

Metadata Management for Heterogeneous Information Systems

Erhard Rahm

University of Leipzig, Germany

<http://dbs.uni-leipzig.de>

- Introduction
- Metadata management for data warehousing
 - Metadata classification and models
 - ETL: Schema integration + data cleaning
 - Federated Metadata Architecture
- Model Management: an approach to generic metadata management
 - Representation of models and mappings
 - Operators: Match, Compose, Merge ...
 - Application to data warehousing
- Implementation of a generic Match operator

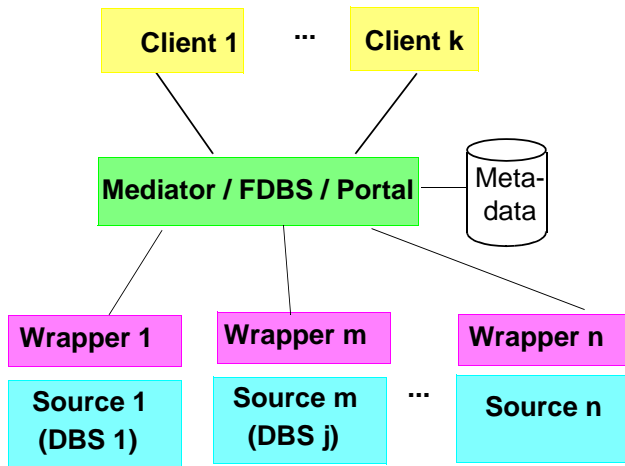
Metadata Management

- Need for metadata management is pervasive
- Many metadata languages: SQL, ODMG, UML, ER, XML, ontologies ...
- Mappings: ER-to-SQL, XML-to-XML, XML-to-SQL, UML-to-XML, SQL-to-web site map, ...
- Application Examples
 - DB design by mapping ER model to SQL schema
 - Web site design via models that map content (DB, files, etc.) to page layout and then generate pages
 - Heterogeneous data interchange (B2B) via XML where source tags are mapped to target tags
 - Generate data warehouse loading programs from mappings of data sources to data warehouse schema
 - Designing workflow applications by mapping from business process definitions to workflows
 - Heterogeneous DB integration, semantic query processing, DB evolution/migration, ...
 - Generating object wrappers from a mapping of classes to persistent storage objects ...

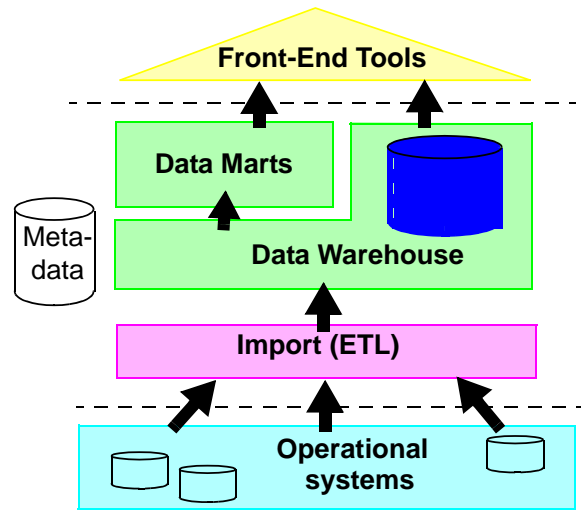
General Alternatives for Data Integration

Virtual Integration

(Mediator/Wrapper Architectures, federated DBS)

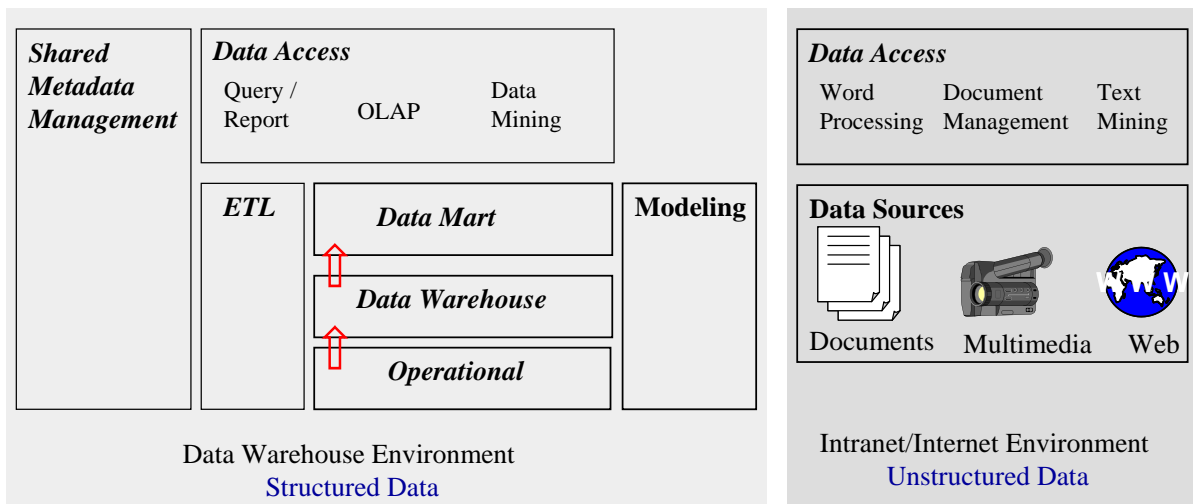


Physical (Pre-) Integration (Data Warehousing)



