International Journal of Electrical and Computer Engineering (IJECE)
Vol. 7, No. 5, October 2017, pp. 2823~2832
ISSN: 2088-8708, DOI: 10.11591/ijece.v7i5.pp2823-2832

Converting UML Class Diagrams into Temporal Object Relational DataBase

Ain El Hayat Soumiya, Bahaj Mohamed
Department of Mathematics and Computer Science, Hassan 1st University, Settat, Morocco

Article Info

Article history:

Received Dec 14, 2016
Revised May 1, 2017
Accepted Aug 11, 2017

Keywords:

Meta-model
Temporal database
Time varying
UML class
Valid time

ABSTRACT

Number of active researchers and experts, are engaged to develop and implement new mechanism and features in time varying database management system (TVDBMS), to respond to the recommendation of modern business environment.Time-varying data management has been much taken into consideration with either the attribute or tuple time stamping schema. Our main approach here is to try to offer a better solution to all mentioned limitations of existing works, in order to provide the non-procedural data definitions, queries of temporal data as complete as possible technical conversion ,that allow to easily realize and share all conceptual details of the UML class specifications, from conception and design point of view. This paper contributes to represent a logical design schema by UML class diagrams, which are handled by stereotypes to express a temporal object relational database with attribute timestamping.

Corresponding Author:

Ain El Hayat Soumiya,
Department of Mathematics and Computer Science,
Hassan 1st University, Settat, Morocco,
Email: soumya.ainelhayat@gmail.com

1. INTRODUCTION

A Temporal object relational database is a collection of 1NF, non-NF, and /or typed tables. Temporal data representation has been formed with two different timestamping schemes: tuple and attribute. Tuple timestamping uses the first normal form (1NF) relations to store time-variant data. Attribute timestamping requires non-first normal-form relations, the advantage of attribute timestamping is that values of time-variant attributes are grouped together and stored as a unit in one column[1].

The literature on temporal database offers three dimensions of time for temporal data support, which are independent each to other: transaction time, valid time, user-defined time. Valid time is the period during which a row is occurred in the reality in the database, it is the fact as is valid in the modeled in the real. Transaction time automatically captures changes made to the state of time-variant data in a database [2].User-defined time is a time that users input according to their needs [3]. There is also another dimension of temporal database called bitemporal data. Bitemporal database systems support both valid time and transaction time. On the other hand, bitemporal databases model our changing knowledge of the changing world, hence associate data values with facts and also specify when the facts were valid. There by providing a complete history of data values and their changes [4]. Several temporal data models, which support different dimension of time, are discussed in previous work.

Temporal databases capture the history of object or activity of database. Since the temporal data models use the term object loosely, one we may think of the possibility of integrating non-temporal database language such OODB or ORDB. With a non-temporal database language, it is a great burden for DB users to deal with temporal data all on their own [5]. For representing time varying data from the real world into objects, class diagram of the Unified Modeling Language (UML) has become a standard of the Information Technology. UML is an industry standard language to specify, visualize, construct, and document the

artifacts of software systems [6]. It is the intention of this paper to develop a general strategy for transformation mechanism from UML class diagram into objects in temporal database based on ORDB.

This article contains definitions of a wide range of concepts specific to and widely used within temporal databases. The aims of this work is to formalize the steps involved in the transformation from UML class diagrams into temporal Object relational database(TORDB) handling valid time at the attribute. We will examine and store data using attribute timestamping. In addition, we address the issues of seamless integration with UML metamodel, and the useful features of temporal query to represent the temporal database schema. The creation of the prototype is carried out on Oracle 12C without the definition of any new tamporal clauses into SQL4, so that no study of a new query is needed.

This paper is structured as follows. An overview of the temporal object relational has been mentioned in section 3. Section 4 then describes how an existing UML class diagram is enriched with OCL to develop a meta-model of temporal database. The mapping rules between UML and TORDB have been proposed in section 5

2. RELATED WORK

Significant researches address the problem of numerous temporal models and query language. Atay compared interval-based attribute and tuple timestamped bitemporal data models, accesses and evaluated their usability using the same data and same queries for both approaches. According to this comparison, Petkoviç examined the performance implication for tuple and attribute timestamping, this test stored data using two different forms, and perform the 36 query on both.

