An Approach to Generating Specifications of a Database Schema for DBMSs based on XML Data Model

Milica Poznanović, Slavica Kordić, Jovana Vidaković, Sonja Ristić, Milan Čeliković

University of Novi Sad,
Faculty of Technical Sciences,
Trg D. Obradovića 6, 21000 Novi Sad, Serbia
Email: mpozanovic@yahoo.com, {slavica, jovanana, sdristic, milancel}@uns.ac.rs

Abstract—In the last few decades, there have been a lot of different approaches to the information system (IS) design and development. They are based on different data models. The selection of a data model for conceptual design depends on the problem domain, the knowledge and the personal preferences of an information system designer. The approaches based on the Extended Entity-Relationship (EER) data model are broadly accepted throughout the community of IS designers. In the recent years the eXtensible Markup Language (XML) has spread to many application fields and it is being widely adopted as a standard language for data representation, exchange and storage. As the number of possible usages of XML grows, so does the need of easy management of XML data sources and their integration. The XML documents represent real-world aspects defined by a conceptual schema expressed by means of a selected data model. Having different model representations is an advantage for the flexibility they offer to the user in selecting the right representation based on their requirements. On the other side, model-based heterogeneity raises challenges for schema interoperability. In the paper we use model-driven approach to address the problem of interoperability between an EER database schema and XML schema specification. The presented transformation of a schema expressed by concepts of EER model to an equivalent schema expressed by means of XML data model is integrated into Multi-Paradigm Information System Modeling Tool (MIST) that provides a model driven approach to information system development and evolution.

I. INTRODUCTION

The emergence of large, complex and integrated information systems that are interoperable in highly changeable environment increases the interest for the design and implementation approaches that successfully address the system complexity and evolution. Model-driven system engineering (MDSE) and Model Driven Software Development (MDSD) applied in the context of enterprise information system design, implementation, maintenance and evolution involves the automatic generation of information system artifacts from structure and functional models. MDSE and MDSD address IS complexity through abstraction and increase the importance and power of models. A model-driven approach allows the developer to work at a higher level of abstraction, with concepts and structures that are closer to the end-users and their perception. Models are used to specify, simulate, test, verify and generate code for the application to be built.

In our previous research we have developed a Multi-Paradigm Information System Modeling Tool (MIST) that provides a model driven approach to information system development and evolution. MIST is aimed to provide an evolutive and incremental approach to IS development, which is based on the form type data model (FT) and the Extended Entity-Relationship model (EER). The approach is purely platform independent and strictly differentiates between the specification of a system and its implementation on a particular platform. A detailed description of the FT data model and FT approach may be found in [1]. A detailed description of the EER data model, domain specific language (DSL) aimed at EER model specification (EERDSL) and EER approach implemented in MIST tool may be found in [2].

Conceptual design is a decisive step for the successful design and development of an IS. Conceptual models can capture the concepts and requirements that are close to users' perception and in a way they can bridge the gap between end-users and designers. In the past decades, EER data model, aimed at conceptual database design, has been broadly accepted throughout the community of IS designers. On the other side, changes in the demands for the data processing have caused emergence of new data storage, retrieval and processing mechanisms. A lot of data sources store unstructured or semi-structured data. Storing and processing such data are beyond the capabilities of traditional relational database management systems (RDBMS). NoSQL database systems have built-in mechanisms to save data in various formats such as: XML, JavaScript Object Notation (JSON) or Binary JSON (BSON). There are several types of NoSQL databases. In the paper we present the extension of our MIST aimed at support of document-oriented NoSQL database type.

Document-oriented databases store data as a collection of XML documents. XML is being widely adopted as a standard language for data representation, exchange and storage. The main reasons behind this adoption are its auto-descriptive structure and textual and non-proprietary format. Consequently, the XML documents can be created both by humans and by software. XML Schema is used to describe the structure and the content of XML data. XML Schema was designed as an interchange language and it is used for specifying and validating XML documents. It is not designed as a conceptual modeling language. Therefore, designers still use EER data model to express database schema. It allows a domain expert to model the problem domain...
independently of the implementation (XML e.g.) and then create corresponding XML schemas, which are used to describe a structure of XML documents.

That was the main motivation to extend MIST with new components designed to provide complete support for: generic XML schema generation from a conceptual database schema based on EER data model, vendor specific XQuery script generation from a generic XML schema, and execution of generated XQuery code under a vendor specific database management system (DBMS). XML Generator currently supports creating XML schema scripts for eXist and Sedna DBMSs.

Apart from Introduction and Conclusion, the paper is organized in six sections. In Section 2, we describe the extended architecture of MIST. The XML schema meta-model is presented in Section 3. The transformation of a conceptual database schema based on EER data model into a generic XML schema is presented in Section 4. A case study that illustrates a usage of the meta-models, extended MIST components and XML code generator is set up in Section 5 and related work is discussed in Section 6.

