AN MDA APPROACH FOR OBJECT-RELATIONAL MAPPING

Cătălin Strîmbei and Marin Fotache
Al.I. Cuza University of Iaşi, Dept. of Business Information Systems
Bd. Carol I, 22, Iasi, 700505 Romania

Keywords: Object-Relational Databases, UML, SQL, Oracle, MDA.

Abstract: This paper reviews several emergent approaches that attempt to capitalize on SQL data “engineering” standard in current <object>-to-<object relational> mapping methodologies. As a particular contribution we will discuss a slightly different OR mapping approach, based on ORDBMS extension mechanisms that allow to publish new data structures as Abstract Data Types (ADT).

1 ARE OBJECT-RELATIONAL DATABASE SYSTEMS SO SUCCESSFUL?

Despite the semantic richness, the huge support and enthusiasm from the academic and scientific community, the Object Oriented (OO) data model has failed to supersede the relational model on the market. This failure could be explained not only by the lack of performance of OODBMSs, but also by the weaker management services, the breaking of the data independence principle which results into diminishing portability between technological application frameworks.

OR databases are not implemented at the scale of relational ones in business applications and Object-to-Object-Relational (O-OR) mapping theory does not seem to be as mature. Also, the number of case studies disseminating O-OR successful projects is quite small, so OR databases are far from fully being exploited. Though, the most obvious strength of O-R databases is the natural representation of complex objects. Business semantics are better implemented using extended types or extended structures of OR model (such as ROW, NESTED TABLE, etc.).

We argue that producing a mass-switch from relational to real OR systems requires building new consistent methodologies for transforming business-oriented object models into data-based OR models. These new methodologies must be incorporated in consistent technological frameworks which integrate existing types of business application components and advanced OR databases. More than the simple UML-to-SQL3 basic mapping these methodological approaches must support:

- embedding the *OR database design* into the development process of the entire (business) information system lifecycle;
- a *code of good design practices* concerning rich OR model extensions at pair with the larger object-oriented design context;
- *a (technological) framework* for the automating schema generation process and for the reverse engineering of existing OR schemas.

2 ANALYSIS AND SOME “CRITICISM” OF OBJECT-RELATIONAL RELATED PAPERS

In the area of **object oriented methodologies for designing OR database schemas,** the debates on object representation as row in a table or value in a column have led to two methodological approaches, one based on SQL standard (SQL:1999 and SQL2008) (Feuerlich et al., 2007), (Stonebraker et al., 1990) and another based on The Third Manifesto “D” language (Darwen and Date, 1995) (Date and Darwen, 2007).

Our opinion is that it is better and more effective to manage these two approaches by trying to compare, correlate and mediate them rather than by setting them one against the other.

An object sometimes is the "equivalent" of a row in a table (the tuple type of table rows corresponds with the class equivalent) and sometimes is the equivalent of the value of a column. The Domain Driven Design methodology approach to Entity types described by Value types opens a relevant and semantically pertinent argumentation. But the discussion about ROW-CLASS equivalence implies another issue: in object oriented design there is a difference between CLASS and TYPE concepts, originated in OO debate about declaration/interface vs. implementation theory.

As for **MDA for OO-OR model mapping,** there are two representative approaches for the transformation of OO models formalized as UML class diagrams into OR schemas. One focuses on building an UML model adapted to OR construct by using UML extension mechanisms (like stereotypes) (Fern et al., 2005), (Marcos et al., 2003) and the other takes into consideration a plain UML model prepared to generate nested normal forms (Mok and Paper, 2001) (Chennamaneni and Grant, 2004).

We argue that MDA mapping rules might provide the basis for another meta-model formalized by using UML extension mechanism and OCL. MDA mapping rule will have to preserve the following semantics: **(a) at base** level: attribute, class, association; **(b) at enhanced** level (complex objects): complex entities through aggregation, composition and derivation.

