Evolution of Relational Database to Object-Relational Database in Abstract Level

Yugopuspito, Pujianto
PT. IPTN, Indonesian Aircraft Manufacture, Inc. | Department of Computer Science and Communication Engineering, Graduate School of Information and Electrical Engineering, Kyushu University

Araki, Keijiro
Department of Computer Science and Communication Engineering, Graduate School of Information and Electrical Engineering, Kyushu University

Evolution of Relational Database to Object-Relational Database in Abstract Level

Pujianto Yugopuspito [1,2] *and* **K***eijiro Araki* [1]

[1] Department of Computer Science and Communication Engineering
Graduate School of Information and Electrical Engineering
Kyushu University,
6-1 Kasuga Koen, Fukuoka 816-8580, JAPAN
E-mail: (pyugop, araki)@dontaku.csce.kyushu-u.ac.jp

[2] PT. IPTN, Indonesian Aircraft Manufacture, Inc.
Jl. Pajajaran 154, Bandung 40174, Indonesia

ABSTRACT

Relational Database is a mature database with a rigorous specification and is broadly applicable. The deficient in data representation has been known since the software application changed to object oriented. An effort should be taken to encounter the connection between existing Relational Database with a new software application that is object oriented. In a case where a complete migration is not the choice of solution since the existing Relational Database should be preserved, then an Object-Relational Database might be a solution. The evolution of Relational Database to Object-Relational Database can be formulated as a transformation process plus information addition process. This work is intended to give a novel view on the evolution of Relational Database to Object-Relational Database and some summaries of the additional information for the evolution. Furthermore we present transformation rules in diagram and its formal specification underlying the rules.

Keywords

Relational Database, Object-Relational Database, IDEF1X, UML, Formal Methods, Z.

INTRODUCTION

The object-relational DBMS (ORDBMS) is the newest commercial breed of DBMSs which embraces some object-oriented features and encapsulates these features to an RDBMS, creating an ORDBMS. ORDBMSs are mainly based upon the criteria defined by Stonebraker et al. [14]. It is called "Third Generation Database System Manifesto." This manifesto is built as an opposing manifesto to Atkinson et al. [1]. Stonebraker suggests to extend the capabilities of an RDBMS to include the support for richer object structures and rules and still be open to other systems. The concept is sharpened and discussed in detail in his book, [12] then revised in the second edition [11].

Stonebraker categorises DBMSs into four quadrants of metrics. Simple data without query capability is in the lower-left file system. The user does not want query access. Simple data with query access in the upper-left quadrant is RDBMSs. This is a DBMS for the users who require query with simple data. Since the data becomes more complex, the database will be shifted to the right quadrant. The lower-right is Persistent Language [11] (in the previous edition he noted this as OODBMS [12]). Users did not do any queries because data are manipulated with a third generation programming language such as C++. For the upper-right, he put the ORDBMS, it supports complex data with query operation.

Since ANSI X3H2 has not completed their work on establishing the SQL3 yet, there is no standard currently existing for ORDBMS. The concept of ORDBMS is not mature, yet some affirmative characteristics have been agreed on ORDBMS. No vendors can truthfully claim to be SQL3 compliant, but some vendors have done major changes to their RDBMS to include object relational features, e.g. ORACLE, IBMs DB2 and Informix. The ORDBMS may model its data based upon the extension to the relational model and relational calculi.

In this paper, we argue that the ORDBMS is an evolution of RDBMS. The "changing" from RDMS to ORDBMS is smooth and may have a place in settled industries, who reluctant to migrate their big RDBMS but want to support the object-oriented programming. In this paper, we are using abstraction level of solution i.e. informal method (diagram) and formal methods. The Informal method is represented by diagram that intuitively captures the real world. Since the informal dia-

gram contains ambiguous information by its nature, a formal method is used to assure the rigorous specification of evolution. We are using an example of IDEF1X mapping into an UML Class diagram as an example in informal method level and Z specification for the formal specification level.