A research work in [7] proposed a temporal object relational SQL language handling valid time at the attribute level in a temporal transparency environment his paper. The approach in [8] presented a database application development environment for both temporal and non-temporal using SQL: 2003 following the attribute timestamping. Comparison of three different storage models (OODB, ORDB, and XML) for the parametric temporal data model and estimate storage costs are discussed in [9].

The ISO (international organization for standard) and IEC (International Electrotechnical Commission) committee, initiated o project to create a language extension to support temporal database, is given in [10]. The most important features in SQL: 2011 to create and manipulate temporal database implemented by IBMDB2, is discussed in [11]. Slavimir Vesić presented a temporal concept and focused on temporal features defined in SQL: 2011 in DB2 [12]. Sandro Radovanović evaluated performance of traditional relational DBMS (RDBMS) and temporal databases using Oracle 12c DBMS [13].

From the all-overhead results, we conclude that most proposed solutions contain limited and simple rules compared to our work, notably regarding the scope and methodology. We believe that studies are based on time-variant data schema to make records of data, not even completed to provide a comprehensive data model; especially at the representation level. Also, they did not contribute to any formal procedure for the realization of mapping methods. Our study is using the concept of “profiles” that give the possibility to use extensions of meta-models based on stereotypes to extend classes. The stereotypes can be mapped into temporal object relational with SQL language. It requires the implementation of a predefined set of rules depending to the type of required in time variant data. In Figure 2, we summarize and illustrates clearly the completeness of our mapping strategy, that include entities, objects, annotation, relationships between classes, attributes in their various forms, data types, value types, class constructs, constraint types and much more.

During our criticized analysis, we feel that some aspects of time variant data was overlooked in many works, which reflect that the challenge for those authors was only to cover the technical part of the storage and retrieval rather than on gain from the offered advantages in temporal object relational. In addition, Articles about the transformation from UML class to temporal database are not as frequent.

The goal of our work is not mainly to create better temporal objet relational database from UML class diagram, but to afford a well arranged and a complete as possible transformation, that can be a concrete reference for further investigations and works in this common area.

We create our solution by combine several results from the existing methods and apply our enhancement using meta-model approach. More precisely, we propose the rules that facilitate the transformation from UML class diagram into temporal database based on ORDB. Therefore, this study presents a meta-model to define set of stereotype for the specification of new characterization of the valid time associated with UML class diagrams, in order to make a correct description of a data structure.

3. TEMPORAL OBJECT RELATIONAL DATABASE REPRESENTATION

Object-Relational Database (ORDB) is increasingly popular as the database storage. Its popularity is based on its ability to capture the object-oriented modeling semantic and the maturity of relational implementation [14]. ORDB emerged to incorporate the object technology to the relation databases, allowing the treatment of complex data type and different type of relationships. To enhance OR database and give the correct description of attribute, we associate time at rows. This model is developed to answer the limitation of object relation model. Temporal data means that the data are defined to have some time-related information associated with them [15].

Different new extension of SQL allow user to define new structured data type according to the required data types for each system. Structured type can be used as the type to define a table or as the type of column. This kind of table is called typed table corresponds to the Object type. In this paper, several temporal concepts will be added to on ORDB database. A temporal table is formally defined as follows: **T={Name, UDT, Constraints},** where name is the name assigned to the table.UDT is the corresponding type from which the table is based. Constraints like primary key or check constraints must be considered because they define characteristic and access path to the objects.

A user defined type (UDT) is a structured type having state, behavior and relationship to other type [mapping UML class]. In this sense, an UDT can be defined as follows: **UDT={name, period, Att},** where name is a name of use defined type. Period means the UDT has an open-close interval time (start and end attributes) representing a valid period. And att is a set of UDType's attributes, which its data type can be predefined type (integer, date,...), user defined constructed (collection, REF, Row,...) or reference . Attributes representation has been formed with two different levels: non-temporal attribute, and attribute history or attribute timestamping.

