Data Integration in Two Steps and From Above

Palo Alto, February 3, 2009
Felix Naumann
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
“Data Fusion in Three Steps”
DE Bulletin, 2006
“Data Fusion”
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

Web Service A

Web Service B

Integration Identification Fusion Optimization Visualization

<Federed Database Systems>

<Autor> Amit Sheth </Autor>

<Autor> James Larson </Autor>

</Autoren>

</pub>

<Titel> Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases </Titel>

<auth> Scheth & Larson </auth>

<year> 1990 </year>

</publication>
Information Integration

Web Service A

Web Service B

Integration Identification Fusion Optimization Visualization

Felix Naumann | Information Systems | Hasso Plattner Institut
Information Integration

Fusion Optimization Visualization Identification

Integration Identification Fusion Optimization Visualization

Schema Integration Schema Mapping

Web Service A

Web Service B

Federated Database Systems

Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases
Information Integration

Web Service A

Web Service B

<publication>
<title> Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases </title>
<auth> Scheth & Larson </auth>
<year> 1990 </year>
</publication>

Fusion

Optimization

Visualization

Integration

Identification

Federated Database Systems

Amit Sheth

James Larson

Scheth & Larson

1990
Information Integration

Federated Database Systems
Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases

Integration Identification Fusion Optimization Visualization

Felix Naumann | Information Systems | Hasso Plattner Institut
Information Integration

Web Service A

1sec.

Web Service B

5sec.

Integration | Identification | Fusion | Optimization | Visualization

Federated Database Systems</Titulo>

<Autoren>
  <Autor>Amit Sheth</Autor>
  <Autor>James Larson</Autor>
</Autoren>

<year>1990</year>

Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases</Titulo>

<Autoren>
  <Autor>Scheth & Larson</Autor>
</Autoren>

<year>1990</year>

Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases

<Autoren>
  <Autor>Amit Sheth</Autor>
  <Autor>James Larson</Autor>
</Autoren>

<year>1990</year>
Completeness and Conciseness

Intensional completeness

Extensional completeness

Duplicate Detection

Common objects

Conflicts!

Common attributes

Source 1

Source 2

Data Fusion

Schema Matching

Felix Naumann | Information Systems | Hasso Plattner Institut
Overview

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

- ARTICLE
  - artPK
  - title
  - pages

- AUTHOR
  - artFK
  - name

- PUBLICATION
  - pubID
  - title
  - date
  - author

SELECT artPK AS pubID
    title AS title
    null AS date
    null AS author
FROM ARTICLE
UNION
SELECT null AS pubID
    null AS title
    null AS date
    name AS author
FROM AUTHOR
Schematic heterogeneity – solutions

Further interpretations?

SELECT artPK AS pubID, title AS title, null AS date, name AS author
FROM ARTICLE, AUTHOR
WHERE ARTICLE.artPK = AUTHOR.artFK
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 Classification [RB01]

Schema Matching Approaches

Individual Approaches
- Schema-based
  - Linguistic
  - Constraint-based

Instance-based
- Linguistic
- Constraint-based

Combined Approaches
- Hybrid
- Manual
- Automatic

Hybrid
- Constraint-based
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

### Temporary matching

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>Michel</td>
<td>m</td>
<td>601- 4839204</td>
<td>601- 4839204</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B'</th>
<th>F</th>
<th>E'</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michel</td>
<td>maxm</td>
<td>601- 4839204</td>
<td>UNIX</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
### DUMAS Matcher

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>Michel</td>
<td>m</td>
<td>601- 4839204</td>
<td>601- 4839204</td>
</tr>
<tr>
<td>Sam</td>
<td>Adams</td>
<td>m</td>
<td>541- 8127100</td>
<td>541- 8121164</td>
</tr>
</tbody>
</table>

**Temporary matching**

<table>
<thead>
<tr>
<th>B'</th>
<th>F</th>
<th>E'</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michel</td>
<td>maxm</td>
<td>601- 4839204</td>
<td>UNIX</td>
</tr>
<tr>
<td>Adams</td>
<td>beer</td>
<td>541- 8127164</td>
<td>WinXP</td>
</tr>
</tbody>
</table>
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

- Tooling: User in the loop!

**n:1 Matching**
- First name $\rightarrow$ concat() $\rightarrow$ Name
- Surname

**1:n Matching**
- Name $\rightarrow$ extract() $\rightarrow$ First name
- extract() $\rightarrow$ Surname

**m:n matching**
- Name $\rightarrow$ extract() $\rightarrow$ concat() $\rightarrow$ First name
- Title $\rightarrow$ extract() $\rightarrow$ Last name
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

Duplicate Detection

$R_1 \times R_2$

Similarity measure

Algorithm

$\text{sim} > \theta$

$\text{sim} < \theta$

Duplicate

Non-duplicate

Felix Naumann | Information Systems | Hasso Plattner Institut
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

<table>
<thead>
<tr>
<th>Customer</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMW</td>
<td>20.000</td>
</tr>
<tr>
<td>BaMoWe</td>
<td>5.000.000</td>
</tr>
<tr>
<td>Bayerische Motorenwerke</td>
<td>300.000</td>
</tr>
</tbody>
</table>