EVOLUTION CONCEPT

The basic problem to address is handling object-orientation. It refers to the support of DBMS for acquiring, managing, accessing (searching) complex data types. The current RDBMS has at least two solutions. They are migration RDBMS into OODBMS and adding object-orientation in RDBMS, such that it becomes ORDBMS.

Revolutionary migration of RDBMS into OODBMS requires that all applications be modified to handle the new database. Such modification is expensive and probably must be done over time. Another factor being that an object-oriented data model is not always appropriate for all applications of data because some data really fit best in relational databases.

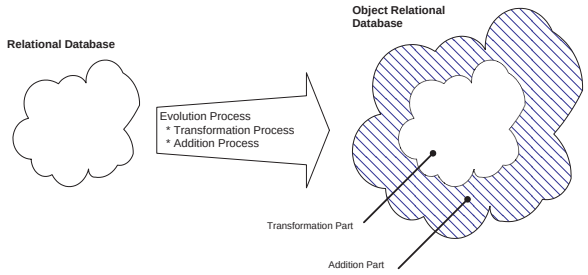
The evolution process of adding object-orientation is more suitable for "changing" RDBMS into ORDBMS rather than migration of RDBMS into OODBMS. First of all, one must consider inter-operability with older generation products since they refuse to die. The current product may work fine without any changes. The development of new application software becomes faster and cheaper, because it allows relational data model and object-orientation in the same DBMS.

Based on Stonebraker's quadrant of database, the basic extension from RDBMS into ORDBMS is the accommodation of complex data in complex query. These complexities are the additional characteristics that should be handled by ORDBMS, such that object orientation can be shown in RDBMS. These characteristics shall be compatible to the object-oriented model.

We call the "changing process" as Evolution of RDBMS to ORDBMS. The evolution is contained in two processes, the Transformation process and Information Addition process, such that the ORDBMS contains two parts, the Transformation part and Addition parts. The Transformation part contains the existing characteristics or items of RDBMS that still exist in ORDBMS while the Addition part contains the characteristics or items that do not exist in the RDBMS but shall exist in ORDBMS. Figure 1 shows the concept.

Considering the three layers of database, i.e. external layer, conceptual layer and physical layer [Coo 1997], the data model is the important, particularly semantic data model. The data model is designed to have constructs which are more closely related to the kind of structuring that human are thought to use when conceptualising about the real world. Here, we are only concerned with the data model.

Data modelling is the process of creating a description of a database which is intended to hold a collection of related information. The description of information is called schema and is created in terms of a data model. Originally, data modelling concentrated on the creation of a description of the structure of the database, but increasingly it is used to describe the restrictions which are to be applied to the constraints and some aspects of how the data is to be manipulated. Often it is called the behaviour of database. The coherent integration of structure construct, constraint construct and behaviour construct are clearly desirable from the stages of software development through maintenance points of view.

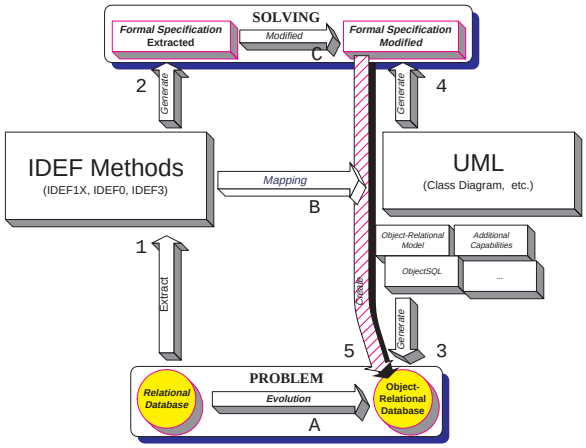


*F*igure 1. The Evolution Concept.