Mapping composed attributes to SQL OR types like ARRAY, MULTI SET or NESTED TABLE might be a natural choice, but, in our opinion, the problem is not such simple. The selection of one of the options available requires a serious semantic analysis (Marcos et al., 2003) (Eessaar, 2006). One must make distinctions between:

- the multi-valued attributes with descriptive function and the associations and other entities in the frame of a composed structure,
- REFs (OIDs) and the foreign keys for simple associations,
- NESTED TABLES either for composition or aggregations,
- collection types like ARRAYS and MULTISET for multi-valued attributes.

SQL NESTED TABLE construct may not be suitable for implementing associations that are not aggregations, because this kind of design may cause redundancy problems (Mok and Paper, 2001). Therefore the database schema will be weaker to BCNF tables.

With respect to MDA layered architecture, the analysed proposals reveal the following alternatives: *MDA1*: from plane simple UML diagrams to OR implemented structures; *MDA2*: from UML-OR extended diagrams to OR implemented structures; and *MDA3* : UML plan diagram (conceptual model) to UML-extended OR diagrams (UML for SQL3) to OR implemented structures (Oracle concrete Model).

In our opinion, the MDA3 alternative can be further refined (let's say in *MDA4*) by taking into consideration the enterprise or business application systems. This refinement divides MDA1 into:

(1) UML-extended for business components and
(1') UML-extended OR diagrams (UML for SQL)
and MDA2 (the implementation level) into:
(2) platform specific business components (e.g. EJB) and (2') OR implemented structures (such as Oracle DB concrete Model).

To sum-up, a full-blown MDA architecture can assume the following levels:

- Layer 1: UML plain model (class diagrams)
- Layer 2: UML "prepared" (restricted) model (class diagrams)
- Layer 3: UML-OR SQL extended model (class diagrams to represent relational and extended structures)
- Layer 4: UML-OR Technological/Product extended model (relational and extended structures)
- Layer 4': UML for business components integration with OR-DBMS structures (DBMS product specific bridge extensions).

3 A DIFFERENT OO-OR MAPPING APPROACH TO IMPLEMENT SQL ABSTRACT DATA TYPES

The basic idea of our approach takes into consideration the possibility to expose SQL compliant O-R types using an ORDBMS. The implementation of types is made with a "true" object oriented language and platform (Java and Java Runtime Environment

The main *objectives* of this approach are: (1) *to develop models* that are s*emantically compliant with OO principles* using an object-orthogonal approach such as the one proposed by Date and Darwen (Darwen and Date, 1995); (2) *to maximize OR abstract data types portability* using ORDBMS extension mechanisms.

The proposed MDA architecture comes into two versions: an extended form for methodological reasons, and a slight compressed form for practical reasons. *The Extended form* (Figure 1) involves four layered models: (1) a platform independent model (PIM) containing the initial data structure types; (2) (3) a platform specific model (PSM-1) for Java platform embedded in the ORDBMS; an SQL specific model that preserves its independence against ORDBMS; (4) a platform specific model of targeted ORDBMS that will both contain: the SQL-ORDBMS specific type definitions, and the integration structures between implementation platform and ORDBMS exposure mechanism.

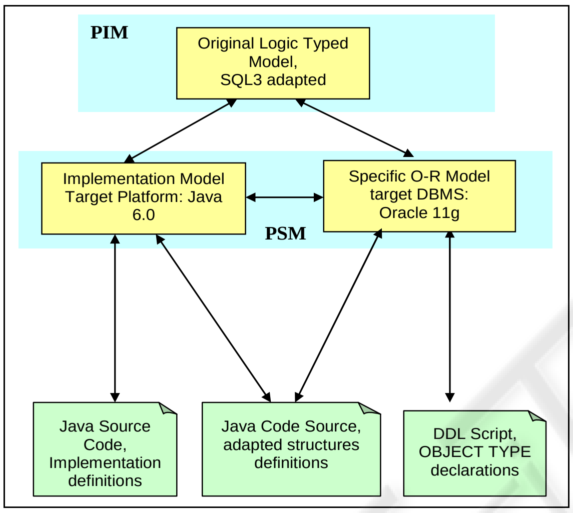
PIM
Original Logic Typed Model, SQL3 adapted
Implementation Model Target Platform: Java 6.0
Specific O-R Model target DBMS: Oracle 11g
PSM
Java Source Code, Implementation definitions
Java Code Source, adapted structures definitions
DDL Script, OBJECT TYPE declarations

Figure 1: Simplified MDA architecture.

