

Model Transformation for Object-Relational Database Development

ABSTRACT

In this paper we define and formalize the model transformations that complete the methodological approach for the development of Object-Relational (OR) Databases (DB) proposed in MIDAS, a model driven methodology for the development of Web Information Systems. In this proposal the Platform Independent Model is the conceptual data model while the Platform Specific Model is the Object-Relational model that represents the OR DB schema. Since both of them will be represented in UML, we also summarize an UML profile for OR DB modeling. In this work we focus on the formalization of the mappings needed to get the OR DB schema from the conceptual data model. We have first specified them with natural language to later formalize them with graph transformation rules.

Categories and Subject Descriptors

H.2.4 [Database Management]: Logical Design – *Data models*

I.6.5 [Computing Methodologies]: Model Development – *Modeling Methodologies*.

General Terms

Design, Standardization, Languages

Keywords

Object-Relational Databases, Model Transformation, Formalization, Graph Transformation

1. INTRODUCTION

In spite of the impact of relational Databases (DBs) over the last decades, this kind of DBs has some limitations for supporting data persistence required by present day applications. Due to the hardware improvements, more sophisticated applications have emerged, which can be characterized as consisting of complex objects related by complex relationships. Representing such objects and relationships in the relational model implies that the objects must be decomposed into a large number of tuples. Thus, a considerable number of joins are necessary to retrieve an object and, when tables are too deeply nested, performance is significantly reduced [3]. A new DB generation appeared to solve these problems; the object-oriented DB generation, which includes Object-Relational (OR) DBs [22]. This new technology is well suited for storing and retrieving complex data because of its support for complex data types and relationships,

multimedia data, inheritance, etc. This well established DB technology is based on standards [12] and includes many characteristics into its model, extending the basic relational model with user defined types, collection types, typed tables, generalizations, etc. Nowadays the OR DBs are widely used in industry and research and they have been incorporated into a lot of commercial products [11][16][20].

Nevertheless, good technology is not enough to support these extensions. It is necessary to provide methodologies that guide designers in the OR DB development task, as it has been done traditionally with relational DBs [6]. Such methodologies should incorporate the OR model, and take into account old and new problems, like choosing the right technology, solving platform migration and platform independence problems, maintenance, etc.

In the line of the new trend in software development, the Model Driven Engineering [21], we propose a model driven approach for OR DB development. That is, using different models for each different technology, abstraction level and phase of the development process, and defining mapping rules between those models. This is the main difference with regard to other existing approaches like the ones from [18] and [19]. The one we present here is a model driven approach, where *models* are considered as *first class entities* and *mappings* between the different models have been defined and formalized. Formalizing the mappings before implementing them leads to detection of errors and inconsistencies in the early stages of software development and can help to increase the quality of the built models as well as the subsequent code generated from them. Likewise the formalization of mappings simplifies the development of tools supporting any model driven approach.

Our approach for OR DB development is framed in MIDAS [15], a model driven methodology for the development of Web Information Systems (WIS). MIDAS is based on the Model Driven Architecture (MDA) [17] and it considers, when modeling the system, the aspects of *content*, *hypertext* and *behaviour* at the levels of Computation Independent Models (CIM), Platform Independent Models (PIM) and Platform Specific Models (PSM). In Figure 1 we can see the simplified MIDAS architecture, where the CIMs, common to all the system, as well as the different PIMs and PSMs to represent the aspects of *content*, *hypertext* and *behaviour* are depicted.

In this work we focus on the *content* aspect, which corresponds to the traditional concept of a DB for the PIM and PSM levels. In our approach, the proposed data PIM is the conceptual data model represented with an UML class diagram. The data PSM is the OR model or the XML Schema model, depending on the technology that will be used to deploy the DB. In this paper, we will focus on the OR model; the XML DB development process is presented in [24]. In the case of the OR technology we have proposed two different PSMs: the first one is based on the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'07, March, 11-15, 2007, Seoul, Korea.

Copyright 2007 ACM 1-59593-108-2/06/0004...\$5.00.

SQL:2003 standard [12]; and the second one on a specific product, Oracle10g [20]. Both models will be defined using UML, as recommended by the OMG. At times, some extensions to UML are needed to be able to represent all the semantics of such models. A coherent set of such extensions, defined for a specific purpose, constitutes a UML profile. This way, we have developed two complete UML profiles for OR DB based on the preliminary work of [14] and adapted them to the latest versions of the standard and the selected product, including also the metamodels of the profiles. They will be summed up in the next section.