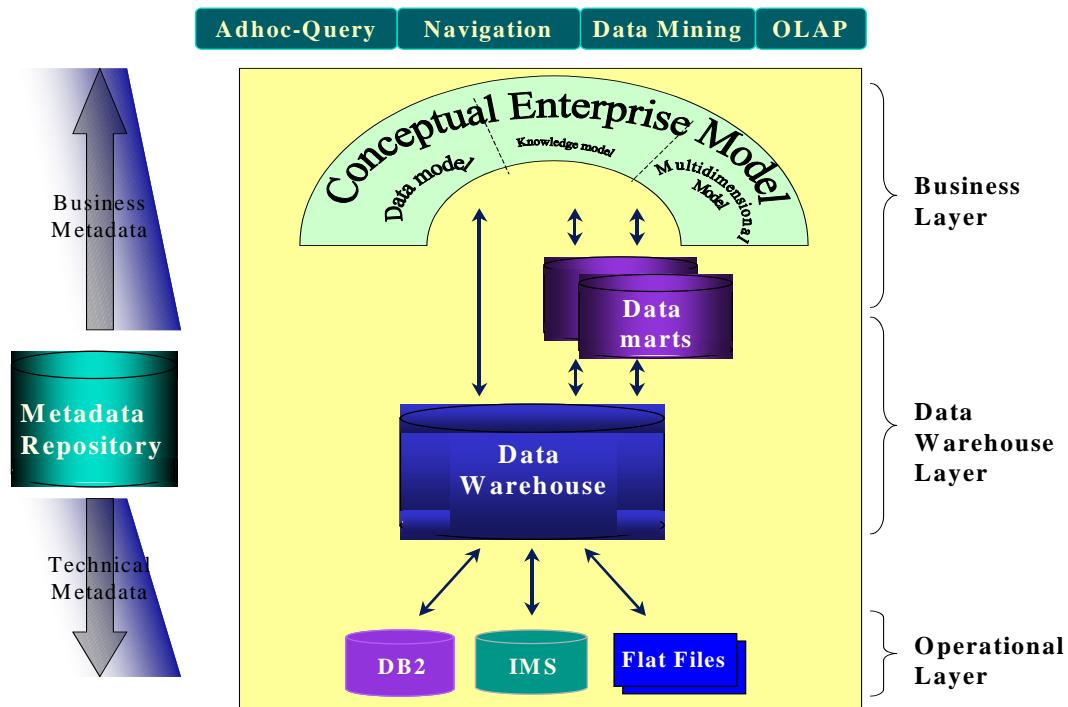
Enterprise Information Portals

Enterprise Information Portal

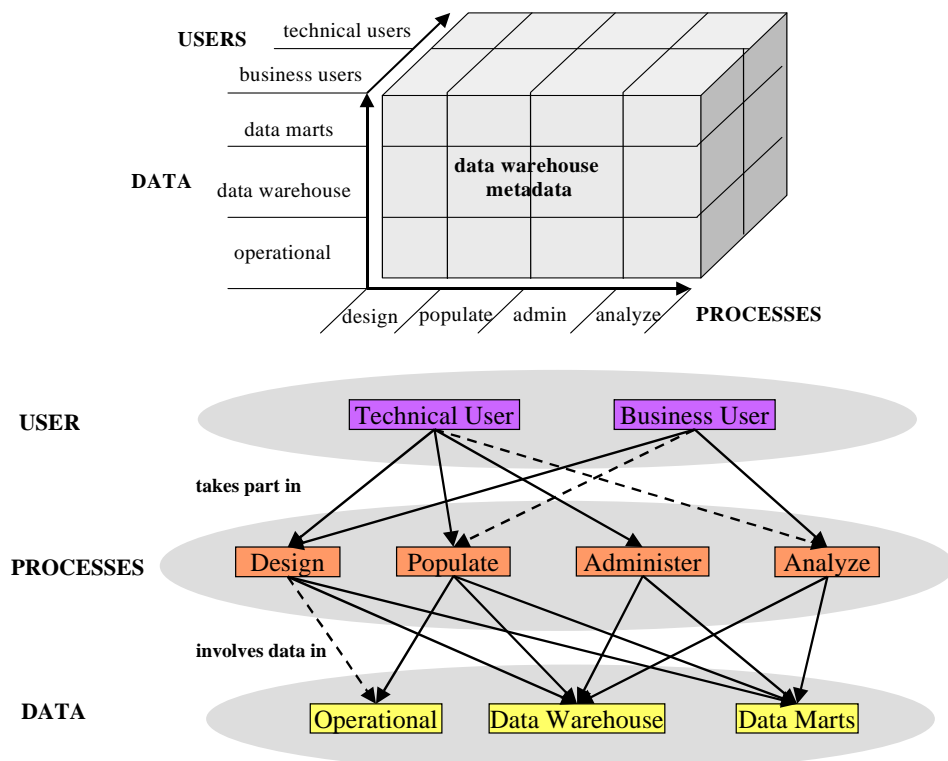


- Requirements: Integration of structured and unstructured data, tool integration, intelligent search, publish/subscribe mechanisms; personalization; authorization concept (user, groups, roles)

Data Warehouse Environment



Classification of Warehouse Metadata



Metadata Models for Data Warehousing

■ Requirements

- flexible representation of all relevant types of metadata
- consistent management of shared metadata
- extensibility
- automatic generation of code / scripts / queries to perform data transformations and data analysis

■ Proprietary models within commercial repositories

■ Research approaches

■ Standardization approaches based on UML:

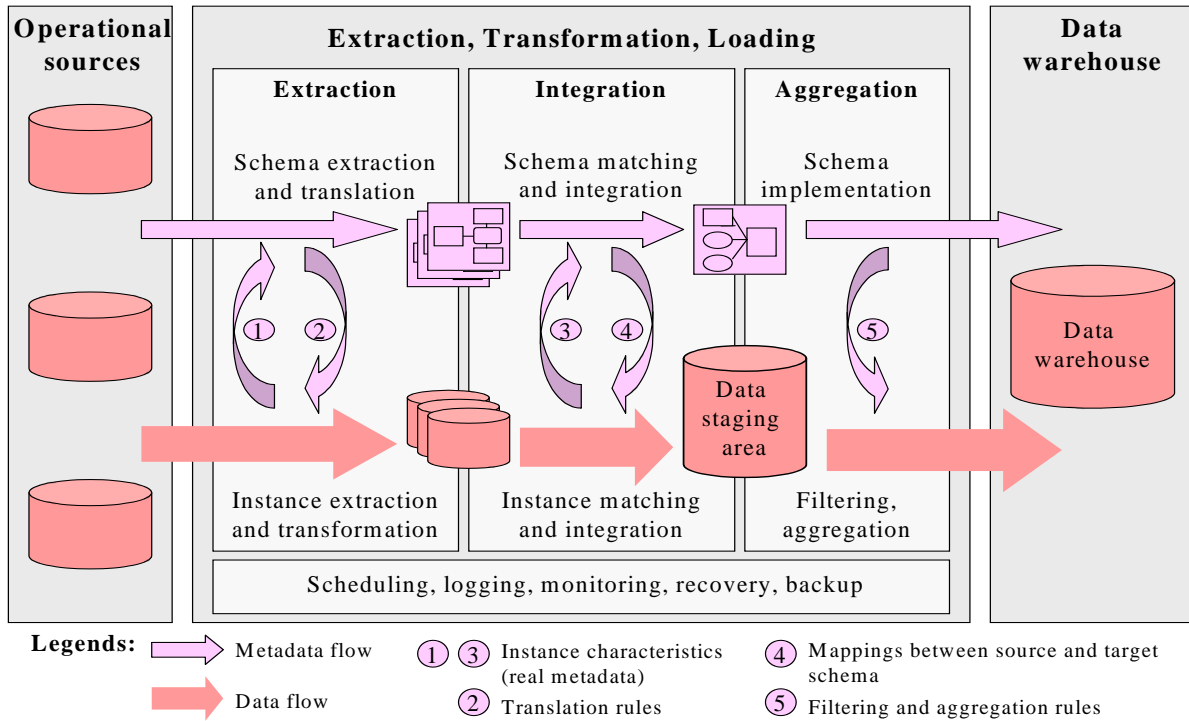
- Open Information Model (OIM): Metadata Coalition (MDC), Microsoft, Platinum, Sterling, ...
- Common Warehouse Model (CWM): Object Management Group (OMG), IBM, Oracle, Unisys, ...

Technical and Business Metadata in OIM, CWM

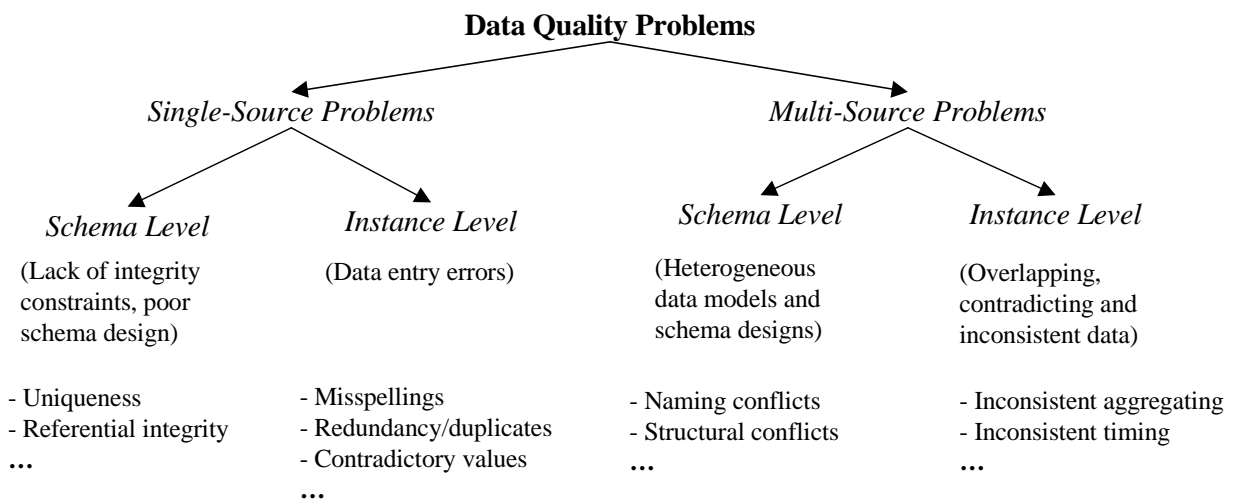
| Technical Metadata | | OIM | CWM |
|-------------------------------------|------------------|--|--|
| Data Schema | Relational | Relational Database Schema | Relational Data Resource |
| | Record-oriented | Record-oriented Legacy Database Schema | Record-oriented Data Resource |
| | Multidimensional | OLAP Schema | Multidimensional Data Resource OLAP |
| | XML | XML Schema | XML Data Resource |
| Data Transformation | | Data Transformations | Transformation |
| Warehouse Operation and Maintenance | | | Warehouse Deployment, Warehouse Process, Warehouse Operation |
| Business Metadata | | OIM | CWM |
| | | Report Definitions Knowledge Descriptions Semantic Definitions | CWM Foundation: Business Information (data stewardship, textual descriptions) |

