Applying Model Management to Classical Meta Data Problems

Paper by: Philip A. Bernstein
2003 CIDR

Paper presentation : Anoop
Paper discussion lead: Mike

Outline

- Motivation
- Basic Terminology
- Discussion 1
- Model management operators
- Application in schema integration
- Discussion 2

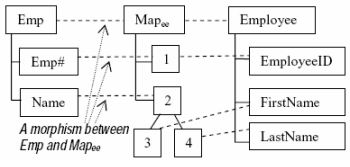
Motivation

- Design, integration and maintenance of application artifacts involves meta data manipulation
- Current method – map the models to an object oriented representation and manipulate using object-at-a-time primitives
- Instead, treat models and mappings as abstractions – use model-at-a-time and mapping-at-a-time operators to improve performance

Basic terminology

- Models
  - Set of objects and relationships between them that effectively model an application artifact e.g. DB schema
- Morphism
  - One-to-one mapping between two objects in two models
- Mapping
  - Between two models is also a model that has morphisms with each of the 2 models that undergoes mapping

Models, objects, morphisms, mapping



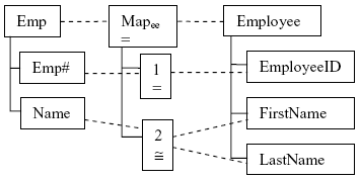
D1 (open) – Main Contributions?

- brand spanking new approach to meta data management
- old approach, with some added value (e.g. extension of operators)
- unified description of operators under a common framework
- detailed and formal descriptions of the operators
- algorithms/means of computing the operators
- proposed system with high level interface to model management
- application of model management to meta data management problems

Operators

- Match – similarities between 2 models
  - Input – 2 models
  - Output – Mapping between models identifying equal or similar objects in models
  - Elementary and Complex match operators
    - Elementary – simple definitions of equality
    - Complex – be able to identify exact matches and similar matches maybe using semantic knowledge

Complex Match Operator

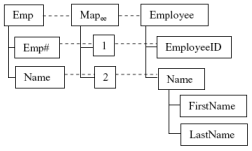


Operators (2)

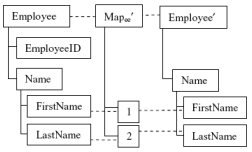
- Diff – difference between 2 models M_1 & M_2
  - Input – a model M_1 and a mapping map_1 which is result of match of M_1 and M_2
  - Output – a model M_1' with objects of M_1 that are not referenced in the mapping map_1 and a mapping to M_1 that distinguishes support objects
  - Support objects – objects to provide model structural integrity

Diff operator

Diff(Employee, Map$_{ee}$)



<Employee', Map$_{ee}$'>



Operators (3)

- Merge
  - Input – 2 models to merge and a mapping
  - Output – Model with all objects in input models with objects that are equal collapsed into a single object and mappings between merged model and input models
- Compose
  - Creates a mapping by combining two other mappings – e.g. M_1 map_1 M_2 and M_2 map_2 M_3 composition of map_1 and map_2 is map_3 between M_1 and M_3 $(map_3(M_1) \equiv map_2(map_1(M_1)))$

Operators (4)

- Apply – takes a model and a function f as input and applies f to each object in model
- Copy – takes a model as input and returns a copy of it
  - DeepCopy – copies model and associated mapping
- ModelGen - takes a model A, and returns a model B based on A (typically B's data model would be different than A's) and a mapping between the two
- Enumerate - returns objects in model one at a time

Application in Schema Integration

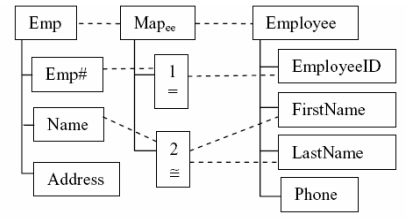
- Problem
  - Suppose we have two databases with different schemas, S_1 and S_2, and we want to create an integrated schema S_3 , as well as the mapping between S_1 and S_3, and S_2 and S_3

Step 1

- First, identify overlapping information in S_1 and S_2
- To achieve this, use Match operator to create a mapping between the two (via Complex matching, since they are likely to be independently developed schemas)

$map_{12} = Match(S_1,S_2)$

Match Result:

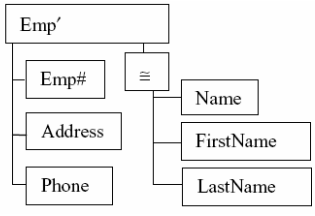


Step 2

- Use the identified overlaps to merge S_1 and S_2
- To achieve this, we use Merge on S_1, S_2 and map_{12} to get the integrated schema and desired mappings

$<S_3,map_{13},map_{23}> = Merge(S_1,S_2,map_{12})$

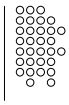
Merge Result:



Step 3

- We have to deal with conflicts in the mapping (i.e. S_1 and S_2 having the same information, but represented differently)
- The conflicting objects are rooted under the object labelled by the ≅ symbol
- The object ≅ is a place holder for an expression property that relates conflicting objects and resolves the conflict
- Thus we have an integrated schema S_3 with its mappings to S_1 and S_2

D2 (idea injection) Feasibility



- is it feasible? why or why not?
- is a subset feasible?
- is it feasible for certain applications/circumstances?