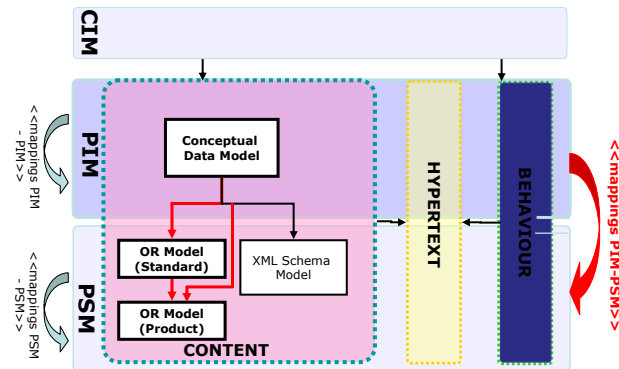


Figure 1. Simplified MIDAS Architecture

Moreover, our proposal includes a set of mappings to transform the PIM into the PSM, in other words, to transform the conceptual data model into the OR DB schema. In this paper we focus on these mappings, which are described in section 3 and based on the ones presented in [14]. They are first textually described and then formalized by means of graph transformation rules. Given that the method is based on a continuous development process in which, according to the MDA principles, the models act as the prime actors, the mappings between the models play a very important role. This process consists basically on the generation of an output model starting from one or more input models on which the mappings are applied. In the remaining steps of the process, this output model acts as one of the input models. In this work, we follow a graph transformation approach to formalize the mappings defined for OR DB development. The term *Graph Transformation* is used to refer to a special kind of rule-based transformations that are typically represented diagrammatically [9]. So, given that we

firstly define the mapping rules in a rule-based manner, it seems properly to use a graph transformation approach to later formalize them. By defining and formalizing the model transformations we complete the method definition, obtaining a complete methodology for OR DB development.

2. OBJECT RELATIONAL DATABASE DEVELOPMENT

As we have already mentioned, the content aspect of a WIS corresponds with the traditional concept of a DB. In our method for OR DB development, the proposed data PIM is the conceptual data model, represented with UML class diagram and developed according to the user requirements. The logical design of the DB is carried out starting from the obtained data PIM. Applying the mappings that will be detailed in the next section to this PIM we obtain the ORDB schema at the PSM level represented in extended UML. In the next subsections we will describe the UML profiles for OR DB modeling (for the standard and Oracle10g). Both of them use as a prerequisite the UML profile for relational data models presented in [1].

2.1 SQL:2003 OR Profile

The SQL:2003 OR profile is a UML profile that supports the modeling of an OR DB schema based on the SQL:2003 standard, using an extended UML class diagram. The diagram showed in Figure 2 (a) represents only the object characteristics of the SQL:2003 OR metamodel, that is, the artifacts corresponding to the relational metamodel are not included. Data types can be roughly classified into Arrays, Multiset, Rows, Reference Types and Structured Types (which allows User Defined Types to be specified). A Typed Table is based on one Structured Type. A Structured Type consists of Attributes and Methods. Additional constraints, described in OCL or natural language, may be specified to express its “well-formedness” rules.

The UML profile that represents this metamodel is described as an UML package, with the stereotype <<profile>>. Figure 2 (b) shows the profile represented according to the UML specification for profiles definition. It contains different stereotypes that correspond to some of the classes and associations of the original metamodel. For more details, see [14].

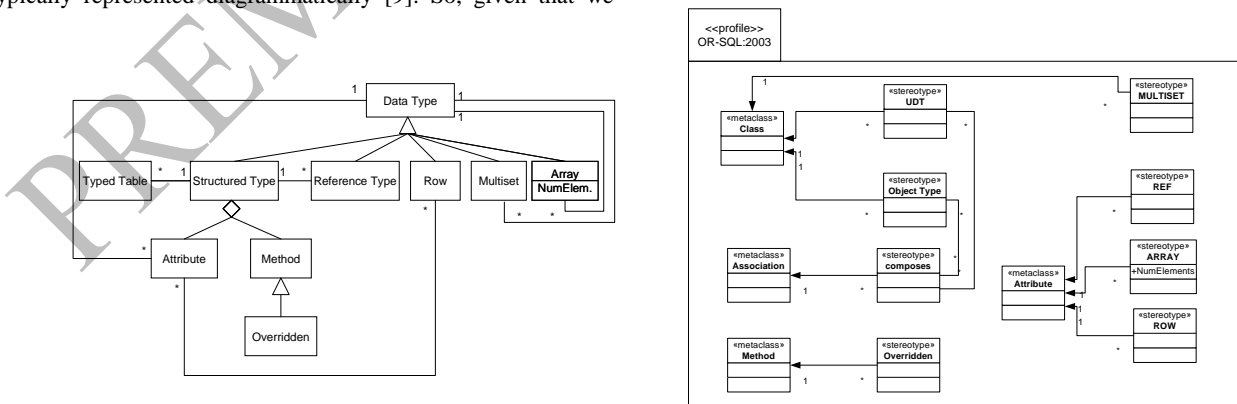
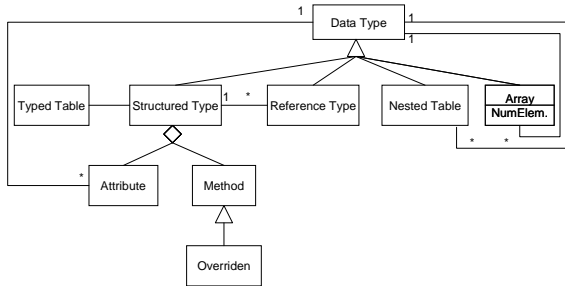


Figure 2. (a) SQL:2003 Metamodel (b) SQL:2003 UML profile

2.2 Oracle OR Profile

This section shows the UML profile for representing an OR model based on a specific DBMS, Oracle10g. Although it is very similar to the one for SQL:2003, we prefer to present it separately, because each product will provide a different profile. Figure 3 (a) shows the metamodel that represents the object



characteristics of Oracle10g's OR model. The main differences with the standard metamodel are: Oracle neither supports the ROW type nor the inheritance of tables and Oracle does support the NESTED TABLE type instead of the MULTISET type (although they represent the same concept). In Figure 3 (b) the corresponding <<profile>> is presented.