- Comprehensive metadata models covering many subject areas of data warehousing
- CWM: little support for business metadata
- OIM: little support for technical metadata about warehouse operation and maintenance, but richer sets of business metadata
- Both models: no support for user management, access rights, personalized views on warehouse data

ETL: Extraction, Transformation, Loading



Classification of Data Quality Problems



Example of Multi-Source Problems

Customer (source 1)

| CID | Name | Street | City | Sex |
|-----|-----------------|-------------|----------------------|-----|
| 11 | Kristen Smith | 2 Hurley Pl | South Fork, MN 48503 | 0 |
| 24 | Christian Smith | Hurley St 2 | S Fork MN | 1 |

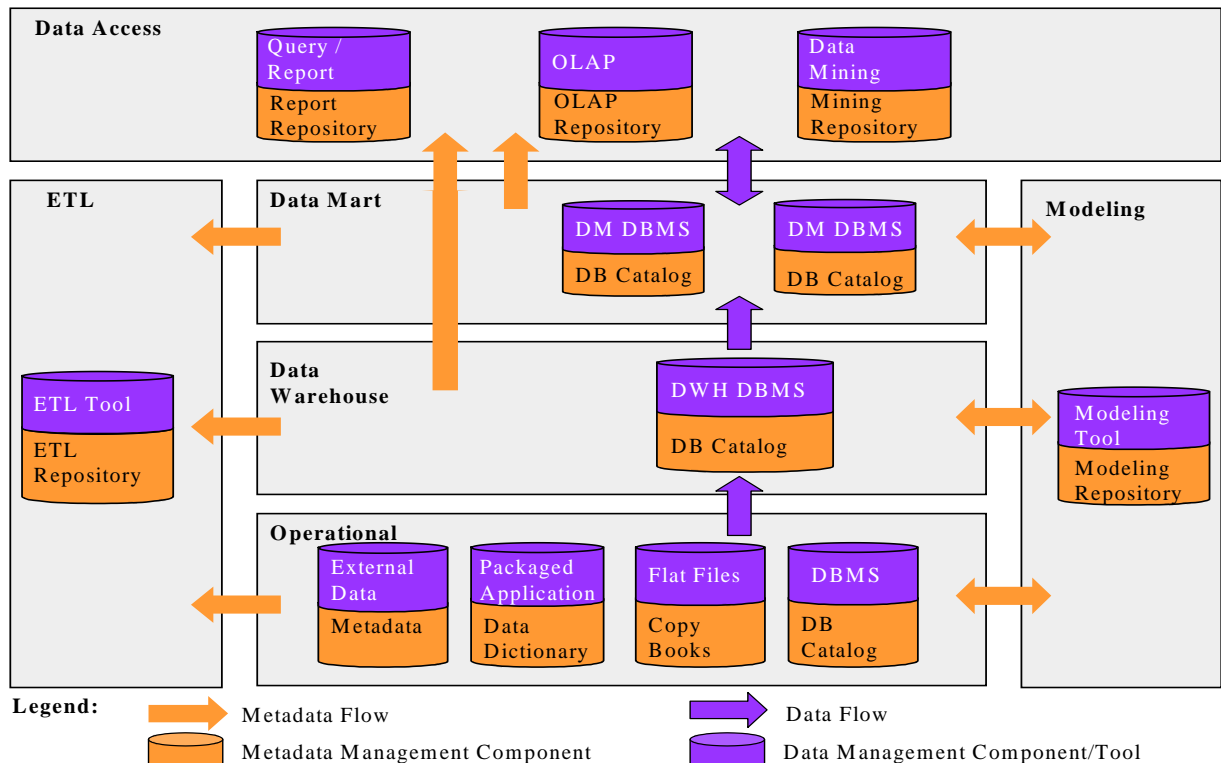
Client (source 2)

| Cno | LastName | FirstName | Gender | Address | Phone/Fax |
|-----|----------|-----------|--------|---|-----------------------------|
| 24 | Smith | Christoph | M | 23 Harley St, Chicago IL, 60633-2394 | 333-222-6542 / 333-222-6599 |
| 493 | Smith | Kris L. | F | 2 Hurley Place, South Fork MN, 48503-5998 | 444-555-6666 |

Customers (integrated target with cleaned data)

| No | LName | FName | Gender | Street | City | State | ZIP | Phone | Fax | CID | Cno |
|----|-------|------------|--------|------------------|------------|-------|------------|--------------|--------------|-----|-----|
| 1 | Smith | Kristen L. | F | 2 Hurley Place | South Fork | MN | 48503-5998 | 444-555-6666 | | 11 | 493 |
| 2 | Smith | Christian | M | 2 Hurley Place | South Fork | MN | 48503-5998 | | | 24 | |
| 3 | Smith | Christoph | M | 23 Harley Street | Chicago | IL | 60633-2394 | 333-222-6542 | 333-222-6599 | | 24 |

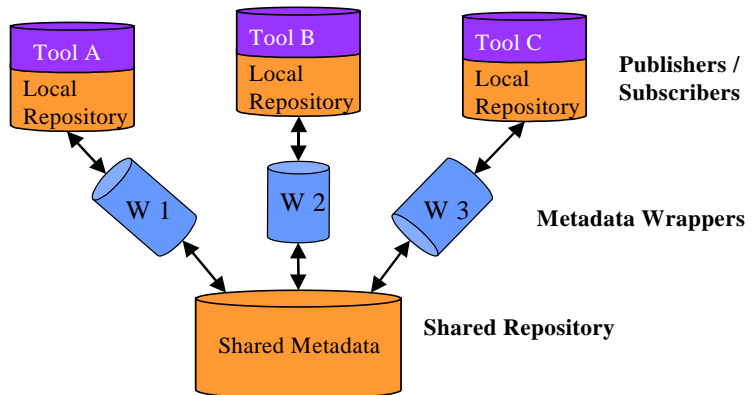
Distributed Metadata



Federated Metadata Architecture

- Use of local repositories + repository for shared metadata

- autonomy of local repositories
- uniform representation of shared metadata
- reduced number of connections between repositories
- controlled replication of metadata



- Metadata wrapper

- mapping of different metadata representations
- asynchronous (file exchange) or synchronous (API-based)

- File Exchange (asynchronous)

- platform-independent, easy to implement
- MDIS, CDIF, XML, ...
- format translation mechanisms hard-coded in tools / repositories

- API (synchronous)

- mostly proprietary APIs
- high effort for application development

Metadata Replication Control

- replication of metadata in warehouse tools / repositories unavoidable

- „Lazy“ synchronization (serializable approaches not possible / too expensive)

- Deferred propagation of updates between „publisher“ and „subscribers“

- notification (push): publishers „push“ updates to subscribers
- probing (pull): subscribers detect and „pull“ changes from publishers

