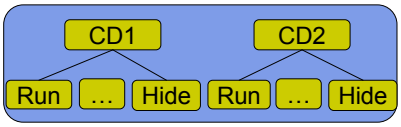
— 500 CDs (with artificial duplicates)
— 260 countries (with artificial duplicates)
Threshold: t_{edit} = 0.15
F-measure (harmonic mean)
= 2 * recall * precision / (recall + precision)

Top-down, bottom-up, and through the middle

Top-down [SIGMOD'05]
- Only compare if parents are equal or duplicates
- Improved efficiency



Bottom-up [EDBT'06]
- Similar Children ⇒ Duplicate
- Improved effectiveness



Through the middle [ICDE'06]
- Begin with most promising pairs
- Best of both worlds

Further techniques
- Object filter
- Edit distance filter
- Transitivity

The Bottom-Up Approach

- Idea: Apapt Sorted-Neighborhood algorithm
  - Construct sorted neigborhood for leaves.
  - Construct sorted neighborhood for intermediate nodes with many similar children
- Implemented on top of relational DBMS
- Not as efficient as top down.
- But more effective
  - More comparisons
  - But not too many (windowing!)

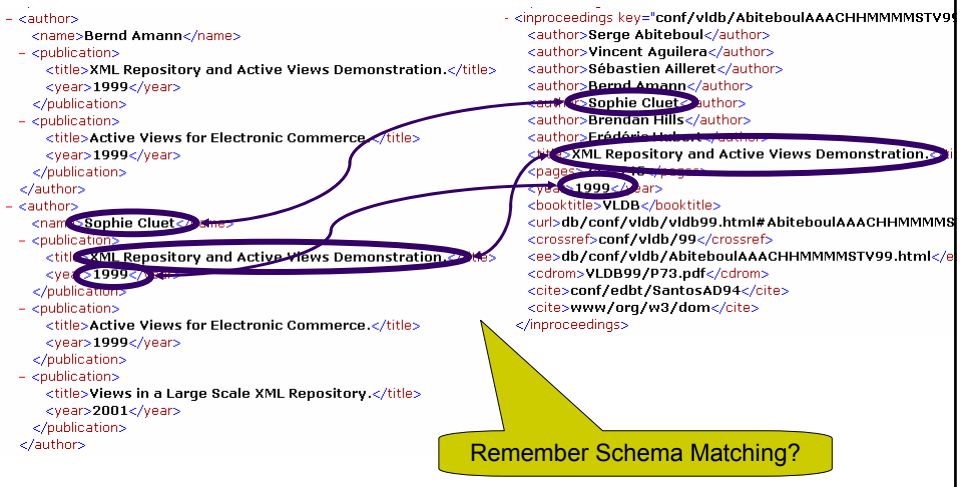
A Through-the-Middle Approach

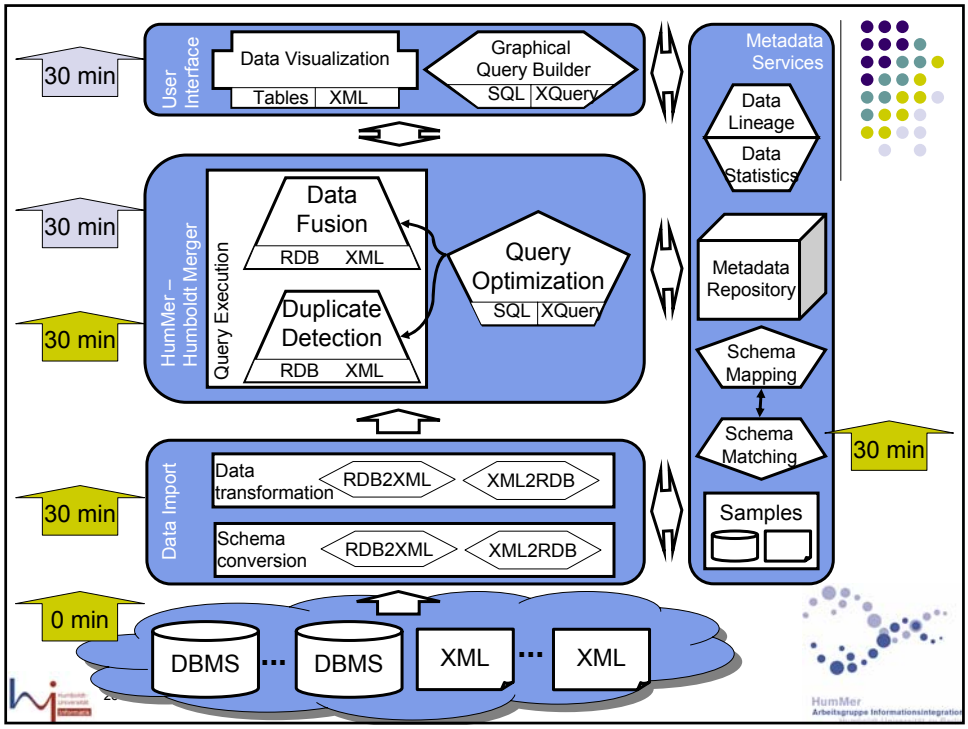
- Intuition:
  - Any detected duplicate pair affects neighbors (parents and children)
  - Rank all pairs of (comparable) objects by their degree of influence on neighbors
  - Compare objects with low influence first
    - Either: Reduce number of recomparisons
    - Or: Improve effectiveness of recomparisons are not allowed.