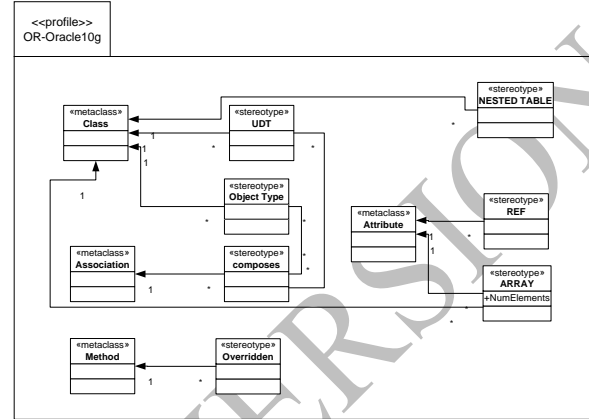


Figure 3. (a) Oracle 10g OR Metamodel (b) Oracle 10g OR profile

3. MAPPINGS FOR OBJECT-RELATIONAL DATABASE DEVELOPMENT

As mentioned before, the proposed approach for OR DB development is based on the definition of models at different abstraction levels, the basis of the model-driven development paradigm [21]. In the previous section we have defined the metamodels (consequently the models) that must be considered in our approach, thus, according to MDA principles, the only issue that must be faced in order to complete our proposal is the definition of the mappings between these models. This process stands for *model transformation* [17].

3.1 Model Transformation Rules for Object-Relational Database Development

According to [17] “the mapping description may be in natural language, an algorithm in an action language, or a model in a mapping language”. In this case, and as a first approach to model transformation for OR DB development, we have decided to describe the transformation rules in natural language for later express them as graph transformation rules.

For space reasons, in Table 1 we sum up the mappings proposed from PIM to OR PSMs, for more details see [14] where these mappings were first sketched. As we have already mentioned, we have considered two different PSMs: the OR DB model corresponding to the SQL:2003 standard, and the OR DB model for a specific product, Oracle10g.

Table 1. Mappings to transform the data PIM into the OR data PSMs

Data PIM	Standard Data PSM (SQL:2003)	Product Data PSM (Oracle10g)
Class	Structured Type + Typed Table	Object Type + Object Table
Class Extension	Typed Table	Table of Object Type
Attributes	Multivalued	Varray/Nested Table
	Composed	Object Type (column)
	Calculated	Trigger/Method
Association	One-To-One	Ref/[Ref]
	One-To-Many	[Ref]/[Multiset/Array]
	Many-To-Many	[Nested Table/Nested Table] [Varray/Varray]
	Aggregation	[Nested Table/Varray] of References
	Composition	[Nested Table/Varray] of Objects
	Generalization	Types/Typed Tables

3.2 Graph Transformations for Object-Relational Database Development

According to MDA principles, the model to model transformation of our approach for OR DB development must be automated, at least in some extent. We have decided to use a graph transformation approach [9][10] to achieve this objective. Using a graph transformation approach results in two main advantages: on the one hand, graph grammars are based on a solid mathematical theory and therefore they present a number of attractive theoretical properties that allows formalizing model transformations; on the other hand, the use of graph grammars for mappings definition could be shown as a direct step towards to implementation since projects like the Attributed Graph Grammar (AGG) System [4], VIATRA [7] or AToM3 [8] will provide us with the facilities to automate model to model transformations defined as graph transformations. Moreover, as previously mentioned, the term Graph Transformation is used to refer to a particular category of rule-based transformations that are typically represented diagrammatically. So, given that we have already formally defined the mappings in a set of rules, it seems appropriate to translate these rules to graph transformations rules. Finally, from a pure mathematical point of view, we can think on UML-like models as graphs. A graph has nodes and arcs, while an UML model have classes and associations between those classes; this way the fact that models are well represented as graphs is particularly appealing to shorten the distance between modelers and model transformation developers, a big problem around model transformation. Rule-based transformations with a visual notation may close the semantic gap between the user's perspective of the UML and the implementation of transformations [25].

To express model transformations by graph grammars, a set of graph rules must be defined. These rules follow the structure LHS:= RHS (Left Hand Side:= Right Hand Side). Both, the LHS and the RHS are graphs: the LHS is the graph to match while the RHS is the replacement graph. If a match is found on

the source model, then it is replaced by the RHS in the target model. In this work we have used the approach already introduced in [4] to define the graph rules that collects the transformation rules proposed in Table 1.

According to these guidelines, we have defined the graph rules for the model transformations needed in our proposal for OR DB development. Now we will present these graph rules. It should be pointed that the graph rules showed are valid for mapping the PIM to the PSM for SQL:2003. Nevertheless, we have already mentioned that the same are valid for the Oracle PSM just by considering some minor differences that will be remarked when needed.