APPROACH TO EVOLUTION PROCESS

Figure 2 shows the general scheme of our approach. The goal of our project is indicated by arrow A. We take a detour path to solve the evolution process. We decided not to use a straight path, since the relational model and object relational model are different concepts of model. We believe that the object-oriented model should be used to create object-relational model. We want to use a different scenario for the solution. We have three levels here, the Database level, Informal level and Formal level. The Database level is the level where the problem exists, the level with arrow A. The Informal level is the abstraction of Database level, arrow B exist in this level. The formal level is the rigorous level of the Informal level, arrow C.

The Informal level, that is a diagrammatic approach, shall consist of IDEF Methods and UML methods. The informal level is an intuitive graphical representation that is easier to be adopted by industrial practice. For instance IDEF1X [7], is a standard for modelling Relational Database data model, while UML [8] is a de facto for modelling object-orientation. By nature, a diagram contains a certain ambiguity. We need something more powerful to assure the Informal level, the rigorous specification. So we call it the Formal level. This will assure the ability to perform high analysis or formal reasoning of the evolution.



*F*igure 2. Approach to Evolution Proces.

We begin by extracting specifications in IDEF from the existing relational database systems which are indicated by arrow 1, then we generate formal specifications from the IDEF graphical notation. These are placed on the Relational side.

On the Object-Relational side, we start from producing UML diagrams. The UML diagram is capable enough to represent the object-relational model. These diagrams can be determined by extending or mapping the IDEF Methods in relational part, arrow B. Then we are able to create the Object-Relational Database by using ObjectSQL, such as SQL3. This is an usual path to create a database from a given data model. We are also able to generate the formal specification from those UML diagrams, arrow 4. Notice that the Formal Specification Modified also can be determined by taking a detour via modification of formal specifications that are generated in Formal Specification Extracted. This specification formerly generated from IDEF to other formal specifications that are generated by UML which models the object-relational data model. Arrow C in Figure 2 indicates this process. The formal specifications on the ORDB side will finally create the target ORDB, which is indicated by arrow 5.

TRANSFORMATION RULES IN DIAGRAMS AND SPECIFICATION

Transformation rules are rules that we proposed for mapping the diagrammatical data model representation of Relational Database to diagrammatical data model of Object Relational Database. For better understanding we will use an example. A Table and Chair company (TcCo) buys parts from vendors and assembles them into tables and chairs, taken from [7]. Concepts such as part, vendor, and soon are of concern to TcCo. This universe of discourse has an existence and reality in TcCo independent of any model of it and is described as Figure 3 using an IDEF1X diagram, domain structure and sample instance tables. The Fully Attributed diagram of TcCo is called as Production View. The example is quite complete to show the essence of IDEF1X. The Independent Entities are *Part* and *Vendor*, the other entities are Dependent Entities. All relationships are Specific Relationships, as prerequisite for Fully Attributed view. The Identifying Relationships are shown in *part* (Independent Entity) to *Structure_Item* and *Made_Part* (Dependent Entity) to *Structure_Item* entities. A Mandatory Non-Identifying Relationship is represented by relationship of *Vendor* and *Brought_Part* entities. An Optional Non-Identifying Relationship is depicted as relationship *Vendor* to *Part* entities. A complete Categorization Relationship is represented by the relationship of part to *Made_Part* entities and *Part* to *Brought_Part* entities. The corresponding UML Class Diagram of this example is depicted in Figure 4.

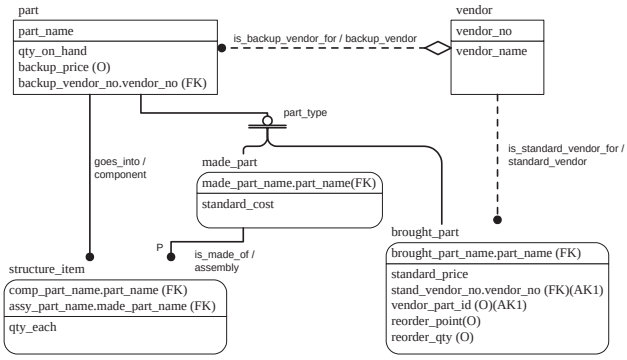


Figure 3. Example in IDEF1X representation.

