Generic Model Management: Experiences and Open Questions

Sergey Melnik

Leipzig University / Stanford University

Supervisor: Erhard Rahm

Goal:

- Reduce amount of programming for building metadata-driven applications

Model Management:

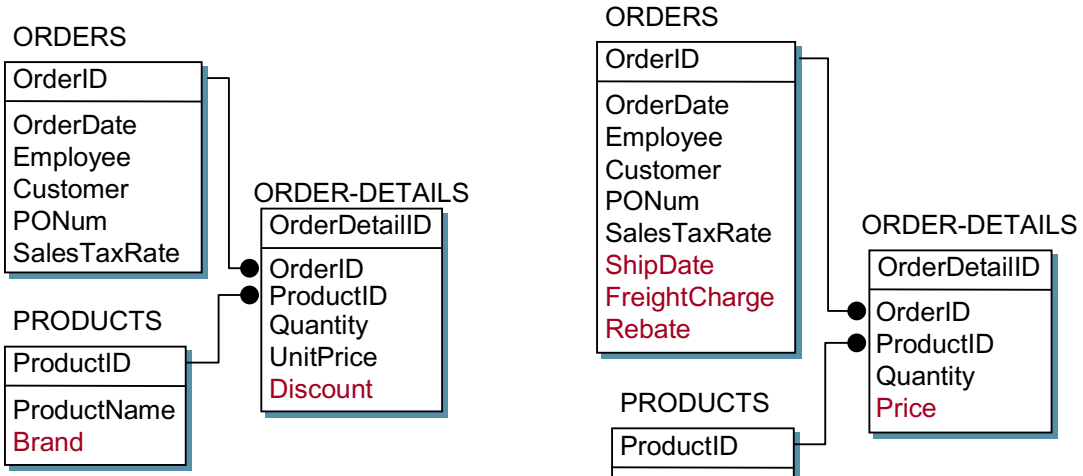
- General-purpose system for managing complex models
- Algebraic operations to manipulate metadata in large chunks

Thesis questions:

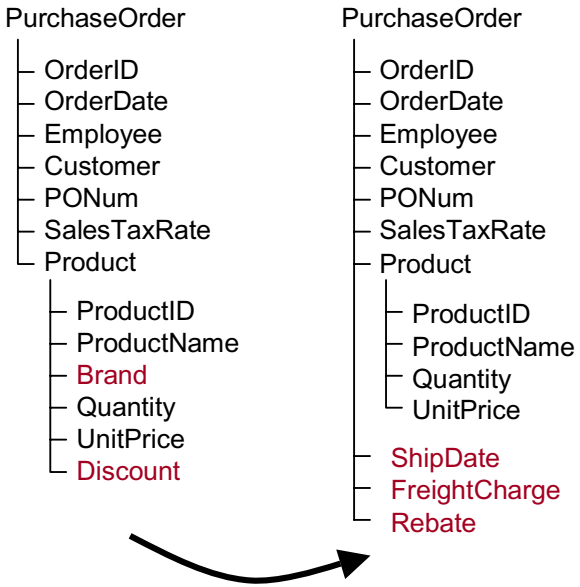
- Can model management be done in a generic fashion?
- Does generic model management offer practical benefits?

Sample scenario: Data Translation

Source schemas (relational):