3.2.1 Persistent Classes

Figure 4 shows the graph transformation rule to map persistent classes (classes in the PIM) to DB schema objects (stereotyped classes in the PSM) as well as how this rule can be instantiated. Whenever an UML class stereotyped as persistent is found on the PIM (①-③), a structured type (also known as object type) and a typed table are added on the PSM (④). The type of the new table will be the structured type. This way, the typed table will be the extension of the structured type. Each property class associated with the persistent class by means of a composition will be represented by adding an attribute to the structured type (②⇒②').

There is no difference between the SQL:2003 and Oracle10g mappings of persistent classes, at least, at the design level. We just have to add the "AS OBJECT" clause when implementing the resulting design in Oracle10g.

On the upper side of Figure 4 we show how this graph transformation rule can be instantiated. On the left side of the figure we have the PIM: the *User* UML class with just one attribute, represented below like an instance of the relative part of the UML metamodel. The right side shows the corresponding PSM as the result of applying the transformation rule: an object type named *user* with just one attribute, the *first_name* of the user.

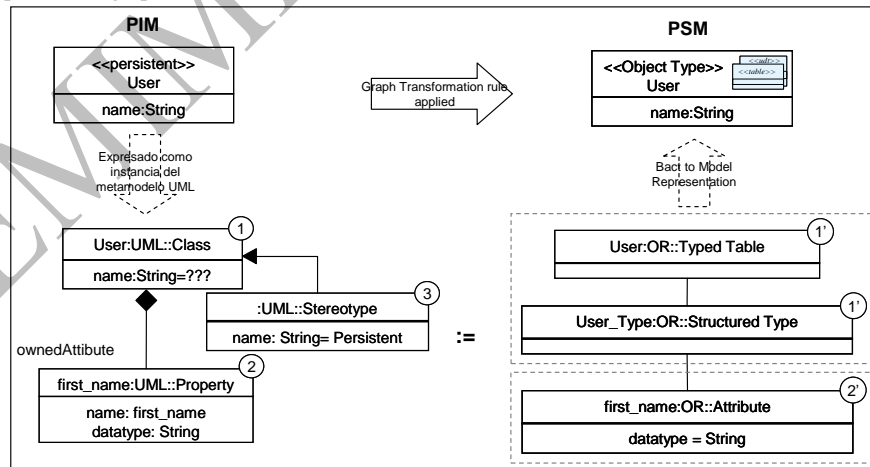


Figure 4. Persistent class mapping example

3.2.2 Attributes

As showed on Table 1, there is specific transformation rules for the different kind of attributes that should be took into account when mapping PIM persistent classes to the OR PSM:

Multivalued attributes on the conceptual model correspond to collection types on OR DB schemas: ARRAY or MULTISET for SQL:2003 and VARRAY and NESTED TABLES for Oracle10g. Figure 5 shows the respective graph transformation

rule for SQL:2003 when we choose the MULTISET type for the PSM. The UML persistent class (①) has a multivalued property (②), as stated by the true value for the *isMultivalued* attribute of the UML property. Therefore, the structured type that maps the persistent class (①), includes a MULTISET attribute (②'-②'). The same is valid when using ARRAY, VARRAY or NESTED TABLES, just by replacing MULTISET with the one selected.

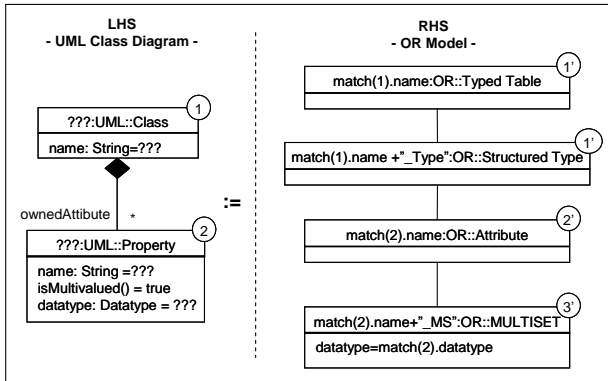


Figure 5. Multivalued attribute mapped to a Collection type

Composed Attributes included in the persistent classes are mapped to ROW or Structured Types in SQL:2003 (Object Type in Oracle10g), but this time without extension, that is, we will define the Structured Type but we won't create the associated typed table since the different instances of the new Structured Type, that is the extension, will be included in the extension of the Structured Type that maps the PIM class in which the composed attribute was found. In Figure 6 we show the graph transformation rule for mapping composed attributes into SQL:2003 ROW types: the UML class included in the PIM owns a property (①-②) whose data type is in turn another UML data type with its own properties (③-④). This data type is mapped to the PSM for SQL:2003 by means of a ROW type (③⇒③'). Each one of the properties of the UML data type for the UML property are mapped to properties of the ROW type (④⇒④'). Finally, the Structured Type corresponding to the persistent class is associated with the ROW type corresponding to the UML data type (①'-③').