4. CASE STUDY AND MODELING APPROACH

4.1. UML Class Diagrams:

UML has a very rich notation offers many aspects of software engineering and applications development. It is a standard language for object oriented analysis that is able to specify a wide range of object oriented concepts by modeling a database schema. In addition, UML provides mechanisms that enable new kinds of modeling elements to be defined, and also enable to relate the information to new modeling elements. This is accomplished by integrating stereotype, constraints and tagged values. It defines several graphical diagrams in terms of the views of system. It has been widely used in database design process for the conceptual, logical and physical representation. In this work, we focus on the use of class diagrams for this purpose. Class diagram includes many elements to model corresponding to the system requirements. We have selected the more commonly used for database design; it is composed of properties, class, association, etc. Different types of relationships between classes can be established: aggregation, composition, association, inheritance.

Figure 1 presents the class diagram modeling data of employee Management System in a business company. This model will be used in the examples presented along this paper.

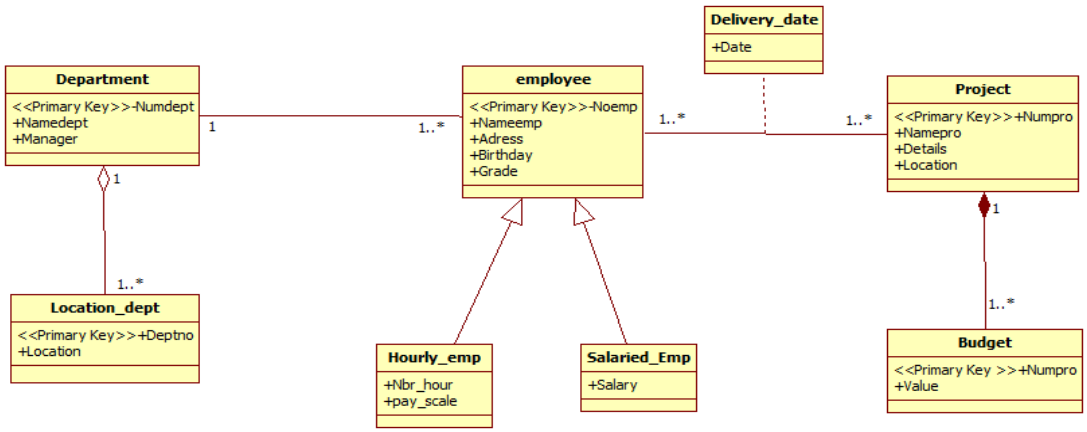

Figure 1.Class diagram for a management employee system

4.2. Meta-Model for Temporal Database

We need in this work to develop an application, which task is to keep records of changes of data in past, present or in the future. This application maintains the history of employees with salary, department and project information. The problem that occurs in non temporal databases is that only current state of data is memorized. This problem can be solved using varying time features. The temporal concepts links time of event or a fact with a time, the need to describe clearly the event when it has occurred in real time to be true or valid. With temporal database, the time varying of employee, project and the department where he works is captured. For example: we can keep track of the specific period of time during an employee has hired in company on February 2, 2001 to now. This Period can be represented as the set of all time points and historical data from its start to now. That employee have worked in financial department during fourth years, he has worked from 01/02/2001 until 10/11/2005. An employee changed his department and he received an additional 6% salary increase when her department changed.

Although the class diagram is a general purpose language for visual modeling, temporal database modeling has not been addressed. Therefore, it is necessary to develop a new Meta-Model for temporal database operation modeling. A Meta-Model is involved for each database and system engineering to get a better comprehension of a data model , because it describes the structure, relationships, constraints and semantic of data. This paper proposes an approach using UML extension mechanism to define a new set of UML model elements that represent the previous class diagram including varying time features. This Meta-Model give the possibility to understand the structure of temporal databases and construct schema translation, which facilate the migration into temporal database based on ORDB. This new model is enriched by OCL expression. OCL is a modeling language used to specify the constraint of elements within a model [16].