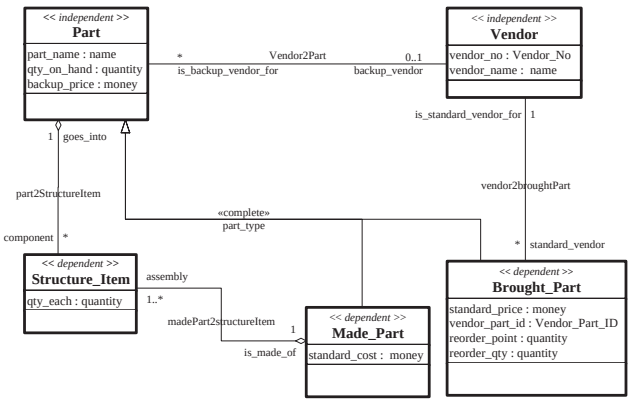


Figure 4. Example in UML representation.

An Entity in IDEF1X is corresponding to a Class UML Class diagram. We are introducing the stereotypes *independent* and *dependent*, for example the entity or class in UML. This is a transformation rule for an Entity to a Class on UML Class Diagram. This is the Transformation part. The Addition part for this Entity-Class rule might be the extension for User Defined Type (UDT), and its constraint, for instance, should be an acyclic graph when it inherits from other types.

Concering operations that are separated from data model in RDBMS but are an integreted definition in UML, the information about the operation that is merged in data model should be added. Those information categories are in Addition Part.

The Entity - Class corresponding in the Formal level is given in formal specification. In this example, we use a Z notatation [13]. The complete transformation IDEF1X into Z can be found in [15]. The excerption of translation IDEF1X into the Z specification can be shown as below.

$$EntityType ::= dependent \mid independent$$

$$\begin{array}{|l} Entity[PKeyAtt, OKeyAtt] \\ \hline entity : \mathbb{P}\, Record[PKeyAtt, OKeyAtt] \\ eType : EntityType \\ \hline \forall r1, r2 : entity \bullet r1.p = r2.p \Rightarrow r1 = r2 \\ \hline \end{array}$$

The instantiation for the example for an independent entity *Vendor* can be specified as follows:

$$\begin{array}{l} VndPK \mathrel{\widehat=} [vndNo : VndNo] \\ VndAtt \mathrel{\widehat=} [vndNm : VndName] \\ VndRec \mathrel{\widehat=} Record[VndPK, VndAtt] \\ VndEnt \mathrel{\widehat=} [Entity[VndPK, VndAtt][vndEnt/entity] \\ \quad \mid eType = independent] \end{array}$$

In an UML Class diagram site, the translation of class be given as follows:

$$\begin{array}{|l} Part \\ \hline ident : PartID \\ structureItem : StructureItem \\ partName : Name \\ qtyOnHand : Quantity \\ backupPrice : Money \\ \hline \end{array}$$

The relation between two entities is called Relationship, in IDEF1X. They are two categorizations, Specific Relationship and Non Specific Relationship. Of those relationships, it can be any of the following: Identifying Relationship, Mandatory Non-Identifying Relationship, Optional Non-Identifying Relationship, Complete Categorization Relationship or Incomplete Categorization Relationship. The UML Class Diagram has several kinds of possible connection between two classes i.e. Association, Aggregation, Composite and Generalisation (Inheritance). These connection also called Relationship [8]. After doing an assesment the Relationship in IDEF1X can be transformed into Relationship in UML Class Diagram. This is the Transformation part. The Addition Part might be all the components that do not exist in RDBMS, such as Dependency and Refinement relationships.