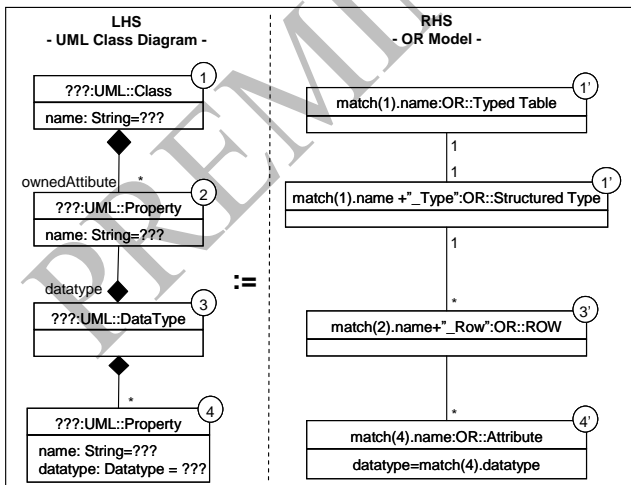


Figure 6. Composed attributes mapped to a ROW type

Derived Attributes are the last type of attributes to consider. This kind of attribute is mapped to the OR DB PSM just by adding a trigger or method in the Structured Type that maps the PIM class in which the attribute is included. This method or trigger calculates the value for the attribute. At graphical level, it involves the specification of the method or trigger signature in the Structured Type. Figure 7 shows the respective graph transformation rule for SQL:2003 (and Oracle10g).

At PIM level the derived attribute is recognized by the true value for the *isDerived* attribute of the UML property (②). At PSM level, the result of applying the transformation rule is similar to the one showed for simple classes with attributes, just differing in the addition of a OR Method associated with the Structured Type (①'-②'). This method will calculate the value for the attribute of the Structured Type.

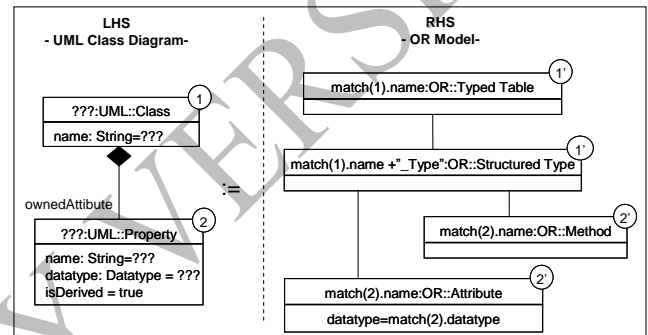


Figure 7. Derived attribute mapped by means of a method

3.2.3 Associations

UML associations found at the PIM level are mapped at the PSM level to uni- or bi-directional relationships in the DB schema, taking into account the additional cost of maintaining bidirectional relationships, given that their consistency is not maintained by the system. The criteria we follow to choose the relationship type depends directly in the kind of queries that will be formulated over the association. This way, if the queries will navigate typically just in one-way of the association, a unidirectional relationship will be used, with the consequent saving in terms of maintenance cost. Nevertheless, if we know that the two-ways of the association will be navigated equally we will have to use a bi-directional relationship to ensure a low response time.

Focusing on mapping associations from the PIM (UML conceptual model) to relationships in the PSM (extended UML model for ORDB), the most important feature is the association cardinality. So, depending on the maximum multiplicity of the classes involved in the association, we propose the following transformation rules. For the sake of simplicity, in all the cases we have considered only uni-directional relationships, given that for bi-directional relationships the same rules can be applied, just by duplicating the construction in both sides of the rule. Likewise we present just the transformation rules for SQL:2003 and after we mention the differences to consider for Oracle10g.

As Figure 8 shows **Maximum One Multiplicity** associations are identified in the PIM by the value 1 for the *upper* attribute included in the UML property (②) of the source class. The corresponding uni-directional relationship in the PSM is build upon an attribute in the Structured Type corresponding to the

source class (①'-②'). This attribute of type *REF* will be a pointer to the Structured Type that maps the target class of the association (②'-②').

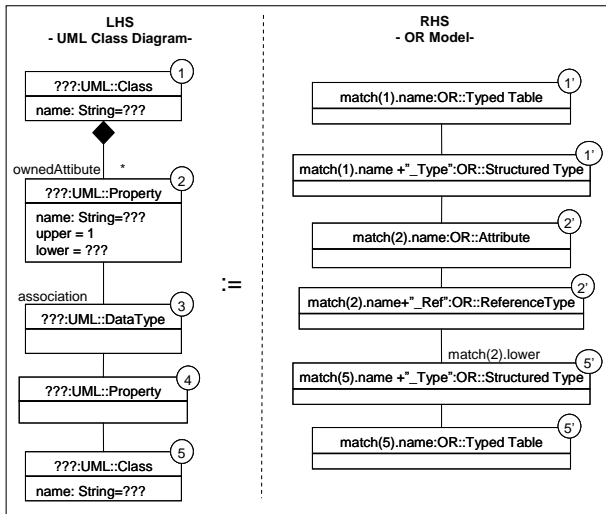


Figure 8. Maximum one multiplicity mapped to an uni-directional relationship

As showed in Figure 9 the only difference between this transformation rule and those for **Maximum K Multiplicity** and **Maximum Many Multiplicity** relies in the type of the attribute included in the Structured Type corresponding to the source class: the former will use an ARRAY type with exactly *K* elements, while the later will use a MULTISET type, since we don't know how many elements will participate in the association. The previous rules are valid for SQL:2003, the same are valid for Oracle10g just by using VARRAY instead of ARRAY and NESTED TABLE instead of MULTISET.