Outlook – Duplicate detection

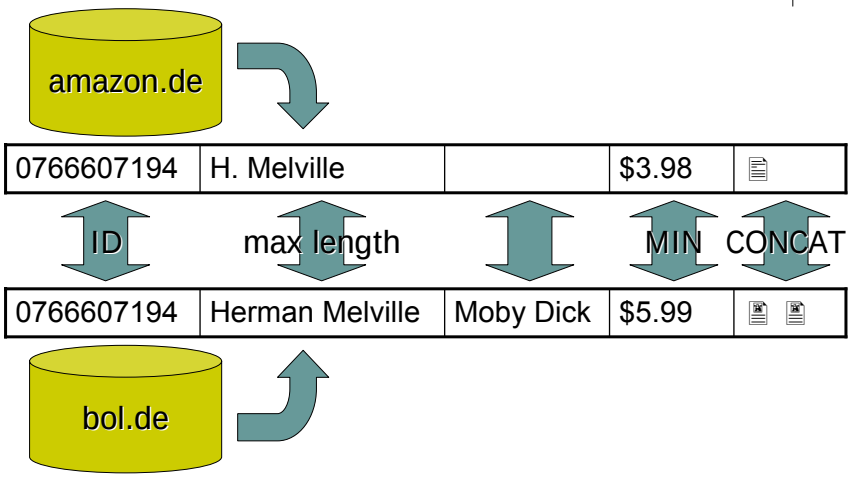
- Definition of Objects
  - Declarative specification
  - Graphical tool
- Similarity measures
  - Use schema information
- Algorithm
  - Scalability (RAM, CPU)
  - XML DBMS (Tamino)
- Experiments
  - Movie data from five sources
  - Graphical tool for manual duplicate detection

Relaxing the object definition



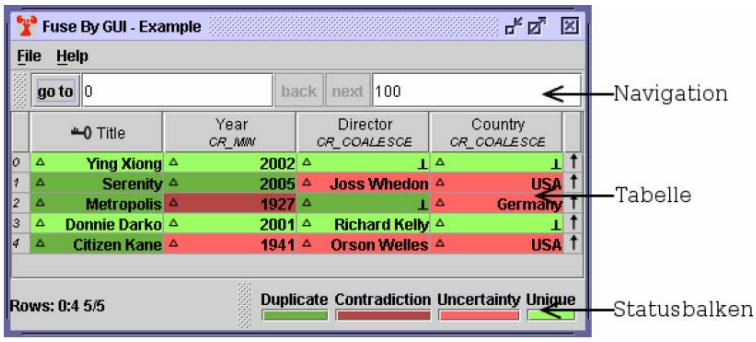


Data Fusion



Visualization of Integrated Data

- Why Provenance & Where Provenance
- Conflicts en detail and in overview (zoom out)



Thank You for Listening

naumann@informatik.hu-berlin.de

Information Integration in Three Steps

1. Schema Mapping
   1.1 Schema Matching
2. Duplicate Detection
3. Data Fusion

References

- [SIGMOD-Record'04] BioFast: Challenges in Exploring Linked Life Science Sources. J. Bleiholder, Z. Lacroix, H. Murthy, F. Naumann, L. Raschid, and M-E. Vidal: SIGMOD Record 33(2), June 2004
- [ICDE'05] Schema Matching Using Duplicates. Alexander Bilke, Felix Naumann: Proceedings of the International Conference on Data Engineering (ICDE 05) Tokyo, Japan.
- [WebDB'05] A Data Model and Query Language to Explore Enhanced Links and Paths in Life Science Sources.
  George Mihaila, Felix Naumann, Louiqa Raschid, Maria-Esther Vidal. In SIGMOD WebDB 2005, Baltimore, MD.
- [ICDE'02] Attribute Classification Using Feature Analysis. Felix Naumann, Ching-Tien Ho, Xuqing Tian, Laura M. Haas, Nimrod Megiddo: ICDE 2002: 271
- [VLDB'05] Automatic Data Fusion with HumMer (demo). Alexander Bilke, Jens Bleiholder, Christoph Böhm, Karsten Draba, Felix Naumann, Melanie Weis. Proceedings of the Int. Conf. on Very Large Databases (VLDB) 2005, Troindheim, Norway
- [SIGMOD'05] DogmatiX Tracks down Duplicates in XML. Melanie Weis and Felix Naumann, (SIGMOD 2005), Baltimore, MD
- [ADBIS'05] Declarative Data Fusion - Syntax, Semantics, and Implementation. Jens Bleiholder and Felix Naumann, (ADBIS 2005), Tallin, Estonia.
- [CIDR'05] (Almost) Hands-Off Information Integration for the Life Sciences. Ulf Leser, Felix Naumann. Proceedings of the Conference in Innovative Database Research (CIDR 2005), Asilomar, CA.
- [BTW'05] Self-Extending Peer Data Management. Ralf Heese, Sven Herschel, Felix Naumann, Armin Roth (BTW 2005), Karlsruhe, Germany.
- [FHP+02] Ron Fagin, Mauricio Hernandez, Lucian Popa, Renee Miller, and Yannis Velegrakis, Translating Web Data, VLDB 2002, Hong Kong, China.
- [EDBT'06] XML Duplicate Detection Using Sorted Neigborhoods. Sven Puhlmann, Melanie Weis, Felix Naumann: *Proceedings of the International Conference on Extending Database Technology (EDBT)* 2006, Munich, Germany.
- [ICDE'06] Detecting Duplicates in Complex XML Data. Melanie Weis, Felix Naumann. *Proceedings of the International Conference on Data Engineering (ICDE) 2006,* Atlanta, GA. poster