In the Formal level, the free type definition of *RelationshipType* and its rule are intoduced. The nature of a categorization relationship can be considered as a kind of Identifying Specific Relationship. Those are specified in Z notation as follow:

$$\begin{array}{l} RelationshipType ::= identifyingSpecificRelationship \\ \quad \mid nonIndentifyingSpecificRelationship \end{array}$$

The name of each relationship depends on the name of entities and the relationship name. The IDEF1X role name of relationship name can be used here. That is *e1_n1_n2_e2*, where *e1* and *e2*are the parent and child entity respectively, and *n1* is the relation from *e1* to *e2*, vice versa for *n2*. In this example, it is shortened to *entityNameA2entityNameB*, e.g. *vnd2prt*. In a categorization relationship, the name of relationship is given as "is_a", then added with the name of the child entity name, such that it can be distinguished from one another, e.g. *isAMdPrt*.

$$\begin{array}{|l} ProductionView \\ \hline VndEnt;\ PrtEnt;\ MdPrtEnt;\ BrPrtEnt; \\ \quad StrItmEnt; \\ One2M[VndRec, PrtRec] \\ \quad [vnd2prt/rel, v2pType/rType] \\ EOne2One[MdPrtRec, PrtRec] \\ \quad [isAMdPrt/rel, isMPType/rType] \\ EOne2M[BrPrtRec, VndRec] \\ \quad [vnd2brPrt/rel, v2bPType/rType] \\ EOne2One[BrPrtRec, PrtRec] \\ \quad [isABrPrt/rel, isBPType/rType] \\ EOne2M[PrtRec, StrItmRec] \\ \quad [prt2strItm/rel, p2sIType/rType] \\ EOne2EM[MdPrtRec, StrItmRec] \\ \quad [mdPrt2strItm/rel, mP2sIType/rType] \\ \hline \forall t1 : prtEnt \bullet t1.o.backUpVnd\mathsf{FKey}\ vndEnt \\ v2pType = nonIdentifyingSpecificRelationship \\ \forall t2 : mdPrtEnt \bullet t2.p.mdPrtNm\mathsf{FKey}\ prtEnt \\ isMPType = identifyingSpecificRelationship \\ \forall t3 : brPrtEnt \bullet t3.o.stdVndNo\mathsf{FKey}\ vndEnt \\ v2bPType = nonIdentifyingSpecificRelationship \\ \forall t4 : brPrtEnt \bullet t4.p.brPrtNm\mathsf{FKey}\ prtEnt \\ isBPType = identifyingSpecificRelationship \\ \forall t5 : strItmEnt \bullet t5.p.comPrtNm\mathsf{FKey}\ prtEnt \\ p2sIType = identifyingSpecificRelationship \\ \forall t6 : strItmEnt \bullet t6.p.asPrtNm\mathsf{FKey}\ mdPrtEnt \\ mP2sIType = identifyingSpecificRelationship \\ \hline \end{array}$$

The fomal level of UML Class Diagram for instance Association between Part and Vendor class, is called *Vendor2Part*, and can be representated as follows:

$$
\begin{array}{|l}
\hline
\textit{Vendor2Part} \\
\textit{parts} : \mathbb{P}\, \textit{Part} \\
\textit{vendors} : \mathbb{P}\, \textit{Vendor} \\
\textit{vendor2part} : \textit{One2Many}[\textit{Vendor}, \textit{Part}] \\
\hline
\mathrm{dom}\, \textit{vendor2part} \subseteq \textit{vendors} \\
\mathrm{ran}\, \textit{vendor2part} = \textit{parts} \\
\textit{vendor2part} \in \textit{One2Many}[\textit{vendors}, \textit{parts}] \\
\forall\, p1, p2 : \textit{parts} \bullet p1.\textit{ident} = p2.\textit{ident} \\
\qquad \Rightarrow p1 = p2 \\
\forall\, v1, v2 : \textit{vendors} \bullet v1.\textit{ident} = v2.\textit{ident} \\
\qquad \Rightarrow v1 = v2 \\
\hline
\end{array}
$$

The reasoning or higher analysis of both specification are still on going works.