II. THE ARCHITECTURE OF MIST

In this section we present the architecture of MIST. Its global picture is depicted in Fig. 1. MIST comprises the following components: FTDSL, Synthesis, Business Application Generator, EERDSL, EER2Rel, EER2Class, R2FT, SQL Generator, Java Generator, EER2XML, and XML Generator.

Components from the upper part of Fig. 1 are already present in our previously published papers [2, 5, 6, 7, 8, 9, 10]. In this paper we present the components positioned in the rectangle at the bottom of Fig. 1: EER2XML and XML Generator.

Figure 1. The architecture of MIST

EER2XML component of the MIST provides the model-to-model (M2M) transformation of EER database model to generic XML schema specification. The input into the EER2XML component is a conceptual database model which is expressed by the concepts of the EERDSL modeling language. Models being transformed conform to the EER meta-model [2] and XML meta-model, respectively. The output of the EER2XML component, generic XML schema specification, may be further used in the process of XQuery code generation. For this purpose, the XML Generator component is developed. It generates the XML Schema document and Xquery scripts for creating collections of appropriate XML documents. In order to enable implicit validation in specific DBMS, database configuration files must be changed. Below is an example how it can be done for eXist database.

To enable implicit validation, the eXist database configuration must be changed by editing configuration file. All XML Schemas that are used for validation must be registered with eXist using OASIS catalog file, which is supported by our XML Generator. For the code generation we used Xtend tool.

III. XML META-MODEL

In this section, we present in more details XML meta-model. This meta-model represents the abstract syntax of DSL used for specifying data models at the conceptual level. The abstract syntax is implemented in a form of a meta-model that conforms to the Ecore meta-meta-model. The XML meta-model consists of approximately sixty components. The most important components of the XML meta-model are given in Fig. 2 and Fig. 3. In the rest of this section, we describe each of the most important components of the XML meta-model with corresponding meta-model class written in italics inside parentheses.

Meta-model of XML schema basic concepts are depicted in Fig. 2. Schema element (SchemaElement) is the root element of each XML Schema document. It contains the definition of an XML Schema document and it may contain some attributes. Each schema element comprises zero or more elements. Elements can be grouped by their function: top-level elements (GlobalElement) and particles (LocalElement). Top-level elements are elements that appear at the top level of an XML Schema document, as a direct child of a schema element. Particles are elements that always appear as part of complex type definition. As a parent element of the local elements, sequence element (XMLSequence) is used. Sequence element defines ordered list of elements. Each element may have a name attribute, which is the tag name that will appear in the XML document. Local elements, as well as the sequence elements, can have minOccurs and maxOccurs attributes. A minOccurs value can be any number greater than or equals to zero, while maxOccurs can be provided with the same value as the minOccurs, but also no limit can be set on the maximum number, using the value "unbounded".

Meta-model of XML schema constraints are depicted in Fig. 3. Each element comprises one or more constraints inheriting the abstract concept constraint (Constraint), which may have a name attribute. At the top levele of XML schema specification, for each element following constraints may be specified: (i) primary key constraint (Key); (ii) unique constraint (Unique); and (iii) foreign key constraint (KeyRef). Each constraint comprises one and only one constraint content (ConstraintContent). Constraint content contains at least one field element (Field) and one and only one selector element (Selector).

The selector is modeled to specify an XPath expression that selects a set of elements for an identity constraint, while as the field element specifies an XPath expression that specifies the value used to define an identity constraint. Unique and key references inherit the abstract class refer (Refer). This concept is specified in order to allow foreign key constraints to have both, reference to unique and primary key constraint (UniqueRefer and KeyRefer).
Figure 2. Meta-model of XML schema basic concepts

Figure 3. Meta-model of XML schema constraints
IV. TRANSFORMATION OF EER DATABASE MODEL INTO A GENERIC XML SCHEMA

In this section we present EER2XML transformation for transforming an EER database model into a generic XML schema. EER2XML transformation is based on the meta-models. Transformation is specified in ATL transformation language (ATL). In this paper we present conceptual mappings between appropriate concepts from different meta-models without the ATL implementation details. In Table I, columns represent all of the corresponding concepts between the EER and XML meta-models. Based on these correspondences, concrete ATL transformation rules are specified.

<table>
<thead>
<tr>
<th>EER meta-model concepts</th>
<th>XML meta-model concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERModel</td>
<td>DataBase, XMLCollection, XMLSchema, Declaration, SchemaElement, GlobalElement, ComplexType, XMLSequence</td>
</tr>
<tr>
<td>Entity</td>
<td>LocalElement, ComplexType, Key</td>
</tr>
<tr>
<td>Relationship</td>
<td>LocalElement, ComplexType, KeyRef, KeyRefer, Key, Unique, ConstraintContent, Selector, Field</td>
</tr>
<tr>
<td>Attribute</td>
<td>Attribute</td>
</tr>
<tr>
<td>Attribute domain</td>
<td>AttrBuiltType</td>
</tr>
<tr>
<td>Key</td>
<td>Key, ConstraintContent, Selector, Field</td>
</tr>
<tr>
<td>ISA</td>
<td>LocalElement, ComplexType, Key, KeyRef, KeyRefer, ConstraintContent, Selector, Field</td>
</tr>
<tr>
<td>Categorization</td>
<td>LocalElement, ComplexType, KeyRef, KeyRefer, ConstraintContent, Selector, Field</td>
</tr>
<tr>
<td>Categories</td>
<td>LocalElement, ComplexType</td>
</tr>
<tr>
<td>Identification dependency (weak entity)</td>
<td>LocalElement, ComplexType, Key, KeyRef, KeyRefer, ConstraintContent, Selector, Field</td>
</tr>
<tr>
<td>Identification dependency (regular entity)</td>
<td>LocalElement, ComplexType, Key</td>
</tr>
<tr>
<td>Gerund</td>
<td>LocalElement, ComplexType, KeyRef, KeyRefer, Key, Unique, ConstraintContent, Selector, Field</td>
</tr>
<tr>
<td>Domain</td>
<td>AttrBuiltType</td>
</tr>
</tbody>
</table>