- Shared Repository

- has both roles, publisher and subscriber
- registers publishers / subscribers and their sets of published / subscribed metadata for change detection and impact analysis

- Two-step update propagation: 1: publisher - shared repository; 2: shared repository - subscribers

| Combination | Implementation of Step 1 | Implementation of Step 2 |
|----------------|--------------------------|--------------------------|
| 1: Push / Pull | Publishers | Subscribers |
| 2: Push / Push | Publishers | Shared Repository |
| 3: Pull / Pull | Shared Repository | Subscribers |
| 4: Pull / Push | Shared Repository | Shared Repository |

Observations

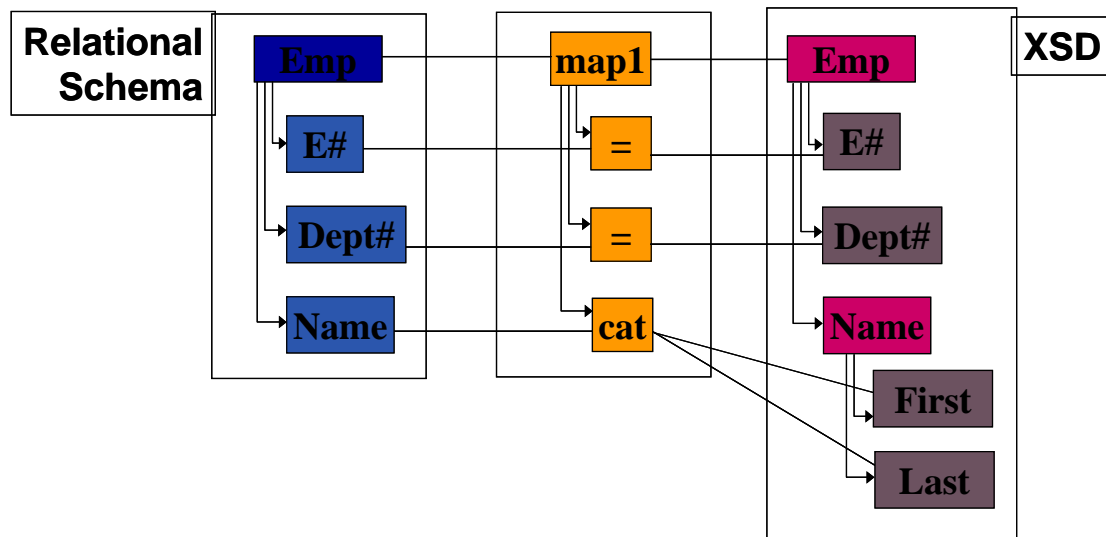
- Data integration, portals, mediators, digital libraries, E-business ... require flexible metadata management and metadata interoperability / integration
- Current metadata repositories not flexible and powerful enough
 - low-level repository APIs make it difficult to develop tools and metadata-based applications
 - re-implementation of similar metadata management functionality in many tools and applications
 - difficult metadata interoperability and integration
- More powerful, more generic metadata management needed
 - easy integration of new models (schemas, vocabularies, ...)
 - much easier development of metadata-based applications
- XML helpful but not enough
 - primarily covers syntax, not semantics
 - many similar but different schemas
 - competing „standards“
- Fully automatic approaches to metadata integration not possible

Model Management

- Models and mappings are first-class objects
- Define generic high-level operations on models and mappings, e.g., Match, Merge, Select, Compose,
- Apply operations to real problems
- Implement operations on a DBMS
- Use the implementation

A Model for Model Management

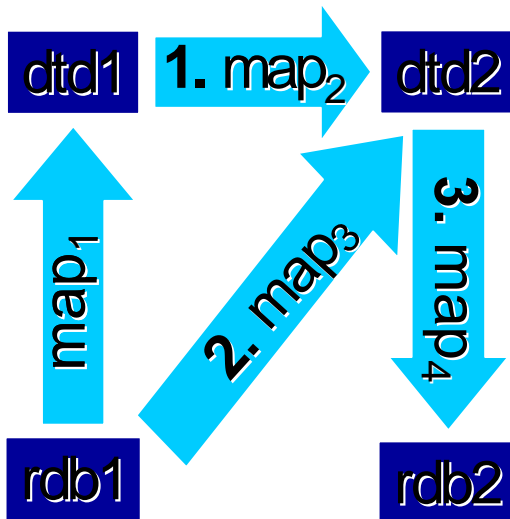
- A model is a directed graph with one root
- A mapping is a model each of whose nodes connects nodes of two other models



Basic Operations

- Match ($M1, M2, \cong$)
- Merge ($M1, M2, \text{map}$)
- Compose ($\text{map1}, \text{map2}$)
- ApplyFunction (M, f)
- Set Difference ($M1, M2$)
- Select (M, pred)
- Insert, Update, Delete, Copy, ...

Example



1. $map_2 = \text{Match}(\text{dtd1}, \text{dtd2})$

2. $map_3 = map_1 \bullet map_2$

3. $\langle map_4, \text{rdb2} \rangle = \text{Copy}(map_3^{-1})$

Model

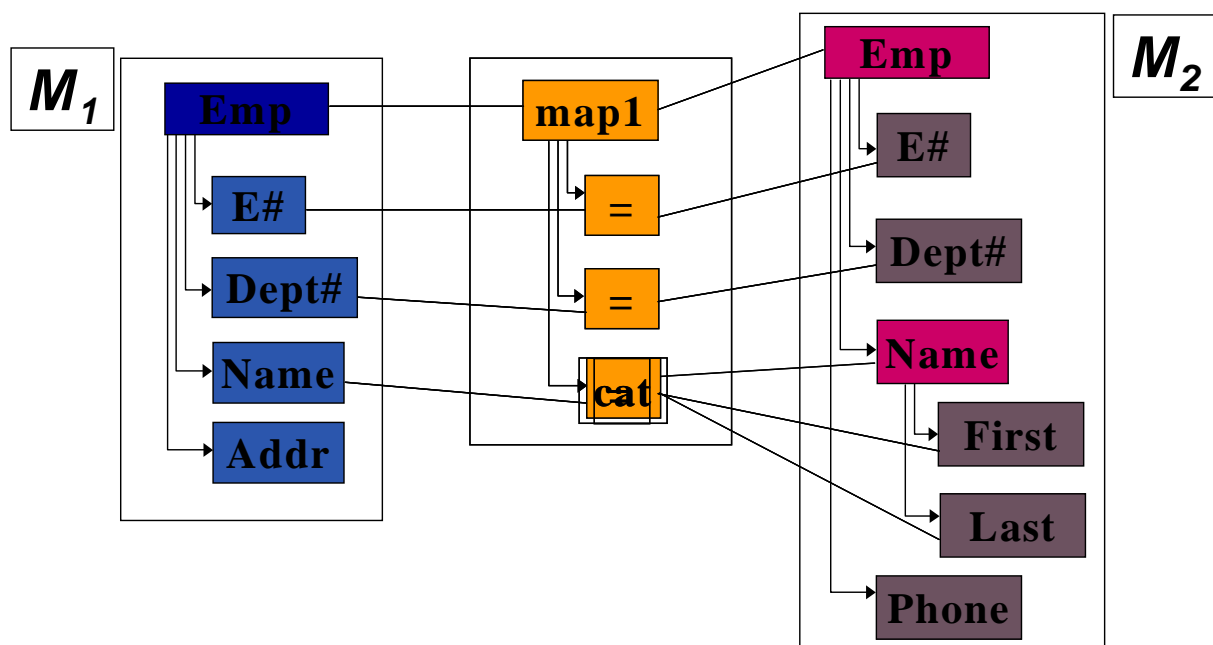
- Represent a model by a directed graph
- Some edges are of type containment
- Model = the transitive closure of containment edges reachable from the model's root.
- Nodes have content (i.e. properties)
- Non-containment edges are connections between models
- How much semantics should be inherent to the concept of Model?
 - Not too much, so it's generic across application areas (trade off generic-ness vs. expressiveness)
 - Enough to define powerful operations
 - At least: entity, attribute, data type, key; Isa, derived-from; contains, aggregates

Mapping

- A mapping is a model, so it can be copied, deleted, selected, etc.
 - Mappings often connect different types of models (e.g., DB schema & XML schema)
 - Like any model, a mapping can have internal structure ...
 - A mapping can be a function, invertible, partial or total, onto, etc.
- Mapping objects: domain objects, range objects, mapping expression
- Semantics – an expression per mapping object
 - Mapping can be purely structural (no expressions)
 - Still adds value, e.g. by enabling Match and Diff
- Extensibility for different expression languages (based on logic, algebra, grammars, etc.)

