MDM
Management of Multiple Models in an Extensible Database Design Tool

Paolo Atzeni, Riccardo Torlone
Proc of EDBT 1996, pp. 79-95

Overview

- Goal
  - a model-independent data dictionary
  - a component of an integrated (flexible, open) CASE tool
- Motivation
  - many data models exist

Many models

- with different features and goals
  - semantic models and logical models:
    - E-R, functional, (conceptual) object
    - relational, network, object
  - general purpose models (for all seasons) and problem oriented models (for specific contexts: DW, statistical, spatial, temporal)
- Variations of models
  - models with different levels of abstraction
  - versions within a family …

The E-R model

- It is not really a model, but a family of models:
  - choose n books (or methodologies or tools);
  - each will claim that it adopts the Entity-Relationship model;
  - you will find m > n versions of the E-R model
- Indeed:
  - Binary vs N-ary
  - With or without attributes for relationships
  - With or without external identification
  - With or without generalizations (total vs partial, overlapping vs disjoint, …)

Why should we handle different models

- a methodology is chosen and a independent tool is available, and their models differ:
  - each CASE tool uses a different model
  - each methodology uses a different model
  - models in tools and methodologies are often not related (the "impedance mismatch")
- designers of a complex project work with their favorite models, but their work has to be exchanged, reused and integrated;
- specific sub-problems are handled with different models, specific for each
- the results of independent design activities have to be integrated

And ...

- the need to exchange data

An ideal goal

- An environment that:
  - allows the definition of any possible model
  - given two models M1 and M2, and a scheme S1 of M1 (the **source** scheme and model), generates a scheme S2 of M2 (the **target** scheme and model), corresponding (**equivalent**) to S1

Is the goal realistic?

- what does “any possible model" mean?
- what does “corresponding" (or “equivalent") mean?

What does “any possible model" mean?

- each model has its own constructs
- each model gives the definition (the semantics) of the constructs in a different way
- each model introduces specific features that have no counterpart in other models

Could there be a “universal" description of models?

- first order logic?
- set theory?

- maybe, but how could we handle the descriptions?

What does “equivalent” mean?

- Various notions of equivalence have been proposed, different from one another
- Given a notion of equivalence, there are cases where, fixed the models and the source scheme, there is no equivalent scheme in the target model; example:
  - the source scheme is an E-R model with cardinality constraints and the target E-R model without them
- Also there are cases where there are two or more “corresponding‘‘ target schemes; example:
  - the source scheme is an E-R model with is-a relationships and the target an E-R model without them

However, the situation is not that bad

- The constructs in the various models are rather similar: they can be classified into a small number of categories (“metaconstructs")
- That is:
  a metamodel approach

E-R model (a version)

- entity: objects in the domain of interest
- relationship: pairs (or n-tuples) of entities
- domain: set of values
- attribute : function from entities (or relationships) to domains

Relational model

- domain : set of values
- relation : subset of the cartesian product of domains (n-tuples) of values

Functional model (a version)

- domain : set of values
- object in the domain of interest
- function : from objects to objects or domains

A classification (Hull & King, 1987)

- Lexical types : sets of printable values
  - Domain
- Abstract types
  - Entity type , set of objects in the world
  - Class , set of objects in the system
- Aggregation : a construction based on (subsets of) cartesian products
  - Relationship in the E-R model
  - Relation in the relational model
- Function
  - Attribute in the E-R model
  - Function in a functional data model
- Grouping
- Hierarchies

Summarizing

- We can fix a set of metaconstructs of interest (each with a set of possible variants):
  - lexical, abstract, aggregation, function, ...
  - the set can be extended if needed, but this will not be frequent
- Then a model can be defined in terms of the metaconstructs its constructs refer to

Some models

- The relational model
  - a lexical construct, called domain
  - an (n-ary) aggregation construct, called relation
- (A simple version of) the E-R model
  - a lexical construct, called domain
  - an abstract construct, called entity
  - an (n-ary) aggregation construct, called relationship
  - a (monovalued monadic) function construct (from an entity or a relationship to a domain)

The Supermodel

- A model that includes all the constructs (in their most general forms)
  - each scheme for any model is also a scheme for the supermodel (up to renaming)
  - translations can be performed within the supermodel
  - each translation from the supermodel SM to a target model M eliminates all constructs that are not allowed M
  - therefore, each translation from SM to M is also a (possibly redundant) translation from any other model to M

Translations in this framework

- The constructs corresponding to the same metaconstruct (e.g. entity in the E-R model and class in an object model both corresponding to abstract) have the same "meaning"
- Translations can refer to metaconstructs, rather than to constructs (which are model specific)
- A translation from a source model to a target model would have to replace constructs in the source (and not in the target) with constructs in the target
- Translations can be built by composing elementary transformations
  - each of them would eliminate some constructs (or “patterns” thereof) and possibly introduce new ones
  - a translation from a source model to a target one would eliminate some constructs and introduce new ones

Elementary steps

- There can be a set of predefined basic translations
- They are assumed to be correct
- We have studied properties of compositions of basic translations:
  - a correct translation from M1 and M2 produces schemes that contain only constructs that are allowed in M2

Examples of basic translations

- eliminate n-ary aggregations; replace them with binary ones (and abstracts)
- eliminate binary aggregations; replace them with functions
- eliminate functions to abstracts; replace them with aggregations
- eliminate complex attributes; replace them with simple attributes and abstracts

Translations, how many?

- If we have n different models, how many translations do we need?
  - apparently, n^2
  - in fact, only n, that is, one for each model, which eliminates the constructs that are not allowed

Actors on the scene (and behind it)

- designers : define schemes within existing models
- model engineers : define models by using metaconstructs and generate (and modify) translations by composing basic translations
- metamodel engineers: extend the whole system, by defining new metaconstructs and the corresponding basic translations (a nontrivial task)