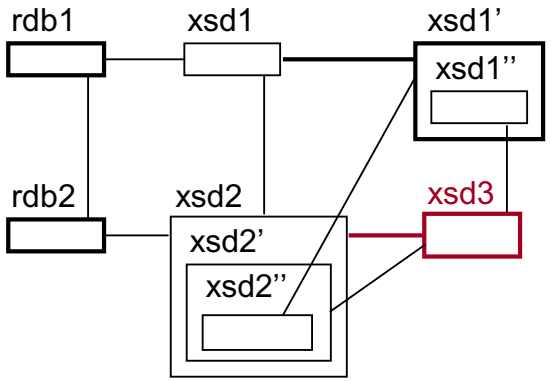


Export schemas (XML schema):




```
operator PropagateChanges(rdb1, rdb2, xsd1', xsd1_xsd1')

    rdb1_rdb2 = Match(rdb1, rdb2, NGram(rdb1, rdb2))
    (xsd1, rdb1_xsd1) = SQLDDL2XSD(rdb1)
    (xsd2, rdb2_xsd2) = SQLDDL2XSD(rdb2)
    xsd1_xsd2 = Match(xsd1, xsd2, Invert(rdb1_xsd1) • rdb1_rdb2 • rdb2_xsd2)
    xsd1'' = Delete(xsd1', Range(RestrictDomain(xsd1_xsd1'), All(xsd1) – Domain(xsd1_xsd2)))
    xsd2_add = All(xsd2) – Range(xsd1_xsd2)
    xsd2' = Select(xsd2, xsd2_add)
    xsd2'' = Delete(xsd2', xsd2_add)
    xsd2'_xsd1'' = RestrictDomain(Invert(xsd1_xsd2) • xsd1_xsd1'), All(xsd2''))
    xsd3 = Merge(xsd2', xsd1'', xsd2'_xsd1'')
    xsd2_xsd3 = (Invert(xsd1_xsd2) • xsd1_xsd1') ∪ Id(xsd2_add)
    return (xsd3, xsd2_xsd3)
```

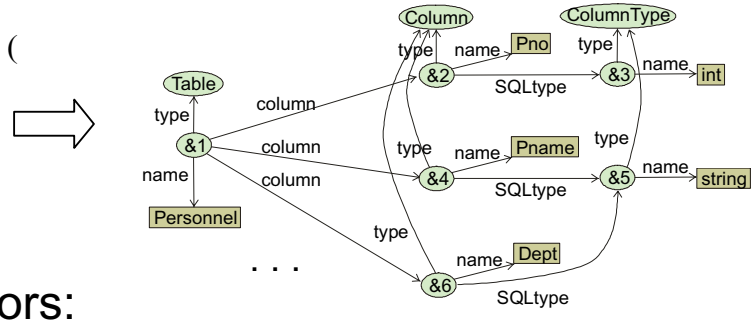


Operators and Data Structures

Data structures:

Model: directed labeled graph w/ OIDs and literals
Selector: set of OIDs
Mapping: (weighted) binary relation on OIDs

CREATE TABLE *Personnel* (
Pno int,
Pname string,
Dept string,
. . .)



Primitive operators:

Domain(map):
set of OIDs that are in the domain of map

RestrictDomain(map, selector):
mapping w/ domain restricted by selector

Id(selector): identity mapping

Invert(map): inverts a mapping

Compose(map1, map2): composition of mappings

TransitiveClosure(map):
returns the transitive closure of map

All(M): set of all OIDs used in model M

∪ , ∩ , – : set operators

Subgraph(M, selector):
subgraph of M induced by the nodes in selector

Derived operators:

```
operator Reachable(selector, map)
    return Range(RestrictDomain(TransitiveClosure(map), selector))

operator Select(M, selector, dependencies)
    return Subgraph(M, selector ∪ Reachable(selector, dependencies))

operator DeleteSoft(M, selector, dependencies)
    cannotBeDeleted = Reachable(All(M) – selector, dependencies)
    toDelete = selector – cannotBeDeleted
    toKeep = All(M) – toDelete
    return Select(M, toKeep, dependencies)

operator DeleteHard(M, selector, dependencies)
    toDelete = selector ∪ Reachable(selector, Invert(dependencies))
    toKeep = All(M) – toDelete
    return Select(M, toKeep, dependencies)
                                                    . . .
```

Semiautomatic operators:

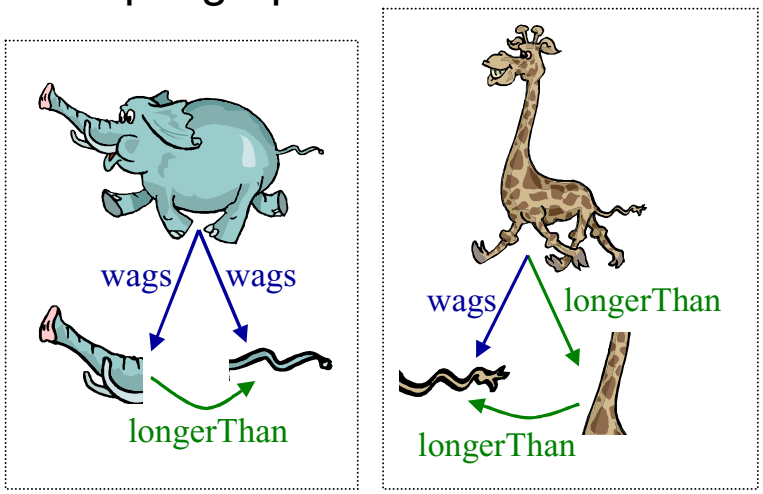
operator Match(M1, M2, map12): correspondences between elements
operator Merge(M1, M2, map12): use map12 for glueing M1 and M2

Language-specific operators:

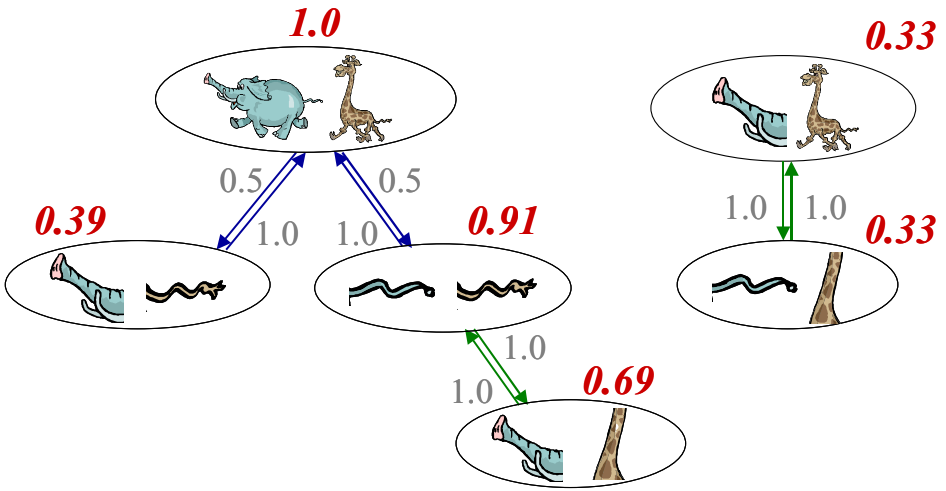
DependenciesSQL(M), DependenciesXSD(M): model elements required for consistency, e.g., field table

Matching: Similarity Flooding Algorithm

Sample graphs:

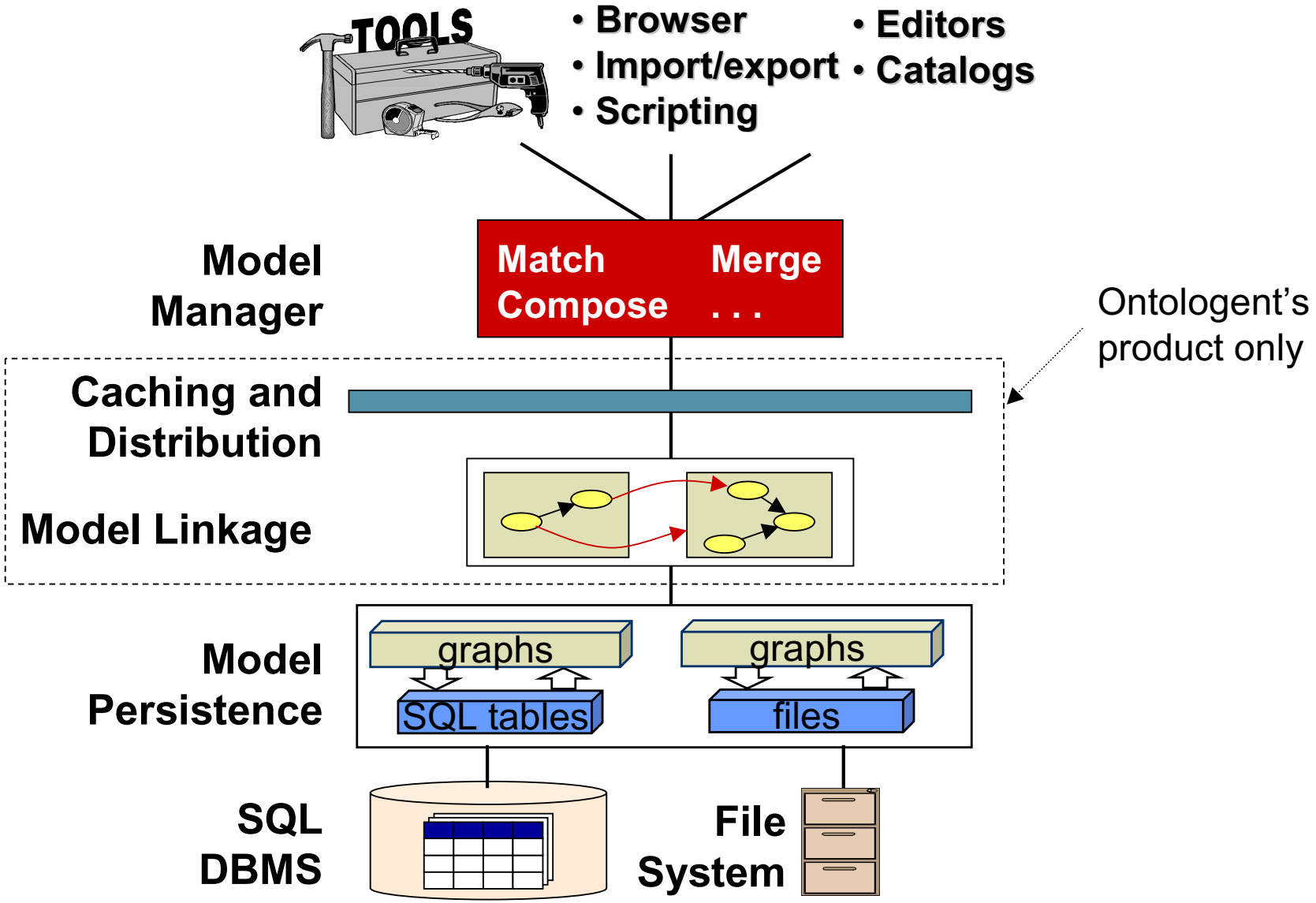


Fixpoint computation on propagation graph:



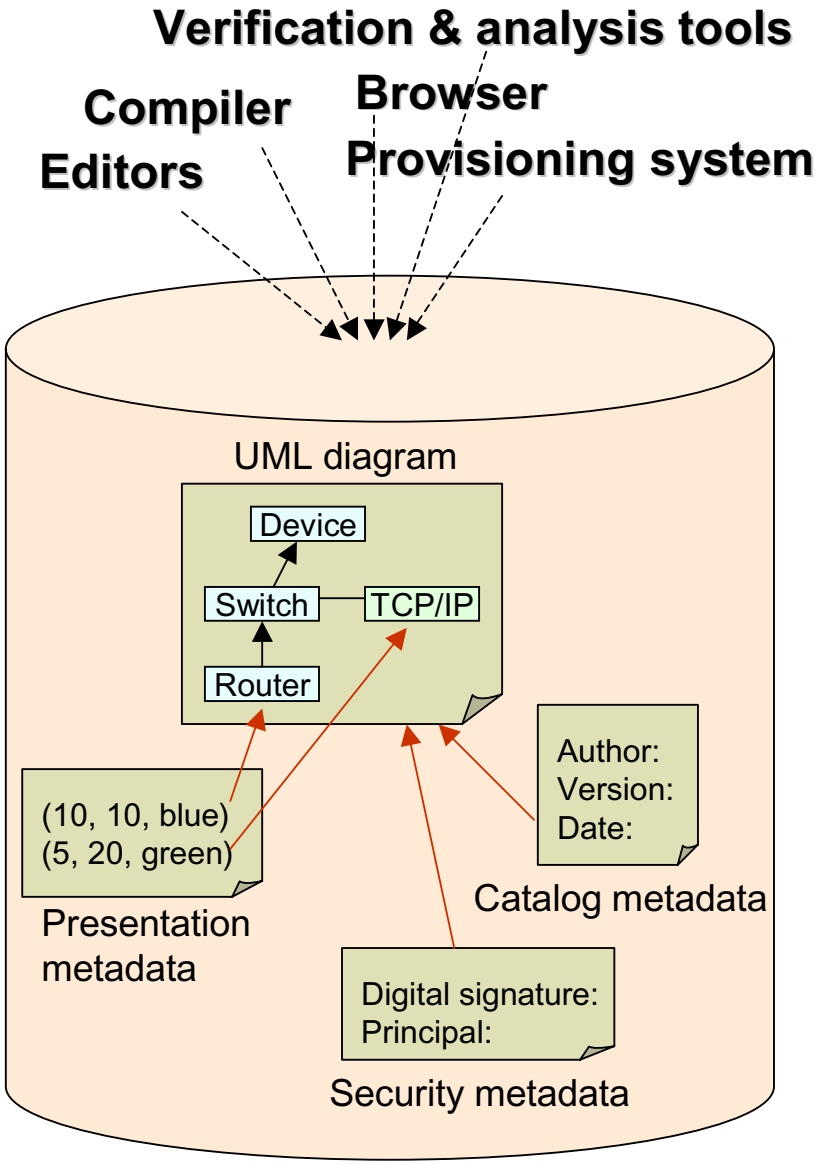
- Intuition: similar objects have similar context
- Basic formula: $\sigma^{i+1}=normalize(\sigma^i+\varphi(\sigma^i))$, with similarity vector σ^i, iteration i
- Corresponds to eigenvector computation $\sigma^{i+1} = \lambda^i \mathbf{M} \sigma^i$
- Filtering of results exploits stable marriage property
- Accuracy metric: $1 - \frac{|wrong|+|missing|}{|correct|} = Recall\left(2 - \frac{1}{Precision}\right)$
- S. Melnik, H. Garcia-Molina, E. Rahm: *"Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching"*, ICDE 2002 (best student paper award)

GEMMYS: A Generic Model Management System



Industrial setting: Ontologent, Inc.

- Goal:
  - Study deployment aspects of GMM
- Product:
  - automated provisioning and support of telecommunication devices and services
- Challenges:
  - Multitude of equipment vendors
  - Devices with incompatible interfaces and varying capabilities
- Approach:
  - Represent device specifications in machine-readable form
  - Manage all metadata uniformly
  - Uses industry-tailored variant of GEMMYS



Ongoing work

- Model “Shuffler”:
  - Simulates evolutionary changes of models
  - Helps clarify semantics of operators
  - Used for quantifying benefits of GMM in typical scenarios
- Scenarios:
  - Model evolution
  - Schema integration
  - Data translation
  - 3-way merge (reintegration)
  - Reverse engineering
- Open questions:
  - Manipulation of complex mappings (SQL views, XSLT, scripts)
  - Ordered relationships in models
  - Instance data transformation
  - DB backend operator execution, optimization
  - Impedance mismatch, GUI support