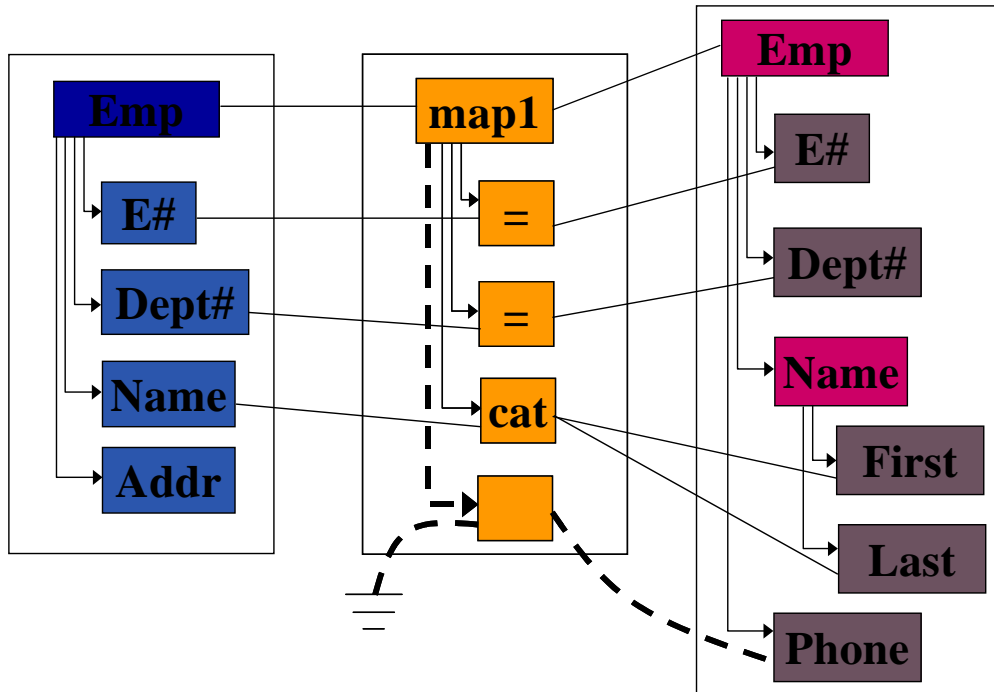
Match

- $\text{Match}(M_1, M_2, \cong)$ returns best mapping between M_1 and M_2 , w.r.t. to \cong



OuterMatch

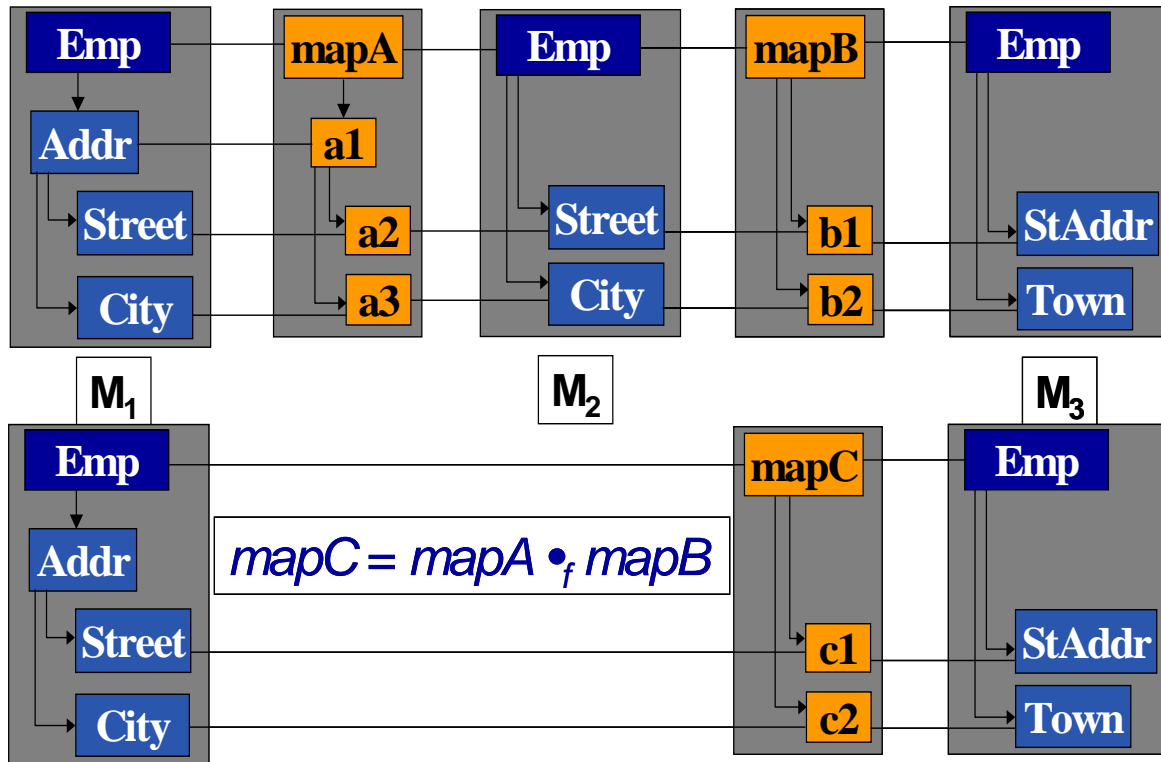
- $\text{RightOuterMatch}(M1, M2, \cong)$ is same as Match but covers all of $M2$.



Composition

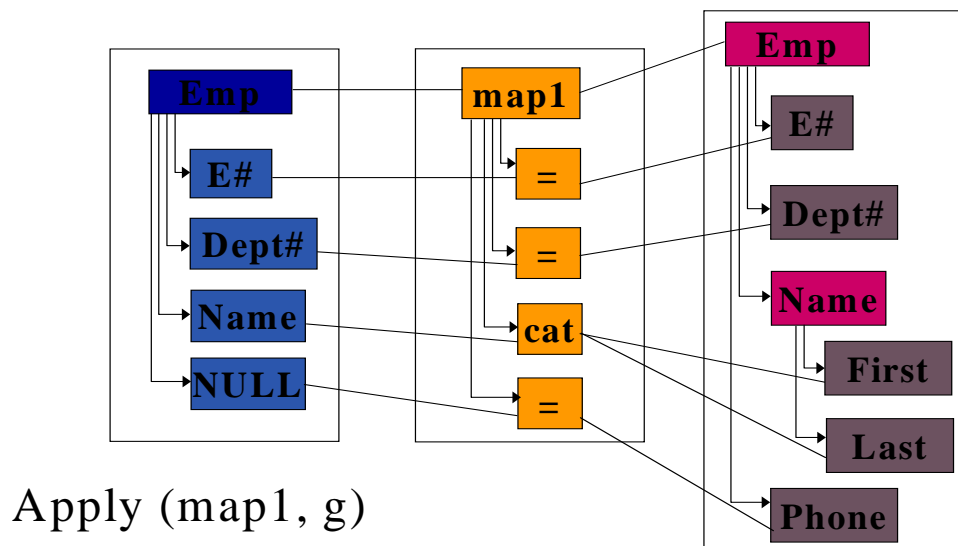
- Notation “ $\text{map1} \cdot \text{map2}$ ”
- Easy for single-valued functions: just use ordinary function composition
- set-valued functions: different composition semantics useful
- use one of the models to drive the composition
 - Left Composition
 - Right Composition