The Compressed form (Figure 1) merges UML and SQL models into a single PIM model taking into consideration their platform-independent nature and their semantic equivalent constructs.

For getting practical results we used two kinds of tools. First, UML Case Tool produces an UML-PIM adapted model, a graphically and encoded form that is compliant with the MOF-XMI standard format of the OMG (http://www.omg.org/mof). Second, an MDA-generation tool parses, interprets and processes the XMI form of the UML model in order to produce the Java and Oracle OR specific SQL code. Our choices are ***ArgoUML*** and ***AndroMDA*** which have the advantage of compatibility and easy integration, aside from being leading open-source products based on open-standards (like MOF-XMI). Our UML-PIM model is formalized with UML *stereotypes*, presented in figure 2.

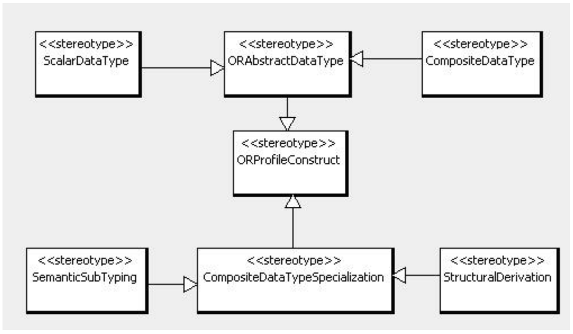
<<stereotype>> ScalarDataType
<<stereotype>> ORAbstractDataType
<<stereotype>> CompositeDataType
<<stereotype>> ORProfileConstruct
<<stereotype>> SemanticSubTyping
<<stereotype>> CompositeDataTypeSpecialization
<<stereotype>> StructuralDerivation

Figure 2: UML-OR Profile from ArgoUML.

The stereotypes mark the UML classes in order to enable the mapping and generation rules of the PSM Java and Oracle SQL structures. The ScalarDataType designates UML classes mapped into the immutable SQL abstract data types defining components (attributes) of SQL structured types. The structured types will result from the UML classes marked as CompositeDataType. These types will be further used to define SQL table structures to store their instances, or to define components of other structured types as single attributes or as collections (ARRAYs or NESTED TABLES). The generation strategies in PSM of those UML classes connected with generalization/specialization relationships will be guided through CompositeDataTypeSpecialization stereotype. Every mapping rule will be converted into a generation rule within AndroMDA context. The AndroMDA engine will process UML elements annotated with stereotypes through the metafacade-template system that produces Java and Oracle SQL structures. In short, the conversion logic should be incorporated in AndroMDA metafacades and the result will be interpreted by scripts from template files (see figure 3).

4 CONCLUSIONS AND OPEN ISSUES

An OO-OR framework could not be complete without closing the "gap" so that the objects stored in typed tables of ORDBMS could seamlessly "pass" forward and backward to concrete business components hosted by application servers or self-contained business applications. Taking into consideration the Java technological space, there are already two persistence frameworks, JPA and JDO

