Metadata Repositories as Infrastructure for Database Oriented Applications

Stefan Jablonski, Ilia Petrov, Christian Meiler, Udo Mayer

Chair for Database Systems, Institute for Computer Science VI, Friedrlich-Alexander-Univeristy of Erlangen-Nuernberg, Martnesstr. 3, 91058 Erlangen Germany
{Jablonski, Petrov, Meiler, Mayer}@informatik.uni-erlangen.de
http://www6.informatik.uni-erlangen.de

Abstract. We argue that adaptability and extensibility can be guaranteed by the utilization of the concepts meta-modeling, formation of concepts and reflection, which require the use of repositories. We present an approach, in which domain knowledge and meta-framework are combined to produce extended meta-schema, to facilitate application design. Extended meta-schemata trigger a shift to repository systems. We discuss how reflection techniques can be used to create meta-aware applications.

1 Introduction

In this paper we investigate database applications concentrating on development and maintenance of those applications. Our goal is to provide sophisticated infrastructure to facilitate these issues. We investigate the advantages of meta-modeling and how method oriented meta-models can be enriched with domain-specific knowledge to create domain-specific building blocks, which eases the creation of information models and lead to reduction of complexity. Additionally we study how reflection techniques can be used to benefit from the rich meta-information at run-time. Reflection assures a degree of decoupling of the application logic form certain specifics of the information model [8]. Based on these observations we motivate the transition from database to repository based applications. Database applications involve lots of data manipulation and processing activities, which implies powerful GUI and intensive user interaction. Traditional three-tier [14] web applications typically undergo many changes in the user interface and in the application logic. Therefore, adaptability and extensibility are typical properties database applications must exhibit on the long run. These properties require a change in the application design style - a shift to open architectures whose key characteristic is the extensive use of meta-layers and reflection [8].

2 Related Work

2.1 Modeling and Formation of Concepts

Let us now move our focus onto modeling methods since they are instrumental for conceptual design of database applications. We will take a closer look on two orthogonal methods within the broad realm of modeling: formation of concepts and meta-modeling.

Formation of concepts is a notion describing operations leading to the construction of new concepts. The term 'concept' can be loosely defined as a type or some form of composition of types, whereas formation of concepts is the process of definition (reconstruction)

of new concepts. A clear distinction between the terms 'formation of concepts' and 'abstraction' must be made. Abstraction is the process of generating a new concept, while concealing its internals [9], [11]. Abstraction has to do with layering and information hiding. Examples for abstraction in RDBMS are the ANSI/SPARC architecture on the one hand and views on the other hand [9]. 'Formation of concepts' is a term more general than abstraction. Formation of concepts involves are specialization/generalization (subordination), the composition relationship, and subsumption. These notions are present in any modern modeling language (UML, ERM etc.) and can be mapped onto implementation languages (OO languages, object relational model) without significant loss of expressiveness. Subsumption refers to gathering items under concepts and presenting them as instances of a type [3]. Composition [3] relates concepts in order to form a new concept.

2.2 Meta-Framework

Formation of concepts does not provide possibilities to describe the structure of the information model. Description of the structure has already been proven to be very beneficial in the field of XML where the information model elements used in an XML document to structure the data are firstly being defined in the XML DTD or XML Schema document. Such description is important in two ways: firstly define very precise information model building blocks and use them to define the information model; secondly document the structure of the information model in a explicit manner which would provide enough (runtime) information so that any program can process it (every program which can process XML DTD or XML Schema documents could process an XML document – however not "understand" it).

From a meta point of view any system is organized as a stack of layers [5] [4]. Each layer is called a meta-level (denoted M0, M1, M2, M3) and is associated with a respective model (schema) e.g. model, meta-model, and meta-meta-model (cf. Fig. 1). The model on the layer (or language level) M0 is called an information model [13], which is a synonym for "conceptual model", UML application model etc [12]. The XML Document form the example above lays on the M0 level whereas the DTD or the XML Schema definition lays on M1.