...
Ironically, “Duplicate Detection” has many Duplicates

Household matching
Mixed and split citation problem
Match
Deduplication
Entity resolution
Identity uncertainty
Hardening soft databases

Doubles
Duplicate detection
Record linkage
Object identification
Object consolidation
Entity clustering
Approximate match
Reference reconciliation
Merge/purge
Reference matching

Householding

Felix Naumann | Information Systems | Hasso Plattner Institut
Duplicate Detection – Research

Duplicate Detection

Identity
- Relational
- XML
- DWH
  - Domain-independent
  - Domain-dependent

Similarity measure
- Partitioning
- Filters
  - Edit-based
  - Token-based
  - Rules
  - Data types
  - Relationship-aware

Algorithm
- Relation-ships
- Clustering / Learning
- Incremental / Search

Evaluation
- Precision / Recall
- Efficiency
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 | City | Country
--- | --- | ---
1 | New York | 1
2 | Los Angeles | 1
3 | New York | 2
4 | Los Angeles | 2
5 | New York | 3
6 | Los Angeles | 3

### ID | Street
--- | ---
1 | First Ave
2 | High St.
3 | Broadway
4 | Embarcadero
5 | Broadway
6 | Second St.
7 | P St.
8 | Pennsylvania
9 | Sunset Blvd
10 | Santa Monica
11 | Ocean Ave
Relationship-aware Similarity Measures – Evaluation

![Graph showing precision vs recall with and without actors]

- Red line: without actors
- Green line: with actors

Felix Naumann | Information Systems | Hasso Plattner Institut
### Iterative RADD

<table>
<thead>
<tr>
<th>Top-down</th>
<th>Bottom-up</th>
<th>From-the-middle</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SIGMOD'05]</td>
<td>[EDBT'06]</td>
<td>[ICDE'06]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Effectiveness</th>
<th>Efficiency</th>
<th>Effectiveness</th>
<th>Efficiency</th>
<th>Effectiveness</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>★★★</td>
<td>★★★</td>
<td>★★★</td>
<td>★★★</td>
<td>★★★★</td>
<td>★★★★</td>
</tr>
</tbody>
</table>

Further techniques: Object filter; Edit distance filter; Transitivity

Felix Naumann | Information Systems | Hasso Plattner Institut
Scalability of the generalized algorithm

Scalability

\( (a_1, a_1') \)

\( (m_1, m_1') \)

\( in-database \)

\( PQ \)

Retrieve

Classify

Update

duplicates

\( (a_1, a_1') \)

\( in-database \)

\( PQ \)

Retrieve

Classify

Update

duplicates

\( (a_1, a_1') \)

\( in-memory \)

\( PQ \)

Retrieve

Classify

Update

duplicates

\( (a_1, a_1') \)
## Scalability – Comparison

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

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 ...

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

Source: wikipedia.de

Felix Naumann | Information Systems | Hasso Plattner Institut
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
“Proper” Data Fusion

Source 1(A,B,C) →

\[
\begin{array}{c}
\text{a, b, c} \\
\text{a, b, d}
\end{array}
\] →

\[
\begin{array}{c}
\text{a, b, c, -} \\
\text{a, b, -, d}
\end{array}
\] →

\[
\begin{array}{c}
\text{a, b, c, d}
\end{array}
\] → Complement.

Source 2(A,B,D) →

\[
\begin{array}{c}
\text{a, , c} \\
\text{a, b, , d}
\end{array}
\] →

\[
\begin{array}{c}
\text{a, b, c, -} \\
\text{a, b, -, d}
\end{array}
\] →

\[
\begin{array}{c}
\text{a, b, c, d}
\end{array}
\] →

\[
\begin{array}{c}
\text{a, , c} \\
\text{a, b, , d}
\end{array}
\] → Identical tuples

\[
\begin{array}{c}
\text{a, b, , -} \\
\text{a, b, , -}
\end{array}
\] →

\[
\begin{array}{c}
\text{a, b, , -} \\
\text{a, b, , -}
\end{array}
\] →

\[
\begin{array}{c}
\text{a, b, , -} \\
\text{a, b, , -}
\end{array}
\] →

\[
\begin{array}{c}
\text{a, b, , -} \\
\text{a, b, , -}
\end{array}
\] → Subsumed tuples

\[
\begin{array}{c}
\text{a, b, , c} \\
\text{a, b, , d}
\end{array}
\] →

\[
\begin{array}{c}
\text{a, b, , c} \\
\text{a, b, , d}
\end{array}
\] →

\[
\begin{array}{c}
\text{a, b, , c} \\
\text{a, b, , d}
\end{array}
\] →

\[
\begin{array}{c}
\text{a, b, , c} \\
\text{a, b, , d}
\end{array}
\] → Conflicting tuples

Felix Naumann | Information Systems | Hasso Plattner Institut
## Conflict Resolution Functions

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

....
Visualization of Integrated Data

![Visualization of Integrated Data](image_url)
Tool-based Data Fusion

### Tool-based Data Fusion

Felix Naumann | Information Systems | Hasso Plattner Institut

![Image of a data fusion tool](image)