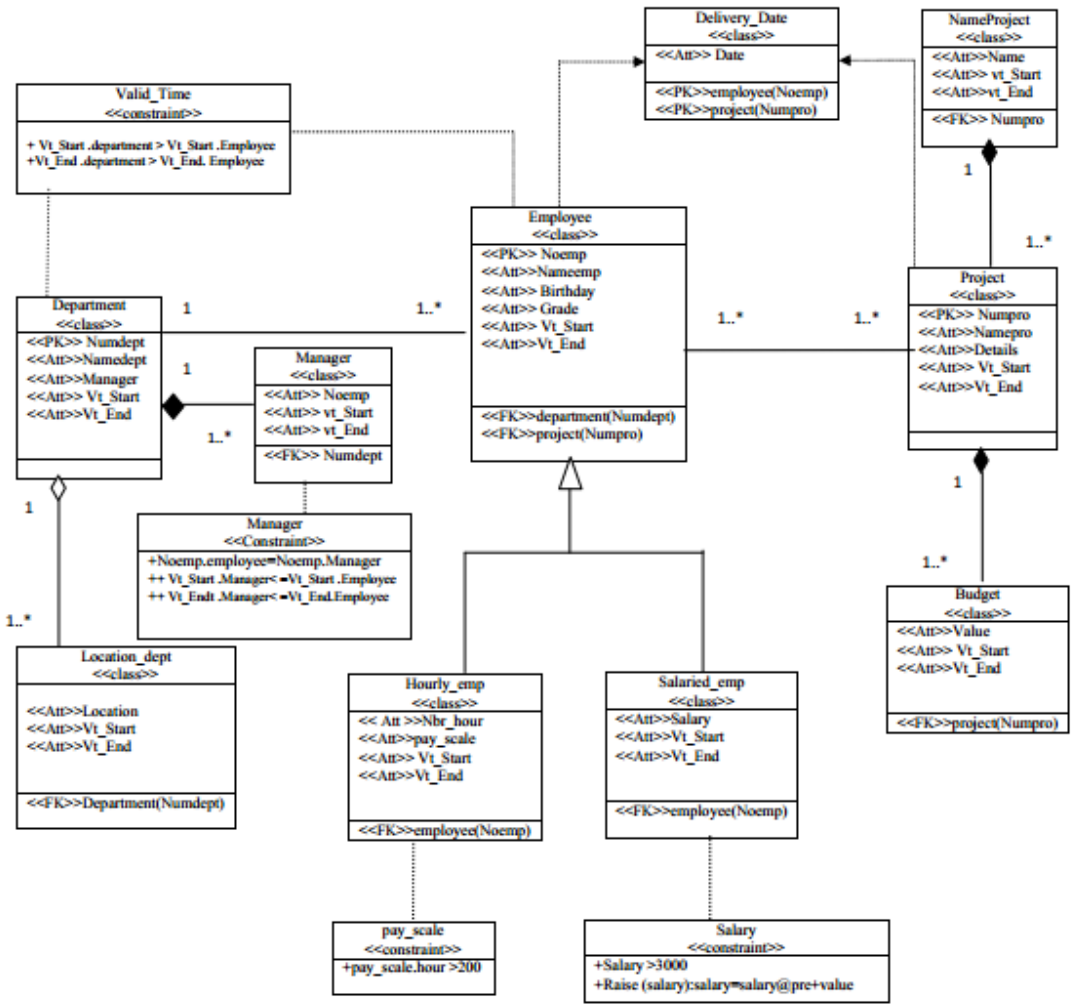

Figure 2. UML profile for management employee system enriched with temporal data

5. PROPOSED METHODS TO MAPPING UML INTO TEMPORAL ORDB

Temporal ORDB is a collection of tables each of which is can derived from any structured type or a typed table. User-defined type includes a list of attribute definitions: attribute and attribute timestamping. Time-varying attributes must be non-atomic values. Also, typed table has a row called REF. The REF is a data type which contains the reference of another typed table. The existence of this column can replace the concept of the foreign key in the relational data model. In this section, the rules of the migration of the UML into TORDB will be discussed. We integrate oracle's concepts of nested tables to create the time varying columns.