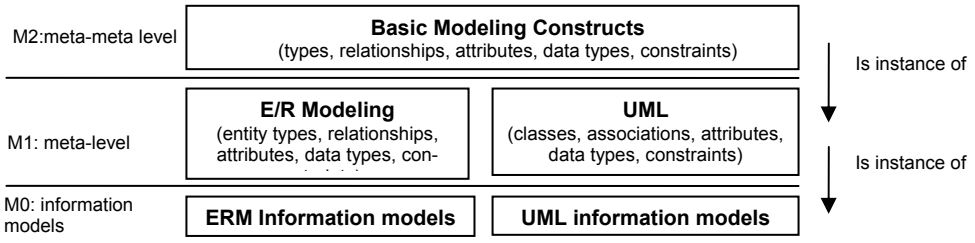

Fig. 1. Modeling Constructs and Language Levels

The process in which a higher level meta-model is constructed to describe the models on the underlying level is often called reconstruction. Reconstruction can be applied iteratively, which leads to a hierarchy of language levels cf. Fig. 1. Each data model regardless of which concrete modeling approach has been employed is built using types, relationships, attributes, data types, constraints. Relationships connect types; various characteristics of types are described by attributes. Attributes have data types and constraints control the allowable characteristics of an attribute.

3 Extended Meta-Schemata

The terms ‘meta’ and ‘formation of concepts’ are orthogonal. The implications of the term ‘meta’ are: change of meta-levels and explicit description of the structure of the the underlying level. Formation of concepts is associated with the creation of domain specific knowledge; however, as stated in section 2.1, its use is constrained to a single level.

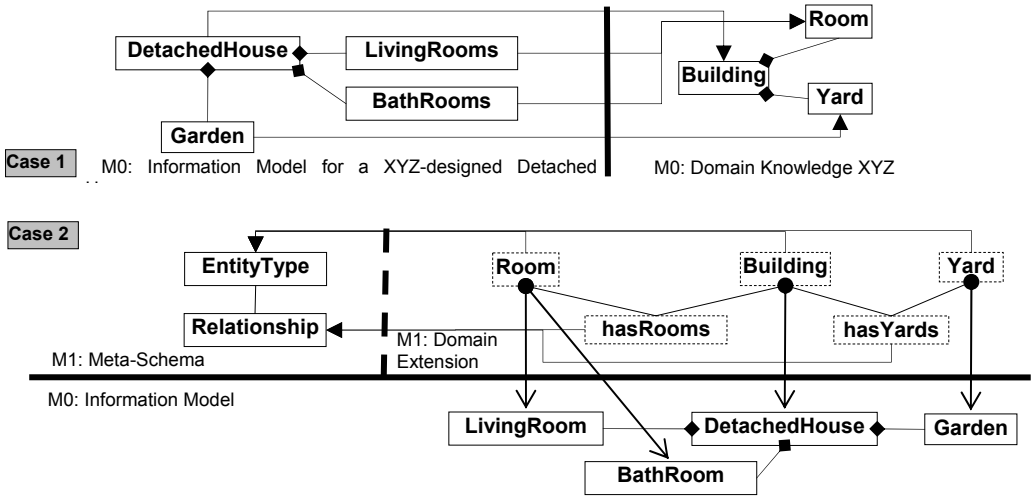

Fig. 2. Information model and Extended Meta-model

Parts of the domain knowledge (on level Mk) are independent on the respective Mk+1 schema. Such parts result from iterative abstraction and are of general validity for a concrete domain. Consider the following example - company XYZ is an architectural agency, which designing “Buildings”, consisting of “Rooms” and a “Yard” (see Fig. 2, Case1). Using only concept formation of all projects XYZ develops can be stored in a schema derived form the domain knowledge as shown in Fig. 2, Case 1. As the example shows the domain knowledge shapes the respective model since every building type is to be derived from “Building”. However it is still a part of the information model which leads to limited reuse and local scope of such knowledge. Hence the idea of transferring parts of the domain knowledge modeled on level Mk to level Mk+1. The result is meta-model containing structural information with tailored domain specific concepts are (extended meta-model) - Fig. 2, Case 2. These domain concepts can then be used as building blocks to underlying Mk models. Although the domain knowledge transfer can be applied between any two adjacent levels, it brings concrete benefits when applied between M0 and M1.

