Applying Model Management to Classical Metal Data Problems

Paper by: Philip A. Bernstein

Dramatis Personae

- Paper Presenter – Billy Cheung
- Discussion Leader – Peng Li

Outline

- Discussion #1
  - Introduction
    - Models and Mappings
    - Brief examination into operators
  - In-depth operator examination (Match)
- Discussion #2
  - Model Management in action: Integration
  - Existing Work

Discussion #1

Do you, regardless of your database background, think this is feasible? why or why not?

- "The idea (model management) might sound like pie-in-the-sky that is hopeless to achieve"

  - A Vision for Management of Complex Models

  - For example, it sounds extremely hard to develop a generic algorithm that finds the best match of two distinct models.

Basic terminology

- Models: Formal descriptions for an information system
- Mappings: Description of how two models are related

Problem Scope

- Mapping between a database schema and its next release to guide data migration or view evolution
- Mapping between data sources and a mediated schema to drive heterogeneous data integration
- More on 1st page of paper
What is model management?

- Meta data management
- Given so many different types of data formats, needs to be ways to mould them efficiently so they play nice together
- Uses its own set of operators to try and achieve **model-at-a-time** and **mapping-at-a-time** programming
- Designed to be generic, so single implementation applicable to all data models

What's so special about it?

- Currently, applications translate given models into an object-oriented representation (i.e w/ variables and attributes) and then manipulate the models and mappings in that representation.
- Traditionally, **object-at-a-time** programming used, since there are no support from database query languages
- Main idea behind model management is that “such object-at-a-time programming can be abstracted as high-level operations on models (i.e., schemas) and mappings between models” in order to improve performance

Operators (brief)

- Match: takes two models as input, return mapping between them
- Compose: takes mapping between models A and B and mapping between B and C, returns mapping between A and C
- Diff: takes model A and mapping between A and a model B, returns sub-model of A that is not in mapping

Operators (perhaps not so brief)

- 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
- Merge: takes two models A and B and a mapping between them, and returns a union C of A and B along with mappings between C and A, and C and B

Operators (Yep, not brief a tall)

- Apply: Given a model S and some function f, then it will apply f to every object in the model. (Note: This is an example of object-at-a-time). Very often, however, it will just apply it to the actual model itself.
- Copy: Takes a model as input, and returns a copy of that model.
  - Deep Copy-Copies both model and mapping.
- Enumerate: Takes in a model as input, and returns a cursor to an element in the model. Can call Next operator to iterate through model

In-depth examination of operator

- Two types of matching:
  - Elementary Match: Simple definitions of equality
  - Complex Match: Be able to identify between exact matches and similar matches (possibly require semantic knowledge)
Simple example for complex matching

Example: Complex matching

- Given a model Emp and a model Employee, the resulting mapping $Map_{ee}$ has both exact matching (Emp# to EmployeeID) and similar matching (Name to FirstName and LastName).
- It is already obvious in this simple example that some sort of semantic knowledge is probably necessary in order for successful mapping beyond trivial models.
- Things such as domain-specific thesauri, machine learning and data mining techniques to identify similarity of data instances and other tools will be essential for this.

Discussion #2

Please identify the easiest and most difficult operator to be realized from the following choices. And why?

1. Match
2. Diff
3. Merge
4. Compose
5. Apply
6. Copy
7. ModelGen
8. Enumerate

Application Scenario:

- Examine Model Management (and its operators) in action, in the form of a Schema Integration problem.
- Very common situation, such as polling data from different sources and then integrating them in order to form some conclusion or another.

Problem:

- Suppose we have two databases with different schemas, $S_1$ and $S_2$, and we want to create a 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).

\[ map_{ee} = \text{Match}(S_1, S_2) \]
Step 2
- Using the identified overlaps in order to merge S1 and S2
- To achieve this, we call Merge on S1, S2, and map12 to get the desired mapping

MATCH RESULT

MERGE RESULT

EXISTING WORK
- MMM - for interoperable modelling of economical processes [http://mmm.wiwi.hu-berlin.de/mmm](http://mmm.wiwi.hu-berlin.de/mmm)
- SLP-IOR: Ongoing Development of a Model Management System for Stochastic Linear Programming [http://www.research-projects.unizh.ch/p70.htm](http://www.research-projects.unizh.ch/p70.htm)