Right Composition (\circ_f)

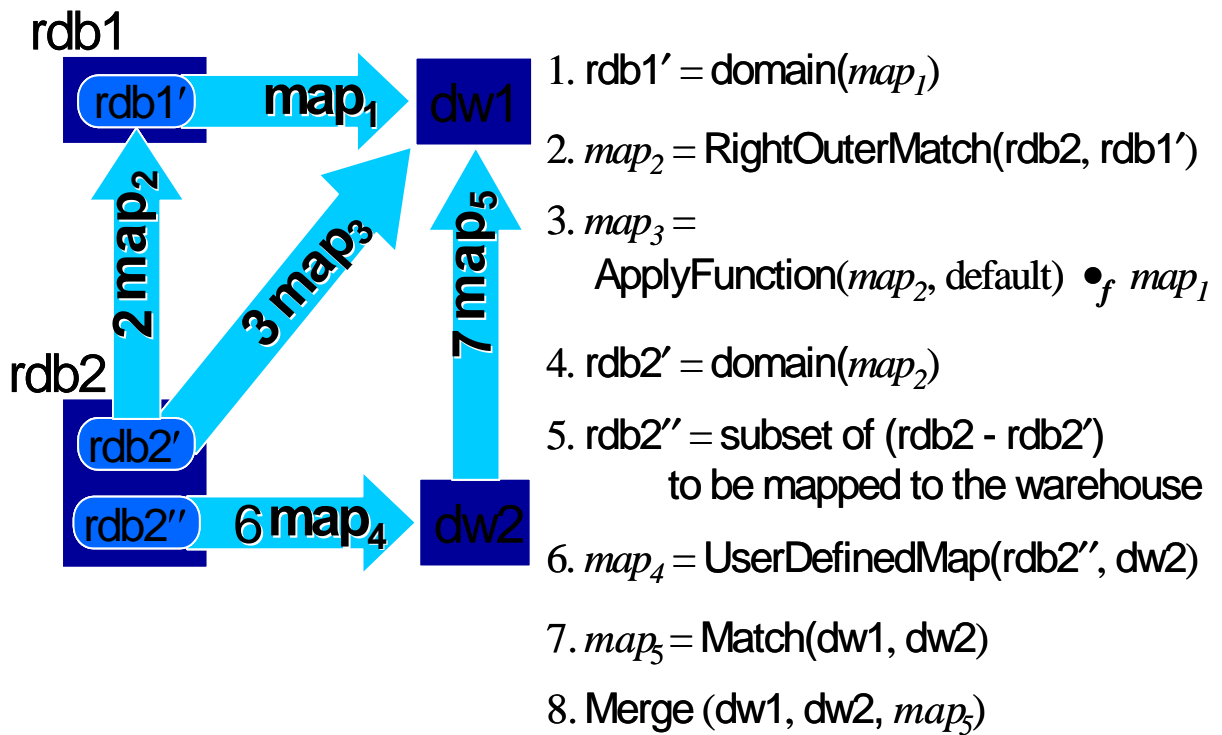


ApplyFunction

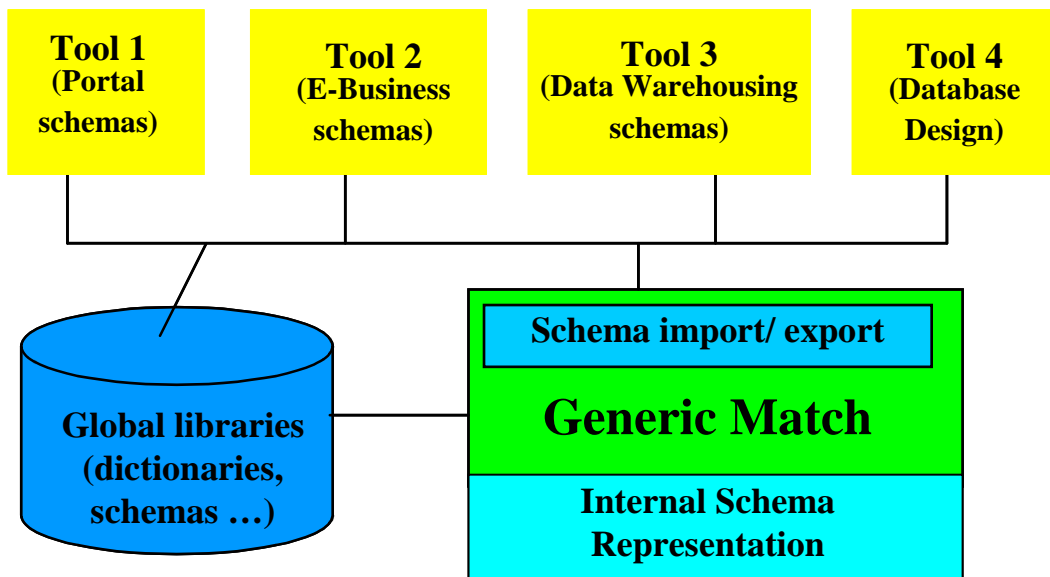
- Apply a function to all objects of a model
- Examples
 - f: Append “_2” to all names
 - g: Set domain(m)= “=NULL” where domain(m)= \emptyset



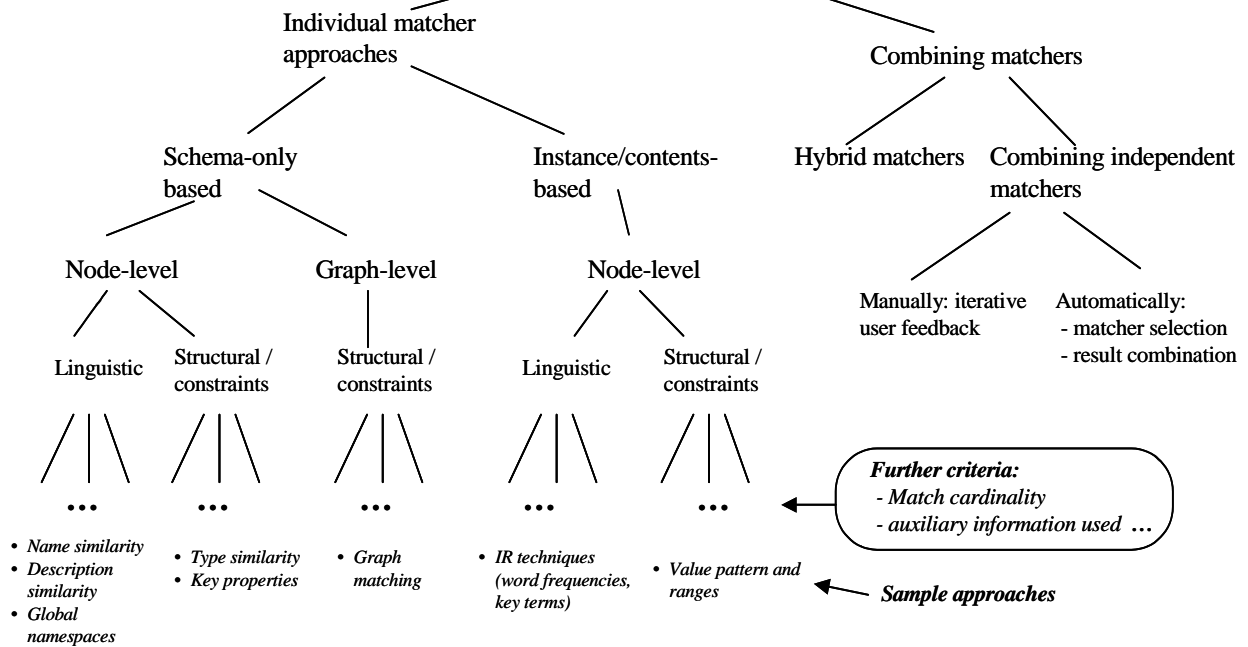
Model Management Scenario for Data Warehousing