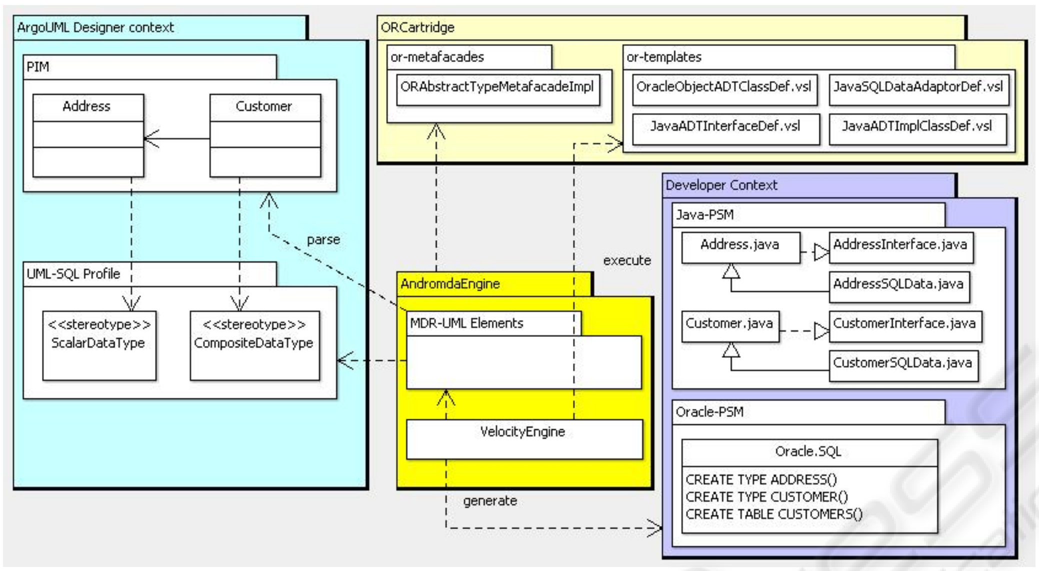

Figure 3: AndroMDA process to generate Oracle SQL Object Types and Java implementation classes.

But because they were built in an opaque manner concerning OR “native” types (SQL3 types), we have to rely on the lower JDBC standard API to communicate with databases. Form its 2.0 version, the JDBC standard has introduced some interesting extensions to support as closely as possible, the abstract-typed values from OR-DBMS.

In order to produce a full-fledged OO-OR framework, that have to be coherent, complete and at pair with JPA and JDO, we must to further explore a rich set of problems such as:

- an extensive set of good practices must be delivered to cover all OR SQL3 types in business application-specific types (e.g. the aggregation and sub-typing or generalization issues);
- the conversion from typed tables to application-specific collections;
- an OR compliant but OO specific query API must be designed and built with reasonable performance concerning scalability and queries' metrics (overall time and cost).

REFERENCES

Chennamaneni, R., Grant, E.S., 2004. *Comparison and Evaluation of Methodologies for Transforming UML Models to Object-Relational Databases*, Proceeding of Midwest Instruction and Computing Symposium, Morris, Minnesota.

Date, C.J., Darwen, H., 2007. *Databases, types, and the relational model. The third manifesto* (Reading, MA: Addison-Wesley).

Darwen, H., Date, C.J., 1995. *The Third Manifesto*, ACM SIGMOD Record, 24(1),, 39-49.

Eessaar, E., 2006. *Whole-Part Relationships in the Object-Relational Databases*, Proceedings of the 10th WSEAS International Conference on Computers, Vouliagmeni, Athens.

Fern M, Golobisky, A, Golobisky, V., 2005. *Mapping UML Class Diagrams into Object-Relational Schemas*, Proc. Of Argentine Symposium on Software Engineering, pg. 65-79.

Feuerlich, G., Pokorny, J., Richta, K., 2007. *Object-Relational Database Design: Can your Application Benefit from SQL:2003?*, Proceedings of the 16th International Conference on Information Systems Development Galway, Ireland, August 29-31.

Fortier, P., 1999 *SQL3. Implementing the SQL Foundation Standard*, McGraw-Hill.

Kleppe, A., Warmer, J., Bast, W., 2003. *MDA Explained: The Model Driven Architecture™: Practice and Promise*, Addison Wesley Professional.

Marcos, E., Vela, B., Cavero, J.M., 2003. *A methodological Approach for Object-Relational Database Design using UML*, Software System Modeling, Springer-Verlag.

Mok, W.Y, Paper, D.P., 2001 *On transformations from UML models to object-relational databases*, Proceedings of the 34th Annual Hawaii International Conference on System SciencesVolume , Issue , 3-6 Jan.

Stonebraker, M., Anton, J., Hanson, E., 1987. *Extending a Database System with Procedures,* ACM Transactions on Database Systems, 12(3), 1987, 350-376.

Stonebraker, M., Rowe, L.A., Lindsey, B., Gray, J., Carey, M., Brodie, M., Bernstein, P., Beech, D., 1990. *Third-Generation Database Systems Manifesto*, ACM SIGMOD Record, 19(3), 31-44.