Metamodel-based Comparison of Data Models

Erki Eessaar
Department of Informatics, Tallinn University of Technology, Raja 15, 12618 Tallinn, ESTONIA
eessaar@staff.ttu.ee

Abstract—A data model specifies the building blocks of databases, the rules how to assemble these blocks and operations that can be performed based on the built-up structures. We have to create increasingly complex applications. Properties of the underlying data model of a Database System (DBMS) determine how easy it is to create an application that uses a database. There are many different data models. We have to choose a DBMS, the underlying data model of which best fulfills the needs of an application. Existing comparisons of data models are mostly based on the experiences of using one or another DBMS. This paper explains how to perform non-empirical comparison of data models by using the metamodels, which describe abstract syntax of these data models. We also present some results of the comparison of the underlying data model of SQL:2003 and the data model that is proposed in The Third Manifesto.

I. INTRODUCTION

The concept “data model” has two different meanings:

- Meaning 1: “An abstract, self-contained, logical definition of the data structures, data operators, and so forth, that together make up the abstract machine with which users interact.” [1]

In this work, we use the concept “data model” in the sense of meaning 1.

Examples of data models are hierarchical, network, relational, object-relational, TransRelational and object-oriented. Some of them are general names because there are different proposals about the exact nature of these models. Examples of proposals about object-relational data model are [2], [3], [4], [5]. In addition, SQL:2003 [6] follows object-relational paradigm [7] and its underlying data model is also one interpretation of object-relational data model.

Applications that use databases become increasingly complex and they demand more and more from the DBMSs. A very important selection criterion of a DBMS is its properties. How should we compare data models? The work [8] is an example of thorough and methodical comparison of two data models. It presents similarities and differences of relational and network data model in the form of discussion and examples. The authors even had to work out definitions of concepts of the network data model based on CODASYL DBTG language proposals in order to do it properly. Additional examples are the comparison of the prescriptions, proscriptions and suggestions of The Third Manifesto with SQL and with ODMG proposal of object model and associated database language [4].

However, many comparisons or judgments of the data models that are presented in the literature are based on the experiences and the intuitive understandings of the researchers and developers. One reason of the prevalence of informal descriptions and empirical observations is that there is often no precise specification of a particular data model. Instead, there is a set of research papers and textbooks that reflect their authors understanding of the model and a set of DBMSs that implement the model with their own limitations and extensions. Inadequacies and shortcomings of the DBMSs as well as lack of understanding what are the parts of a data model can cause unfair criticism of a data model. For example, research [9] shows that some of the criticism towards relational data model is caused by the exactly these reasons. A more precise method for evaluating data models is needed.

Metamodeling is well-known activity in software engineering. Metamodel “makes statements about what can be expressed in the valid models of a certain modeling language.” [10] A data model is also kind of abstract language.

The goal of this paper is to present a non-empirical metamodel-based method for comparing data models. We also present some results of comparison of the underlying data model of SQL:2003 standard (“OR SQL”) [6] and the data model (“OR TM”) that is described in The Third Manifesto [5].

According to The Third Manifesto, all the good features that are expected from object-relational data model can actually be implemented within the framework of the relational model. In particular, the support to complex data types is already present in the relational model in the form of domains [1].

The rest of the paper is organized as follows. Section 2 gives an overview of important concepts and presents a possible improvement of CIM Database Model. Section 3 discusses different comparison methods of data models and proposes a new method. Section 4 presents some results of the metamodel-based comparison of two data models (OR SQL and OR TM). Finally, we draw some conclusions.

Participation in the conference was supported by the Doctoral School in ICT of Measure 1.1 of the Estonian NDP.

II. IMPORTANT CONCEPTS OF DATA MODELS

CIM (Common Information Model) Database Model [11] is a conceptual model that describes common database management concepts. However, it models data model only as an experimental property DataModelType of class CommonDatabase.
We think that it is necessary to model this concept more precisely and present the domain model (see Fig. 1). The classes with grey background are already present in CIM Database Model. The new classes are with white background.

A programming language is a formal language designed specifically for machine processing [12]. A data model is a kind of an abstract programming language [1] that specifies the data structures and operators, which are its structural and behavioural components, respectively (see Fig. 1). In addition, a data model specifies “a collection of general integrity rules, which implicitly or explicitly define the set of consistent database states or changes of state or both” [13].

A specification of a formal language, like modeling or programming language, must contain specifications of abstract syntax, semantics and concrete- and serialization syntaxes [12]. A Database System (DBMS) is a software system used for managing databases. A user can interact with it by using a database programming language that is designed according to some data model. The data model is the basis for the abstract syntax of this language. A database programming language has two sublanguages – a Data Definition Language (DDL) and a Data Manipulation Language. Statements of a DDL are used in order to create data types, structures, operators and integrity rules. Statements of a DML are used in order to perform operations with data.

A database can be divided into conceptual, external and internal levels according to ANSI/SPARC architecture [1]. Ideally, a data model specifies elements that belong to the logical levels - conceptual and external level (and not elements at the internal level). In addition, a DBMS should provide a Storage Structure Definition Language (SSDL) for managing storage structures at the internal level [4]. In practice, there is often no separate SSDL. Instead, it is possible to specify elements of the internal level (indexes, tablespaces, clusters, segments, files etc.) and other properties of data storage by using DDL statements.

Database programming languages provide also features that are independent of a data model. The existence of these orthogonal features does not depend on the underlying data model of a database language and they could be present in many languages that have different underlying models. Examples of these orthogonal features are the support to the nested transactions [4] or security mechanisms (for example, a possibility to specify roles, users and their privileges). A data model can have more than one corresponding database programming languages. Different languages could provide support to different orthogonal features. For example, The Third Manifesto that is a proposal for future database systems uses the language name “D” in order to refer to any language that follows its principles. The manifest book also presents Tutorial D language that is “a computationally complete programming language with fully integrated database functionality” [4]. Nevertheless, the authors acknowledge that their proposed language is a “toy” language that must support learning. Industrial-strength languages would need additional features.

If we want to compare data models and reason about them, then we must have their specifications at our disposal. The relational model is an example of the data model that was formally specified before the appearance of systems that implemented it [13]. Sometimes a data model is formally specified only after its implementations (DBMSs) have been created. This is for example true in case of hierarchic and network data models [13].

Nowadays object-relational data models are of major interest. SQL:1999 and SQL:2003 standards specify the object-relational database programming language. However, these specifications do not contain a clear and compact