5.1. Association

Association is a relationship between 2 classes indicating that, at least one side of the relationship knows about the other side. In TORDB, we propose a method of maintaining the reference type (REF) as collection in the 'one' side with varying Time period. Transformation result is:

5.1.1. 1:N Association

Definition 1: For two classes namely C1 and C2 contain valid time Period. If C1 and C2 has 1:N association relationship(REL), this Relationship is translated in UDT1 and UDT2 corresponding to C1 and C2, where UDT1 has an attribute time stamping store collection of REF value referencing to an object in UDT2 with valid time Period (Vt_Start,Vt-End). Transformation result is: **UDT_1=[UDT_1 .Name, Attributes, vt-start,vt-end, NT(Ref(UDT_2), vt-start, vt-end)] And UDT_2=[UDT_2 .Name, Attributes, vt-start,vt-end]**

Example 1: Classes Employee and Department have 1: N association relationship (see Fig 1). This typical example is adapted to suit our valid time support at the attribute timestamping level. As shown in tables 1 and 2, the time varying information of employee and departments where he works is stored. In these two tables, each valid time period is a closed-open interval [Vt-start, Vt-End).

Table 1. The Employee Temporal Table

No Emp	Name	Birthday	grade			VT_Start	VT_END	Department		
			grade	VT_Start	VT_END			Dept	VT_Start	VT_END
1	Hajar	10/04/1986	engineer	15/02/2007	22/11/2015	15/02/2007	31/12/9999	1	15/02/2007	31/12/9999
			Manager	23/11/2015	31/12/9999					
2	Mohamed	02/06/1956	engineer	15/07/1998	08/10/2006	15/07/1998	22/11/2015	1	15/07/1980	22/11/2015
			Manager	09/10/2006	22/11/2015					
3	Amine	24/08/1976	Account	03/05/2004	12/09/2006	03/05/2004	31/11/2011	2	03/05/2001	12/09/2006
			Commercial	13/09/2006	31/11/2011			4	13/09/2006	31/11/2011
4	Ilyas	30/01/1979	Manager	20/12/2005	31/12/9999	20/12/2005	31/12/9999	3	20/12/2005	29/11/2012
								4	30/11/2012	31/12/9999
5	Imane	31/05/1975	Manager	03/05/2004	31/12/9999	03/05/2004	31/12/9999	3	03/05/2004	31/12/9999

Table 2. The Department Temporal Table

Numdept	Namedept	Manager			VT_Start	VT_END
		Noemp	VT_Start	VT_END		
1	Computer	2	06/06/2010	22/11/2015	15/07/1998	31/12/9999
		1	23/11/2015	31/12/9999		
2	Accounting	5	03/05/2004	31/12/9999	03/05/2004	31/12/9999
3	After-sales	3	20/12/2005	29/11/2012	20/12/2005	29/11/2012
4	Marketing	3	30/11/2012	31/12/9999	21/12/2010	31/12/9999

5.1.2. N: N Association

Definition 2: For two classes namely C1 and C2 contain valid time Period. If C1 and C2 has N:N association relationship(REL) in C3, implement C1 and C2 as UDTs ,where UDT1 has an attribute time stamping store collection of Att.C3 and collection of REF value referencing to an object in UDT2 with valid time Period (Vt_Start,Vt-End). Transformation result is: **UDT_1=[UDT_1 .Name, Attributes, vt-start, vt-end, NT(Ref(UDT_2), Attribute ,vt-start, vt-end)] And UDT_2=[UDT_2 .Name, Attributes, vt-start, vt-end]**

Example 2: Classes employee and project have N: N association relationship (see Fig 1). The references of one class together with the relationships will be mapped as a collection in another class table with closed-open interval [Vt-start, Vt-End).