Most of the concepts may be transformed directly to XML schema concepts. Local elements are created from eight different concepts of EER data model: Entity, Relationship, ISA, Categorization, Categories, Identification dependency (regular entity), Identification dependency (weak entity) and Gerund. Each regular entity concept is transformed into a local element concept. The set of constraint of the regular entity is transformed into a set of constraints that can be defined at the level of the XML data model. The primary key of the regular entity is transformed into key concept, and the unique constraint is transformed into unique concept.

Attributes of the entities and relationships are transformed into the attributes concept of the XML data model. Based on the domain concept, XML data model built-in types are created.

V. CASE STUDY

In this section we introduce a case study of Pharmacy IS in order to illustrate concepts, data models, transformation, and code generator described in previous sections. We present a part of a Pharmacy database model, named Pharmacymodel. Due to complexity of the example, we present in details only a pharmacy, pharmacy company, and medicine. Pharmacymodel EER data model instance is consisting of:

- three entities: Pharmcomp with attributes IDPHC and NAMEPHC; Medicine with attributes IDMED and NAMEMED; and Pharmacy with three attributes IDPHARM, NAMEPHARM and ADDRESSPHARM;
- one Identification dependency ID; with weak entity Medicine and regular entity Pharmcomp;
- one Gerund Hascontract with Pharmacy and Pharmcomp regular entities in the relationship and attributes DATECONTR and EXPIRDATECONTR;
- and two relationships: Subordinate compromises one regular entity Pharmcomp with two roles: company and subcompany; and Offers with weak entity Medicine and gerund Hascontract in relationship.

For the specification of the database model, we use modeling concepts presented in [5]. In Fig. 4, we present our example using graphical notation of EERDSL, while as Fig. 5 presents the source and target models of EER2XML transformation opened in the Eclipse “Sample Reflective Ecore Model Editor”. In EERDSL all concepts are represented with widely used graphical notation presented by Thalheim [1-4].

It can be seen that EER meta-model Pharmacymodel concept is mapped into the XML DB Pharmamodel. Each of three EER entities has been mapped into a local element with the same name. Here, we present local elements. The local element Medicine contains one complex type, whose subtypes, IDMED and NAMEMED, are attributes created from an entity Medicine in EER model. IDPHC represents a foreign key from local element Pharmcomp. Foreign key is created because of the relationship between entities Medicine and Pharmcomp, with maximum cardinalities one and many. Based on that foreign key, key ref fk1_medicine element is created, that references the primary key of the Pharmcomp local element.

Pharmcomp local element contains one complex type, whose subtypes, IDPHC and NAMEPHC, are attributes created from an entity Pharmcomp. It contains an attribute IDPHC_company, created because of the recursive relationship, where the same entity, Pharmcomp, participate as the parent and child. Also, because of the recursive relationship, key ref fk1_pharmcomp element is created. The local element Pharmacy contains one complex type, whose subtypes, ADDRESSPHARM, IDPHARM and NAMEPHARM, are attributes created from an entity Pharmacy. Hascontract local element contains one complex type, whose subtypes, EXPIRDATECONTR and DATECONTR, are the attributes created from the gerund HASCONTRACT. Also, it contains attributes, IDPHARM and IDPHC, from the regular entities who participate in the relationship of
this gerund, Pharmacy and Pharmcomp. Based on that relationship, key ref elements, fk1_hascontract and fk2_hascontract are created, and they reference the primary key of the Pharmcomp and Pharmacy local elements. The Offers local element contains one complex type, whose subtypes are the attributes that represent the attributes of the primary keys from the entities that participate in the relationship offers, Medicine and Hascontract. Based on this relationship with maximum cardinalities many and many, two key ref elements are created, fk1_offers and fk2_offers.

In Fig. 6, we present how implicit validation is enabled for eXist database using OASIS catalog file, which contains all the XML Schemas that are used for validation, including the one that is the result of our generator, pharmacymodel.xsd. The same is done for Sedna database.
The research presented in this paper was supported by Ministry of Education, Science and Technological Development of Republic of Serbia, Grant OI-174023.

REFERENCES