Since the transformation from PIM to PSM should be as accurate as possible we have distinguished among the special kind of attributes, specifying different transformation rules for each one. Likewise, we propose in the next subsections different transformation rules for the special kind of relationships that can be found at PIM level: generalizations, aggregation and composition.

3.2.4 Generalizations

While generalization is supported in two ways in SQL:2003: types and typed tables, it is just supported in one way in Oracle: types. So, in Oracle10g you could have two tables sharing the same structure: attributes, methods and data-types for those attributes and methods.

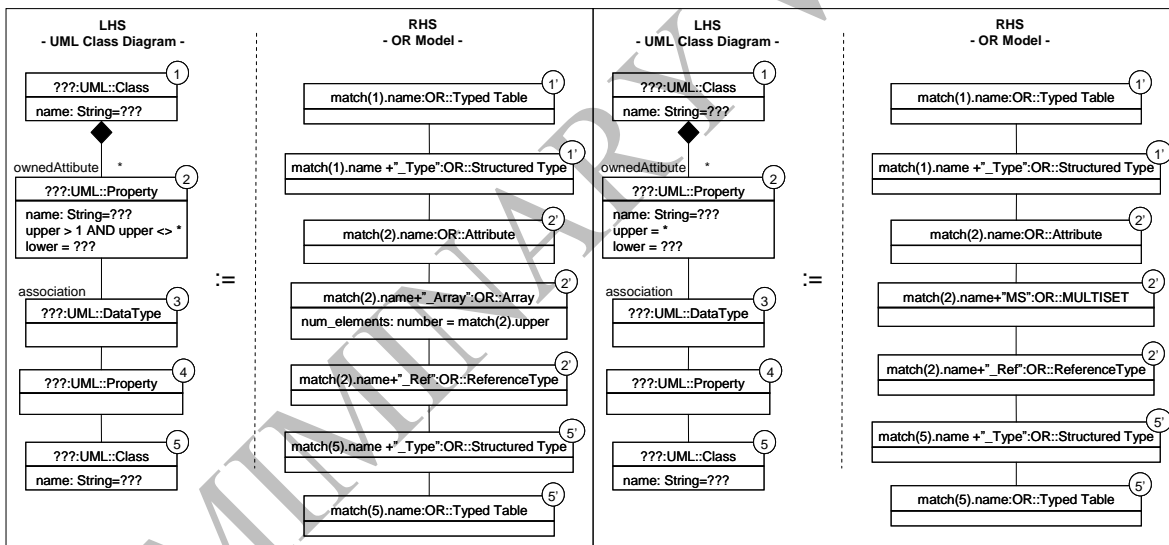


Figure 9. (a) Maximum *K* Multiplicity and (b) Maximum *Many* Multiplicity associations mapped to uni-directional relationships

Nevertheless if you need the two tables sharing the same restrictions, you will have to duplicate the definition of those restrictions in both tables. That's why generalization is barely used when working with Oracle, so we focus in the transformation rules for SQL:2003. As showed in Figure 10, we distinguish two situations: in the simple one the subclass does not include additional features with respect to the parent class (a). This is the typical situation in which we just need to distinguish among different concepts sharing the same structure, i.e. square, rectangle and rhombus, all of them descendants of four edges polygons. In the complex one (b) the descendant presents additional features that should be take into account, i.e. the mean of transport class could be generalized in car, which

owns a number of wheels, and plane, owning a number of wings.

The former situation is the simplest to map: since the descendant does not include any additional feature, we can map it to the DB schema as a typed table descending from the one corresponding to the parent class in the conceptual model. This way, the restrictions defined over the parent typed table will be preserved. The later is more complex to map: to gather the additional features of the descendant class in the PIM, we have to include a new structured type in the PSM, comprising these additional features (properties, methods or whatever). This new type descends from the one corresponding to the parent class in the PIM.

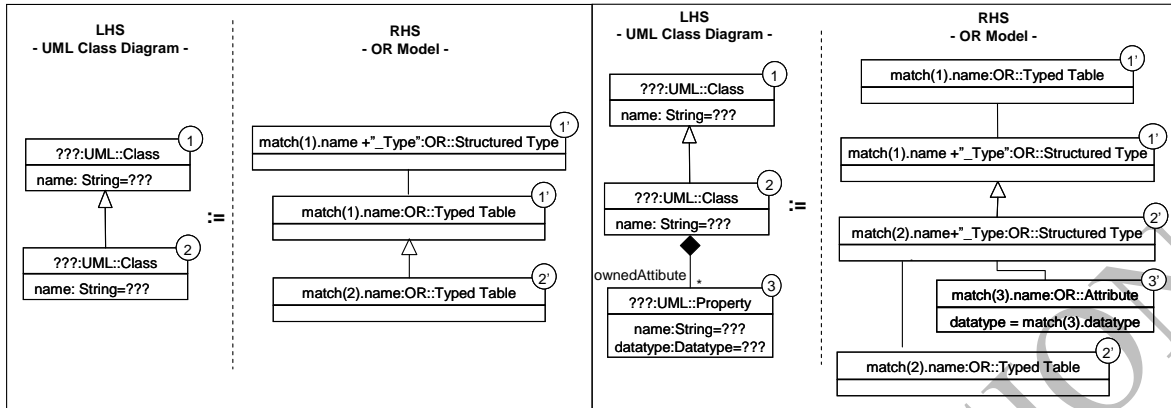


Figure 10. Generalization mappings in SQL:2003: (a) simple situation (b) features added in the descendants