Table 3. The Employee Temporal Table

NumProj	Namepro			Details	VT_Start	VT_END
	Name	Vt-Start	Vt-End			
1	Payment Management	15/05/2006	01/01/2007	Creation of payment management application web	15/05/2006	01/01/2007
2	Employment Management	30/10/2013	01/02/2014	Integration of a module in an erp source	30/10/2013	01/04/2014
	HR Management	02/02/2014	01/04/2014			

Table 4. The Project Temporal Table

No Emp	Name	Birthday	grade			VT_Start	VT_END	Project			
			grade	VT_Start	VT_END			Projet	VT_Start	VT_END	Date_delivey
1	Hajar	10/04/1986	engineer	15/02/2007	22/11/2015	15/02/2007	31/12/9999	1	15/06/2006	15/12/2006	30/12/2006
			Manager	23/11/2015	31/12/9999			2	01/12/2013	31/02/2014	31/02/2014

5.2. Aggregation

An aggregation relationship is a binary association that specifies a whole-part type relationship .The part is shareable and independent from the whole, where each part component (C2) can be a part of more than whole component (C1). In addition, aggregation does not imply any restriction over the life of C2, if shared part could be included in several whole, and if some or all of the wholes are deleted, shared part may still exist. In TORDB, to meet the requirement, we use the collection of UDT with varying time period in the whole table.

Definition 3: For two classes namely C1 and C2 contain valid time Period. If C1 (UDT_1) can be composed by more than one shareable and existence-independent C2, implement C2 as UDT collection attribute with Valid Time Period (Vt-Start, Vt-End) in table C1. Transformation result is: **UDT_1=[UDT_1 .Name, Attributes, vt-start,vt-end, NT(UDT_2)] And UDT_2=[UDT_2 .Name, Attributes, vt-start,vt-end]**

Example 3: Type Department is the aggregation of type location department (see Figure 1). The latter type can still exist outside type department, probably in another class. The aggregation will be mapped as a collection of UDT including Valid Time Period (Vt-Start, Vt-End) as attributes.

Table 5. The Department Temporal Table

Num dept	Namedept	Manager			VT_Start	VT_END	Dept_location		
		Noemp	VT_Start	VT_END			Location	VT_Start	VT_END
1	Computer	2	06/06/2010	22/11/2015	15/07/1998	31/12/9999	Casablanca	15/07/1998	31/12/9999
		1	23/11/2015	31/12/9999					
2	Accounting	5	03/05/2004	31/12/9999	03/05/2004	31/12/9999	Casablanca	03/05/2004	03/06/2010
							Rabat	04/06/2010	31/12/9999

5.3. Composition

It is a special kind of aggregation in which the part components are physically included in the whole. A composition relationship is an association that specifies a whole-part type relationship, but this relation is stronger than the aggregation, due to the part life depends on the whole existence. The part must belong to a unique whole, and it can be explicitly removed before removing its associated whole. So in TORDB, we need to exclusively define the part component inside the whole. For this reason, we use the nested table collection. A nested table is a table can be stored within another table.

Definition 4: For two classes namely C1 and C2 contain valid time Period. If C1 (UDT_1) composed by C2 in a composition aggregation, implement C2 as a collection of multiple attribute from C2 with Valid Time Period in table C1. Transformation result is: **UDT_1=[UDT_1 .Name, Attributes, vt-start, vt-end, NT (Atributes, vt-start, vt-end)]**

Example 4: class Project is the composition of class Budget (see Figure 1). The composition type will be mapped as a collection of multiple columns from budget, can be placed into a single column in project table.