Referring to our example with company XYZ - “Buildings” “Rooms” and “Yards” can be modeled as subclasses of the class “EntityType” (or the class “Class” when using UML) on M1 and therefore used as building blocks for XYZ specific models on M0 (see Fig. 2). The M0 level concepts “LivingRoom”, “Bathroom”, “Garden” and “FamilyHouse” are instances of the M1 concepts “Room”, “Yard” and “Building” respectively (See Fig. 2). The relationships between the M0 concepts “LivingRoom”, “BathRoom” and “DetachedHouse” are instances of the M1 relationship type “hasRooms” and so on. Attributes are left out for reasons of simplicity, but similar reasoning applies to them as well. The practical result is that any information model using the extended meta-model can utilize both extended M1 concepts (like “Building”) as building blocks and the semantics (e.g. all instances of “Building” must be connected with instances of “Rooms”) as modeling rules.

4 Repositories

The idea proposed in sections 2.1, 2.2, 2.3 is based on the fact that meta-models are not static (read-only) rather they can be extended with the domain knowledge. To implement it we need a database which would allow support changes its meta-catalogue. The authors are not aware of any existing database system with such characteristics. Repositories, however, are a good candidate for a target system. There are a number of definitions of what a repository system is cf. [1], [3]. A repository acts much like a database; however it supports modifiable M1 level models. Therefore the extended meta-models can be deployed in the repository utilized further. A repository system implements the meta-level stack discussed in section 2.2 and takes the burden of preserving the consistency between extended meta-model and M0 models i.e. the repository enforces automatically what the extended meta-model defines in terms modeling building blocks and structure onto the M0 models (structural and meta-terminological control). Repositories must support rich reflection mechanisms. Such mechanisms are more powerful than the 'conventional' reflection APIs like RTTI or Java Reflection [7]. Repositories should also offer handy reification mechanisms [6]. This is a challenging task in light of the fact that many application programs written in different languages can use the repository. Both the reflection and reification mechanisms are part of the repository interface. Its importance is discussed in [13].

5 Conclusions

In this paper we proposed a new approach that facilitates the development of applications based on a combination of formation of concepts and meta. The proposed approach contributes to semantically correct model, reuse, and reduction of complexity. We motivate the use of repository systems as successor of database systems as target of the extended meta-schemata.

References

1. Philip Bernstein, Umeshwar Dayal: An overview of repository technology. Proceedings of the 20th VLDB Conference Santiago Chile, 1998
2. R. Elmasri and S. B. Navathe: Fundamentals of Database Systems. Benjamin/Cummings, 2001.
3. Erich Ortner: Aspekte einer Konstruktionssprache für den Datenbankentwurf, Darmstadt 1983
4. Erich Ortner: Wissensmanagement Teil 1, 2 Informatik-Spektrum, Volume 23 Issue 2 pp 100-108
5. Object Management Group: Meta Object Facility Specification Version 1.4.
6. Krzysztof Czarnecki, Ulrich W. Eisenecker: Generative Programming., Addison-Weseley, 2000
7. Dale Green: Java Reflection API. Java Tutorial.
8. Frank Buschmann: Pattern-orientierte Softwarearchtecktur, Adison-Wesley, 1998
9. John Miles Simth, Diane C.P. Smith: Database Abstractions: Aggregation and Generalization, ACM Transactions on Dstabsse Systems, Vol. 2, No. 2, June 1977, Page 103-133.
10. Sheetal V. Kakkad, Mark S. Johnstone, and Paul R. Wilson: Portable Run-Time Type Description for Conventional Compilers, USENIX '97, 1997.
11. Jürgen Mitelstraß: Enzyklopädie Philosophie und Wissenschaftstheorie, B.I.-Wissensverlag, 1980
12. Wedekind H., Schlundt M., Neeb J. "Repositories for Workflow Management Systems in a Middleware Environment", 33rd Hawaii International Conference on System Sciences, 2000
13. Bernstein P.A. "Repositories and Object Oriented Databases." SIGMOD Record 27 p.88-96, 1998
14. Inderjeet Singh et al. : Designing Enterprise Applications with the J2EE Platform, Adison-Wesley, 2002