#### Automatic Fusion

<table>
<thead>
<tr>
<th>Group</th>
<th>Title</th>
<th>Firstname</th>
<th>Lastname</th>
<th>Company</th>
<th>Country Code</th>
<th>Street</th>
<th>Number</th>
<th>Zip</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1253</td>
<td>Frau</td>
<td>Kostas</td>
<td>Daimle</td>
<td>96 A</td>
<td>10559 Berlin</td>
<td>Frau</td>
<td>1253</td>
<td>10559</td>
<td>Berlin</td>
</tr>
<tr>
<td>1353</td>
<td>Herr</td>
<td>Markus</td>
<td>Bauer</td>
<td>HPC Y.</td>
<td>10878 Berlin</td>
<td>Herr</td>
<td>1353</td>
<td>10878</td>
<td>Berlin</td>
</tr>
<tr>
<td>1782</td>
<td>Herr</td>
<td>Frank</td>
<td>Louis</td>
<td>Arenh</td>
<td>12103 Berlin</td>
<td>Herr</td>
<td>1782</td>
<td>12103</td>
<td>Berlin</td>
</tr>
<tr>
<td>1674</td>
<td>Monsieur</td>
<td>Frank</td>
<td>Eicker</td>
<td>D</td>
<td>79A 13599 Berlin</td>
<td>Monsieur</td>
<td>1674</td>
<td>13599</td>
<td>Berlin</td>
</tr>
<tr>
<td>2159</td>
<td>Herr</td>
<td>Horst</td>
<td>Schütz</td>
<td>D</td>
<td>10177 Berlin</td>
<td>Herr</td>
<td>2159</td>
<td>10177</td>
<td>Berlin</td>
</tr>
<tr>
<td>2159</td>
<td>Dr.</td>
<td>Familie</td>
<td>Hofmann</td>
<td>D</td>
<td>10569 Berlin</td>
<td>Familie</td>
<td>2159</td>
<td>10569</td>
<td>Berlin</td>
</tr>
<tr>
<td>2496</td>
<td>Frau</td>
<td>Julia</td>
<td>Görss</td>
<td>D</td>
<td>13509 Berlin</td>
<td>Frau</td>
<td>2496</td>
<td>13509</td>
<td>Berlin</td>
</tr>
<tr>
<td>2552</td>
<td>Herr</td>
<td>Ehlers</td>
<td>D</td>
<td>Über F.</td>
<td>10587 Berlin</td>
<td>Herr</td>
<td>2552</td>
<td>10587</td>
<td>Berlin</td>
</tr>
</tbody>
</table>

#### Manual Fusion

16. Group:

<table>
<thead>
<tr>
<th>Group</th>
<th>Title</th>
<th>Firstname</th>
<th>Lastname</th>
<th>Company</th>
<th>Country Code</th>
<th>Street</th>
<th>Number</th>
<th>Zip</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1874</td>
<td>Monsieur</td>
<td>Frank</td>
<td>Eicher</td>
<td>D</td>
<td>79A 13589 Berlin</td>
<td>Monsieur</td>
<td>1874</td>
<td>13589</td>
<td>Berlin</td>
</tr>
<tr>
<td>1874</td>
<td>Herr</td>
<td>Frank</td>
<td>Eicher</td>
<td>D</td>
<td>13589 Berlin</td>
<td>Herr</td>
<td>1874</td>
<td>13589</td>
<td>Berlin</td>
</tr>
<tr>
<td>1874</td>
<td>Herr</td>
<td>Frank</td>
<td>Eicher</td>
<td>D</td>
<td>13589 Berlin</td>
<td>Herr</td>
<td>1874</td>
<td>13589</td>
<td>Berlin</td>
</tr>
<tr>
<td>1874</td>
<td>Herr</td>
<td>Frank</td>
<td>Eicher</td>
<td>D</td>
<td>13589 Berlin</td>
<td>Herr</td>
<td>1874</td>
<td>13589</td>
<td>Berlin</td>
</tr>
<tr>
<td>1874</td>
<td>Monsieur</td>
<td>Frank</td>
<td>Eicher</td>
<td>D</td>
<td>78A 13589 Berlin</td>
<td>Monsieur</td>
<td>1874</td>
<td>13589</td>
<td>Berlin</td>
</tr>
</tbody>
</table>

### Tools

- Fuzzy Fusion
- Additional Information
- Test/Debug
- Automatische Fusion
- Regelbasierte Fusion
- Manuelle Fusion

### Parameters

- CITY
- Wert: Berlin
- Filter

### Columns

- Title
- Firstname
- Lastname
- Company
- Country Code
- Street
- Number
- Zip
- City
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

Required
Authors

Optional
Linda

Unwanted
Database Server
localhost
localhost:1234
localhost:8080

Files
longIDs_cust_AL.csv
longIDs_cust_MR.csv
longIDs_cust_SZ.csv

Transformations
LDAP Reader
LDAP Writer
Lookup

Felix Naumann | Information Systems | Hasso Plattner Institut
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
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 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

Felix Naumann | Info

naumann@hpi.uni-potsdam.de