Table 6. The Project Temporal Table

Num Proj	Namepro			Details	VT_Start	VT_END	Budget		
	Name	Vt-Start	Vt-End				Value	Vt-Start	Vt-End
1	Payment Management	15/05/2006	01/01/2007	Creation of payment management application web	15/05/2006	15/05/2006	100000	15/05/2006	15/05/2006
2	Employment Management	30/10/2013	01/02/2014	Integration of a module in an erp source	30/10/2013	01/04/2014	50000	30/10/2013	30/10/2013
	HR Management	02/02/2014	01/04/2014				50010	01/03/2014	01/04/2014

5.4. Inheritance

In Practice, the inheritance is very important and easy type of relationship. For the creation of types that represent the inheritance, we add **under** for the sub class, the keyword **not final** if the type has subtypes, and **final** if the type has no subtypes. Furthermore, Additional properties of subtype are defined in the usual way with time varying features. For example, salaried employee is a full time employee inherits class employee with attributes timestamping. The details are illustrated in the following tables:

Table 7. The Employee Temporal Table

No Emp	Name	Birthday	grade			VT_Start	VT_END
			grade	VT_Start	VT_END		
1	Hajar	10/04/1986	engineer	15/02/2007	22/11/2015	15/02/2007	31/12/9999
			Manager	23/11/2015	31/12/9999		

Table 8. The Salaried-Employee Temporal Table

No Emp	Name	Birthday	grade			VT_Start	VT_END	Salary		
			grade	VT_Start	VT_END			Salary	VT_Start	VT_END
1	Hajar	10/04/1986	engineer	15/02/2007	22/11/2015	15/02/2007	31/12/9999	5000	15/06/2006	15/12/2007
			Manager	23/11/201	31/12/9999			5200	16/12/2007	31/02/2010
								10010	01/03/2010	31/12/9999

6. TRANSFORMING THE EXAMPLE FROM UML INTO TEMPORAL ORDB QUERY

This section describes the schema definition of the employee management in a business company example Figure 2, using the commercial database oracle 12C. With the studies presented in the previous sections, it can be able to produce temporal queries for relationships. The temporal queries formed as shown in Figure3:

```
Create type t_location as object (
Location  Varchar(20),
Vt_Start Date,
Vt_End  Date)/

Create type NT_deptlocation as object (
Location_dept t_location )/
Create type Loc_dept is table of  NT_ deptlocation ;


Create type NT_Manager  as object (
EmpNo Number,
Vt_Start Date,
Vt_End  Date)/
Create type Manager_dept is table of  NT_Manager ;

Create type t_Department as object(
Numdept NUMBER ,
Namedept VARCHAR(20),
Manager  Manager_dept ,
Vt_Start Date,
Vt_End  Date,
Dept_location  Loc_dept)/

Create table Department of t_Department  CONSTRAINT dept-PK PRIMARY KEY(Numdept), NESTED
TABLE Manager STORE AS Manager_tab, NESTED TABLE  Dept_location STORE AS Location_tab;

Create type NT_Name  as object (
Name VARCHAR(10),
Vt_Start Date,
Vt_End  Date)/
Create type Name_proj is table of  NT_Name;

Create type NT_Budget  as object (
Value VARCHAR(10),
Vt_Start Date,
Vt_End  Date,
Create type Budget_pro is table of  NT_Budget;


Create type t_project as object(
Numproj NUMBER ,
Nameproj  Name_proj ,
Details VARCHAR(20),
Vt_Start Date,
Vt_End   Date
Budget   Budget_pro )/
Create table project of t_project  CONSTRAINT proj-PK PRIMARY KEY(Numproj), NESTED TABLE
Nameproj STORE AS name_tab, NESTED TABLE Budget STORE AS Budget_tab;

Create type NT_dept as object(
Dept REF  t_Department,
Vt_Start  Date
Vt_End   Date)/
Create type dept_emp is table of  NT_dept;

Create type NT_project as object(
project REF  t_project,
Vt_Start  Date
Vt_End   Date ,
Delivery_date  DATE)/
Create type proj_emp is table of  NT_project;
```

```
Create type NT_grade  as object (
Grade Varchar(20),
Vt_Start  Date,
Vt_End  Date)/
Create type grade_emp is table of  NT_grade ;

Create type t_employee as object(
NoEmp NUMBER ,
Name VARCHAR(20),
Birthday DATE,
Grade Grade_emp ,
Vt_Start  Date,
Vt_End   Date,
Department  dept_emp
project   proj_emp) Not Final /

Create  table  employee  of  t_employee  CONSTRAINT  emp-PK  PRIMARY  KEY(NoEmp),  NESTED
TABLE   Department STORE AS dept_tab, NESTED   TABLE   grade STORE AS grade_tab, NESTED
TABLE  project STORE AS proj_tab;

Create type NT_Salary  as object (
Salary Number,
Vt_Start  Date,
Vt_End  Date)/
Create type Salary_emp is table of  NT_Salary ;

Create type NT_Hourly  as object (
Pay_scal Number,
Hour Number,
Vt_Start  Date,
Vt_End  Date)/
Create type Hourly_emp is table of  NT_ Hourly  ;


Create type T_Salaried_emp UNDER t_employee(  salary  Salary_emp  ) Final;
Create  table  Salaried_emp  of   T_Salaried_emp  UNDER  employee  NESTED  TABLE  salary  STORE  AS
salary_tab ;

Create type T_hourly_emp UNDER t_employee(hourly    Hourly_emp    ) Final;
Create  table  hourly_emp  of   T_hourly_emp  UNDER  employee   NESTED  TABLE  hourly  STORE  AS
hourly_tab ;
```