3.2.5 Aggregations/Compositions

The transformation of aggregation and composition associations from PIM to PSM (from conceptual model to DB schema) follows the same approach in both cases: the inclusion of a collection type attribute in the structured type of the PSM corresponding to the whole entity of the PIM. Figure 11 shows the graph transformation rules for aggregation (a) and composition (b) associations. In the case of aggregation, recognized by the *shared* value for the *aggregation* attribute in the UML property (2), since the parts, that is, the entities associated to the whole, can exist beyond the lifetime of the

whole, the collection type will be a set of references to the part objects stored in their own typed table (2'-5'). On the other hand, the part objects do not exist beyond the lifetime of the whole object in the case of composition, so the collection type will be a collection of objects instead of a collection of references. That's why in Figure 11 (b), the second typed table, as well as the Reference Type are suppressed: the part objects are included directly in the whole object that is in the typed table corresponding to the persistent class (1'-2'-5') while in the aggregation mapping, they were referenced from the whole object.

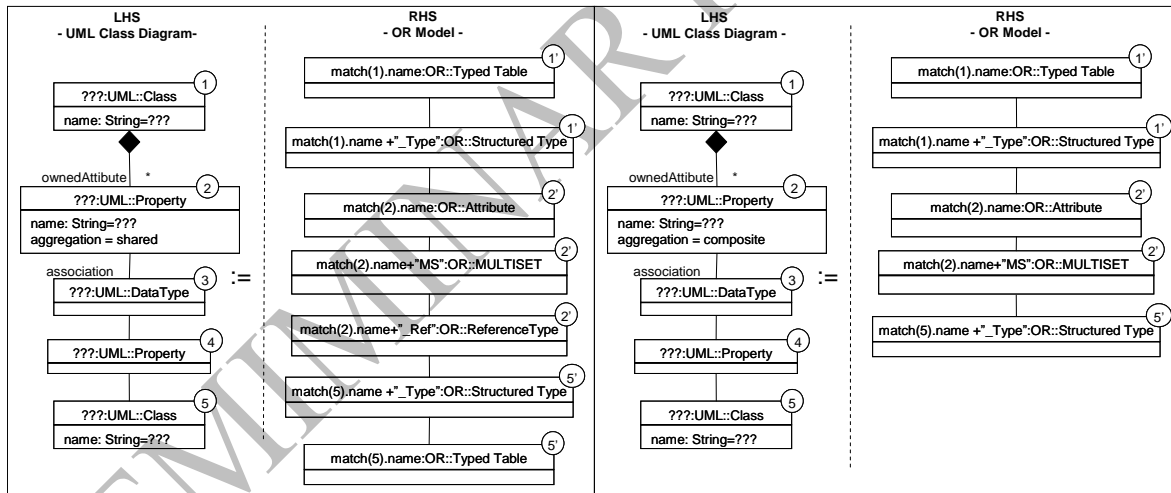


Figure 11. (a) Aggregation and (b) Composition mappings

4. CONCLUSION AND OPEN ISSUES

In this paper, by defining the model to model transformations from PIM to PSM, we have completed the definition and formalization of our proposal: a model driven development process for OR DB framed in MIDAS. In this proposal the PIM is the conceptual data model represented with an UML class diagram while two different PSMs to model the OR DB schema have been proposed and formalized by defining the corresponding profiles and metamodels. The first one supports the modeling of the DB schema for the SQL:2003 standard and the second one serves to model the DB schema for a specific product, Oracle10g. To complete the definition of the model driven development process for OR DB we have defined and

formalized the mappings to transform the data PIM into the data PSM. Firstly, we have summed up the transformation rules in a declarative manner to later map them to graph transformation rules. The formalization of these mappings is a previous step towards automating these graph rules using some of the existing facilities to automate graph transformations or a domain specific language for model transformation, such as ATL [13].

Moreover, the formalization of the model transformations as well as the previous formalization of the metamodels can be used to detect errors and inconsistencies in the early stages of the development and can help increasing the quality of the built models as well as the subsequent code generated from them. These activities are specially important in proposals aligned with

MDA (like MIDAS and the proposed method for OR DB development) because it proposes to use the models as the mechanism to carry out the whole software development process.

This paper allows confirming one of the hypotheses around model transformation sketched in [4]: PIM to PSM transformations are more prone to be automatized than PIM to PIM transformations. Because, while the former implies decreasing the abstraction level and consequently handling more specific artifacts easier to be modeled; the later requires a higher level of decision making from the designer due to a higher abstraction level of the implied elements.

At the present time we are working in the integration of the OR DB development process in M2DAT (MIDAS MDA Tool), a case tool which integrates all the techniques proposed in MIDAS for the semiautomatic generation of WIS which is now under development in our research group. Its early functionalities have been already presented in previous works [23]. Besides, the open issue of automatizing the graph transformations defined by using some of the existing technologies for model transformation has been tackled.