Generic MATCH



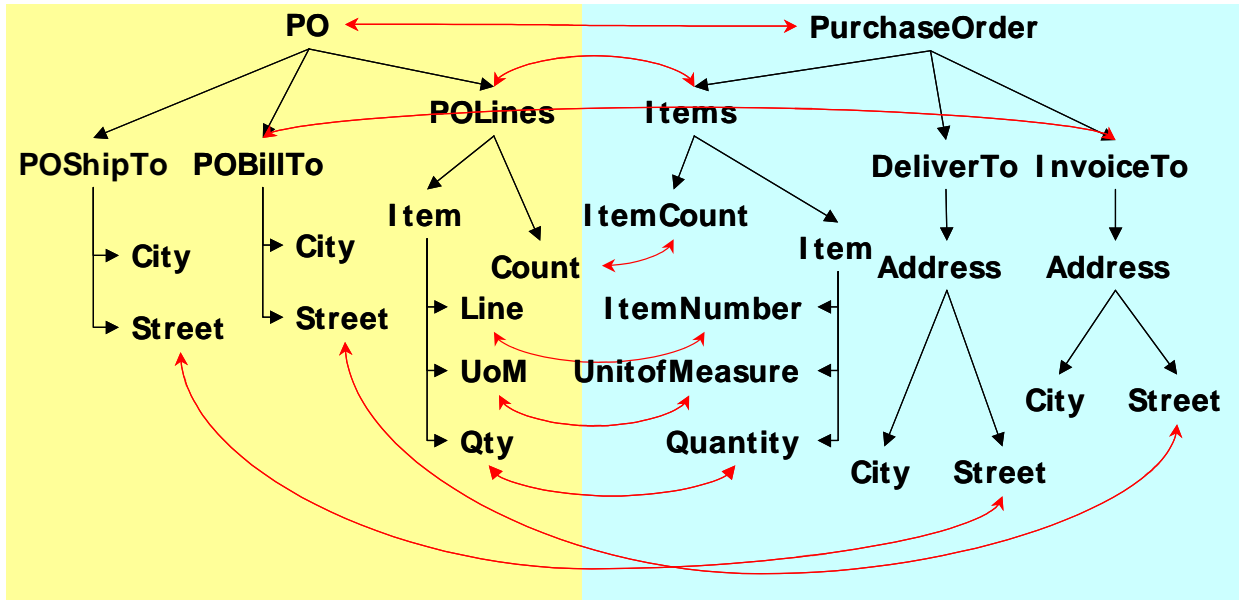
Schema Matching Approaches



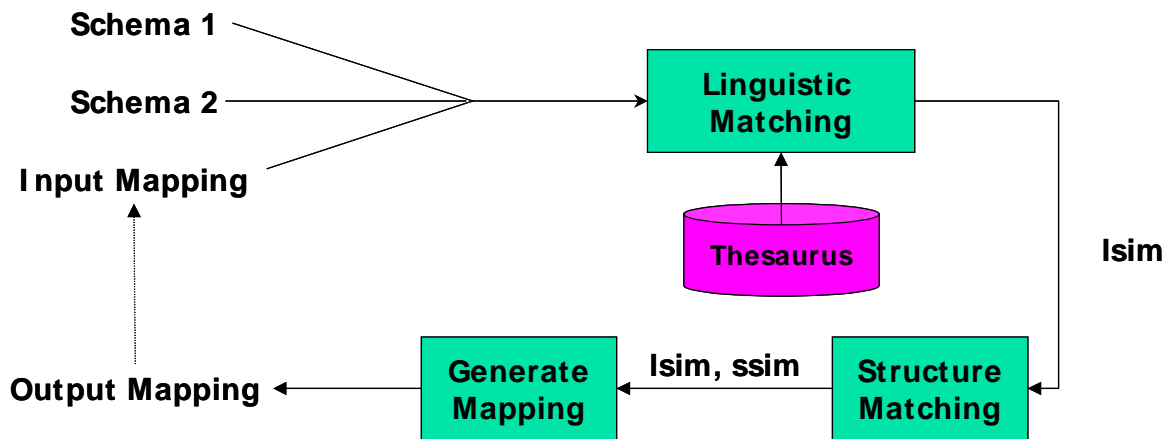
Cupid Approach to Match (VLDB-01)

- New algorithm to match schemas – using linguistics, data types, structure and referential integrity
- Prototype that demonstrates the approach
- Experimental validation and comparison with other systems (MOMIS, Bergamaschi et al.; DIKE - Palopoli et al.)
- Characteristics
 - Schema based
 - Structure
 - Linguistic
 - Auxiliary information
 - Hybrid

Cupid in action



The Cupid Architecture



Linguistic Matching

- Names, data-types, aggregation
 1. Normalization of names of schema elements
 - Tokenization, Expansion, Elimination
 2. Categorization
 - Clustering to reduce number of comparisons
 3. Linguistic similarity computation
 - Elements belonging to compatible categories
 - Thesaurus with similarity coefficients is used
- Linguistic Similarity Coefficient (lsim)

Structural Matching

- A schema is a tree of schema elements
- Intuition –
 - Atomic elements are similar if
 - Individually similar (linguistic and data type)
 - Ancestors are similar
 - Non-leaf elements are similar if
 - Linguistically similar
 - Subtrees rooted at the nodes are similar
 - Subtrees are similar if
 - Immediate children are similar
 - Leaf sets are similar

Tree Match

Tree Match(SchemaTree S, TargetTree T)

For each pair of leaves s,t in the two trees

Initialize $ssim(s,t) = \text{datatype-compatibility}(s,t)$

For each s in S (post order)

For each t in T(post order)

Compute $ssim(s,t) = \text{structural-similarity}(s,t)$

$wsim(s,t) = g(lsim(s,t), ssim(s,t))$

If ($wsim(s,t) > th_{high}$)

Inc-struct-similarity(leaves(s), leaves(t))

If ($wsim(s,t) < th_{low}$)