CONCLUSION AND FURTHER WORKS

The process of changing from RDBMS into ORDBMS can be performed by the Evolution process. The process contains two other processes, Transformation process and information Addition process. The Transformation process will lead to corresponding existing RDBMS characteristics into ORDBMSS characteristics. The missing information or characteristics of ORDBMS should be added in the second process, the Information Addition process. This is opposing the migration of RDBMS to OODBMS.

This Evolution gave a different point of view on object-orientation. It is different from the usual object/relational mapping, that suggest to create a RDB data model based on their application. The direction is from application to a new relational database. We suggest to evolve the existing RDBMS to ORDBMS such that OO application can be supported. The direction is from database to a new application. It will give a significant contribution to practitioners for developing their OO application and preserving their current application. Because their current DBMS able to support both types of applications.

The further works of the novel concept of Evolution RDBMS into ORDBMS shall include an extension of relational algebra and relational calculus. The evolution process might be patronised in patterns, such that by following the patterns a guided evolution can be presented. Furthermore the patterns of evolution shall be formalised in a rigorous method such as Formal Methods.

REFERENCES

1. M.P. Atkinson, F. Banchilhon, D. DeWitt, K. Dittrich, D. Maier and S. Zdonik, "The Object-Oriented database System Manifesto",*ALTAIR Technical Report No. 30-89. GIP ALTAIR,, LeChesnay, France (September 1989)*
2. *D. Chamberlin, "A Complete Guide to DB2 Universal Database",Morgan Kaufmann Publisher Inc.(1998)*
3. *E.F. Codd, "A Relational Model for Large Shared Databanks",Communication of the ACM, 13(6), 377-390 (June 1970)*
4. *R. Cooper, "Object Databases an ODMG Approach", Database Technology Series, International Thompson Computer Press (1997)*
5. *H. Darwen, C.J. Date, "The Third Manifesto", SIGMOD Record, 24(1), 39-49 (March 1995)*
6. *W. Kim, "Bringing Object Relational Down to Earth", Database Programming and Design (July 1997)*
7. *NIST, "Federal Information Processing Standard Publication 184: Integration Definition for Information Modeling (IDEF1X)" National Institute of Standards and Technology (Dec 1993)*
8. *Object Management Group, "OMG UnifiefUnified Modeling Language Specification (Draft)", 1999, http://www.rational.com/uml/resources/documentation/OMG-UML-1.3.-Alpha2.pdf*
9. *C.M. Saracco, "Universal Database Management a Guide to Object/Relational Technology" Morgan Kauffmann Publishers Inc.(1998)*
10. *A. Silberschatz, H.F. Korth and S. Sudarshan, "Database System Concepts", Computer Science Series, McGraw Hill International Editions (1997)*
11. *M. Stonebraker. and P. Brown with D. Moore, "Object-Relational DBMSs Tracking the Next Great Wave", Morgan Kauffmann Publisher Inc.(1999)*
12. *M. Stonebraker with D. Moore, "Object-Relaional DBMSs the Next Great Wave", Morgan Kauffmann Publishers Inc., 1995.*
13. *J.M. Spivey, "The Z Notation - A Reference Manual 2nd ed." Prentice Hall (1992)*
14. *Stonebraker M., L.A. Rowem, B. Lindsay, J. Gray, M. Carey, M. Brodie, P. Berstein, D. Beech, "Third Generation Database System Manifesto", SIGMOD Record, 19(3) (September 1990)*
15. *P. Yugopuspito, K. Taguchi and K. Araki, "Formalizing IDEF1X in Z Specification", Proceeding of the International Symposium on Future Software Technology ISFST-1998 (October 1998)*