5. ACKNOWLEDGMENTS

This research has been carried out in the framework of the GOLD project financed by the Spanish Ministry of Education and Science (TIN2005-00010).

6. REFERENCES

- [1] S. Ambler, *Agile Database Techniques*. Wiley, 2003.
- [2] P. Atzeni, S. Ceri, S. Paraboschi and R. Torlone, *Database Systems. Concepts, Languages and Architectures*. McGraw-Hill, 1999.
- [3] E. Bertino and E. Marcos, *Object Oriented Database Systems*. In: *Advanced Databases: Technology and Design*, O. Díaz and M. Piattini (Eds.). Artech House, 2000.
- [4] F. Buttner and M. Gogolla *Realizing UML Metamodel Transformations with AGG*, Proceedings of GT-VMT. Electronic Notes in Theoretical Computer Science. Vol. 109, December 2004, pp. 31-42. 2004.
- [5] P. Cáceres, V. de Castro, J.M. Vara and E. Marcos. *Model transformations for hypertext modeling on web information systems*. In: Proc. of the 2006 ACM Symposium on Applied Computing (Dijon, France, April 23 - 27, 2006). SAC '06. ACM Press, New York, pp. 1232-1239, 2006.
- [6] P.P. Chen, *The Entity-Relationship Model – Toward a Unified View of Data*. ACM Transactions on Database Systems, Vol. 1, No. 1. March 1976, pp. 9-36, 1976.
- [7] G. Csertan, G. Huszerl, I. Majzik, Z. Pap, A. Pataricza and D. Varro, *VIATRA – Visual Automated Transformations for Formal Verification and Validation of UML Models*, in: Proc. of 17th IEEE International Conference on Automated Software Engineering (ASE'02), IEEE Computer Society, Los Alamitos, CA, USA, 2002, pp. 267-285.
- [8] J. De Lara, H. Vangheluwe and M. Alfonseca, *Meta-Modelling and Graph Grammars for Multi-Paradigm Modelling in AToM3*, Software and Systems Modelling, Vol 3(3), Springer-Verlag. August 2004, pp.: 194-209.
- [9] H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg. *Handbook of Graph Grammars and Computing by Graph Transformation*. Vol(1). World Scientific, 1999.
- [10] K. Ehrig, E. Guerra, J. de Lara, L. Lengyel, T. Levendovszky, U. Prange, G. Taentzer, D. Varró, and S. Varró-Gyapay, *Model Transformation by Graph Transformation: A Comparative Study*, in: Proc. of Model Transformations in Practice Workshop, at MoDELS Conference, Montego Bay, Jamaica, 2005.
- [11] IBM DB2 Universal Database. <http://www-306.ibm.com/software/data/db2/>
- [12] ISO / IEC 9075 *Standard, Information Technology – Database Languages – SQL:2003*, International Organization for Standardization, 2003.
- [13] F. Jouault and I. Kurtev, *Transforming Models with ATL*, in: Proc. of the Model Transformations in Practice Workshop at MoDELS Conference, Montego Bay, Jamaica. 2005.
- [14] E. Marcos, B. Vela and J. M. Cavero, *Extending UML for Object-Relational Database Design*. Fourth Int. Conference on the Unified Modeling Language, UML 2001. Toronto (Canada), LNCS 2185, Springer-Verlag, pp. 225-239. October, 2001.
- [15] E. Marcos, B. Vela, P. Cáceres and J.M. Cavero, *MIDAS/DB: a Methodological Framework for Web Database Design*. DASWIS 2001. Yokohama (Japan), November, 2001. LNCS 2465, Springer-Verlag, pp. 227-238. September, 2002.
- [16] Microsoft SQL Server. <http://www.microsoft.org/sql/>
- [17] J. Miller and J. Mukerji (eds.), *MDA Guide Version 1.0.1*, OMG, Framingham, Massachusetts, June 2003.
- [18] R. Muller *Database Design for Smarties*. Morgan Kaufmann, 1999.
- [19] E.J. Naiburg and R.A. Maksimchuk, *UML for Database Design*. Addison-Wesley, 2001.
- [20] Oracle Corporation. Oracle10g Release 2 (10.2). SQL Reference. In www.oracle.com, 2005.
- [21] B. Selic, *The pragmatics of Model-Driven development*, IEEE Software, Vol. 20, Issue 5, Sept.-Oct. 2003, pp:19 – 25, 2003.
- [22] M. Stonebraker and P. Brown, *Object-Relational DBMSs. Tracking the Next Great Wave*. Morgan Kauffman, 1999.
- [23] J.M. Vara, V. De Castro and E. Marcos, *WSDL automatic generation from UML models in a MDA framework*, in: International Journal of Web Services Practices. Volume 1 – Issue 1 & 2. November 2005, pp.1 – 12. 2005.
- [24] B. Vela, C. Acuña and E. Marcos, *A Model Driven Approach for XML Database Development*, 23rd. International Conference on Conceptual Modelling Springer Verlag, LNCS 3288, pp. 780-794. 2004.
- [25] T. Weis, A. Ulbrich and K. Geihs, *Model Metamorphosis*, IEEE Software, Vol. 20, no. 5, September/October, 2003, pp. 46-51.