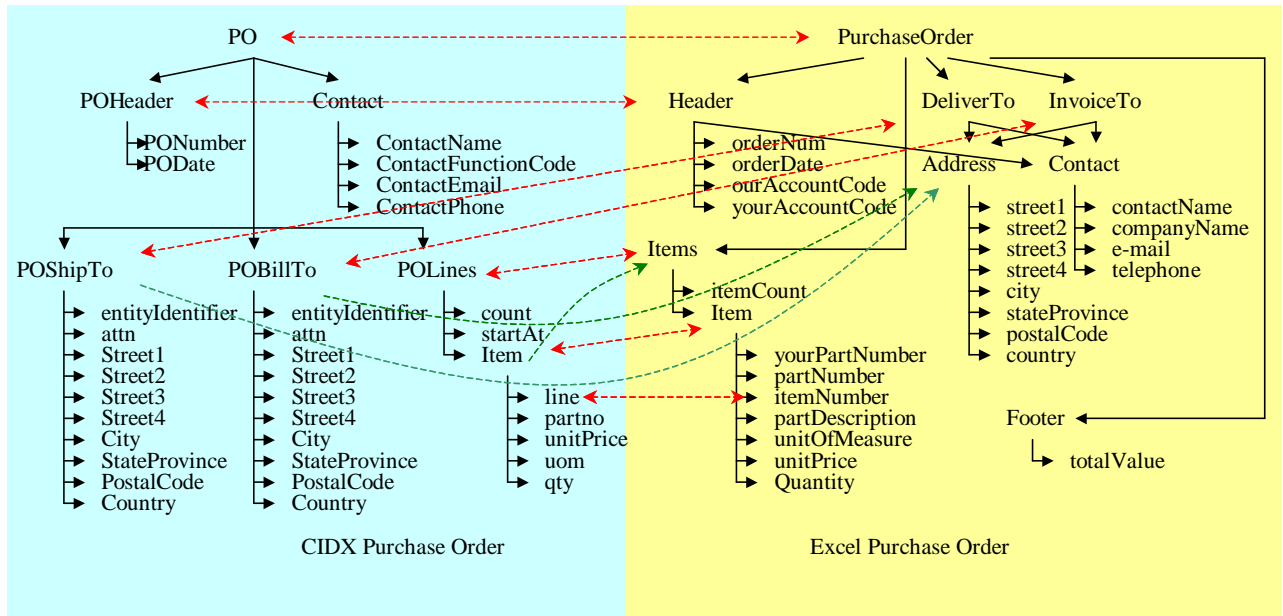
Dec-struct-similarity(leaves(s), leaves(t))

Evaluation

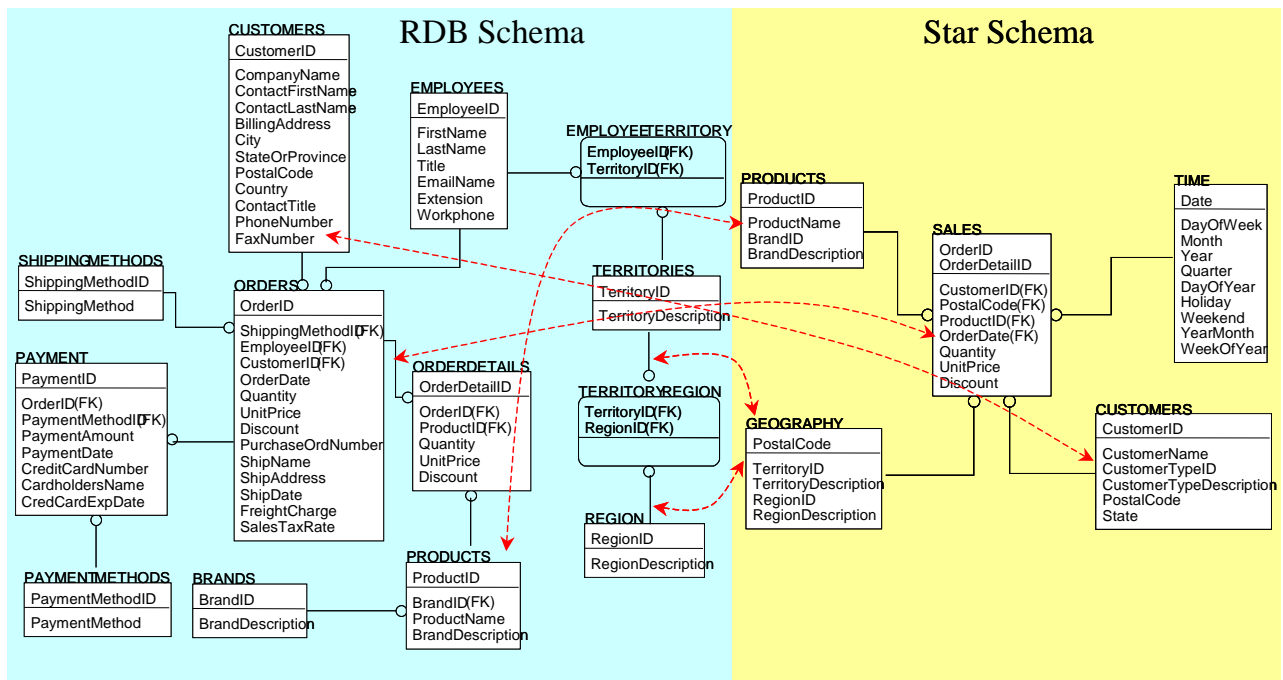
- Comparison with two other schema-based, structural matchers: MOMIS, DIKE
- Evaluation for canonical and real world examples (XML-XML, SQL-XML, SQL-SQL)

| | MOMIS | DIKE | Cupid |
|---|-------|------|-------|
| Identical schemas | Y | Y | Y |
| Attributes with identical names, but different data-types | Y | Y | Y |
| Attributes with same data-types, but slightly different names | Y | N | Y |
| Different class names, but same attribute names | N | Y | Y |
| Different nesting of schema elements | N | Y | Y |
| Type substitution | N | Y | Y |

Real XML schemas



Real SQL schemas



Evaluation insights

- Linguistic matching
 - Mode of linguistic input – WordNet, manual
 - Role of the thesaurus
 - Linguistic similarity without structural similarity
- Structural similarity
 - Granularity of similarity computation
 - Leaves vs. immediate structure
 - Similarity beyond immediate vicinity
 - Context dependent mapping

Summary

- Web applications, data warehousing etc. depend on flexible metadata management and metadata interoperability and integration
- Current metadata situation:
 - co-existence of heterogeneous local repositories with proprietary metadata models
 - mapping and integration problems
 - low-level repository APIs
- New generation of metadata approaches needed, e.g. Model management
 - uniform representation of models and mappings
 - high-level operations: Match, Merge, Compose, ...
 - generic: applicable to different domains and different languages
- Implementation of a generic Match operation
 - utilization of several criteria: linguistic + structural
 - utilization of schema information + instance data
- Fully automatic solutions not possible, e.g. for metadata integration / schema match

Open Problems

- Model management
 - more precise definitions of operations
 - plug-in capability for different expression languages
 - efficient algorithms / implementations for operators (Match, Compose, Merge)
 - evaluation of effectiveness of Match etc. (precision / recall problem)
- Applications / tools utilizing model management
- Standardization to limit heterogeneity
- Other „next-generation“ metadata management approaches

References

- Model Management
 - P. Bernstein, E. Rahm: *Data Warehouse Scenarios for Model Management*. Proc. 19th Int. Conf. on Entity-Relationship Modeling, LNCS, Oct. 2000. dol.uni-leipzig.de/pub/2000-24
 - P. Bernstein et al.: *A Vision of Management of Complex Models*, ACM SIGMOD Record, Vol. 29, No. 4, Dec. 2000
- Match
 - E. Rahm, P. Bernstein: *On Matching Schemas Automatically*. Techn. Report, Feb. 2001. dol.uni-leipzig.de/pub/2001-5
 - J. Madhavan, P. Bernstein, E. Rahm: *Generic Schema Matching with Cupid*. Proc. 27th Intl. Conference on Very Large Databases (VLDB), Rome, Italy, Sep. 2001
- Data Warehouse Metadata Management
 - E. Rahm, H. Do: *Data Cleaning: Problems and Current Approaches*. IEEE Techn. Bulletin on Data Engineering, Dec. 2000. dol.uni-leipzig.de/pub/2000-45
 - R. Müller, T. Stöhr, E. Rahm: *An Integrative and Uniform Model for Metadata Management in Data Warehousing Environments*. Proc. DMDW'99, dol.uni-leipzig.de/pub/1999-22
 - H. Do, E. Rahm: *On Metadata Interoperability for Data Warehouses*. Univ. of Leipzig, 2000, dol.uni-leipzig.de/pub/2000-13
- Web: dbs.uni-leipzig.de bzw. dol.uni-leipzig.de