Figure 3. Result of the migration into Temporal ORDB

7. RESULTS AND ANALYSIS

Our study simplifies the conversion from UML class diagram as input into Temoporal database using ORDB concepts.The transition is done with methods that extract various functions from UML class diagram , which based on it logical data structure and data behavior. From these informations, we have provided our meta-Model representation by adding varying time features and more semantic metadata.On the other hand; we have associated a time at attribute to get the correct description of records. One advantages of Meta-model is based on its ability to overcome the complications thus occur during matching keys, time varying attributes ,and promotes the visualization of navigational path between objects. In the next step, we have proposed the rules that facilitate the creation of physical schema of a temporal database.also, we have generated the main queries for the model using SQL4 standard.

This solution encourages non temporal database users to produce temporal reports or temporal queries according to their requirements and abilities. Furthermore, our implementation is considered the most essential and critical process in the development of an application supporting temporal data, to be made it is necessary to begin by transforming the conceptual model (Uml class Diagram) into logical Model, and logical model into physical Model. This work is the first step to implement an algorithm for converting method from UML class diagram into temporal database, which not requires any human interference.

8. CONCLUSION

This paper has proposed an approach for using UML as the basis for a comparative analysis of migrating to temporal object relational database. The solution is considered as a complete study that which shows how we can maintain the collection semantics stated in the conceptual model into the implementation using temporal object relational database. The study is based on a set of methods and rules depend on the UML class specifications.The obtained results were encouraged and especially regarding the special cases that was not proposed on existing works, like relationships between classes. These new methods can gain benefit for specific cases such as tables with varying time aspects and temporal query, etc. Our subsequent work will be focused on this approach to develop a framework to allow the automatic mapping from temporal relational database into temporal object relational database.

REFERENCES

[1] D. Petković . "Performance Issues Concerning Storage of Time-Variant Data".Egyptian Computer Science Journal". 2014;Vol. 38.
[2] PI Santosa, "Cost and benefit of information search using two different strategies., *TELKOMNIKA Telecommunication Computing Electronics and Control*, vol. 8, no. 3, pp. 195-206, 2010.
[3] R.Shuxia, Z.Zheng,Z. Xiaojian."An Indeterminacy Temporal Data Model based on Probability".Telkominca. 2013;vol 11 N°11;pp:6686-6692.
[4] Atay, C. , " A Comparison of Attribute and Tuple Time Stamped Bitemporal Relational Data Models", Proc. of the Int. Conf. on Applied Computer Science. 2010; p: 479-489.
[5] Chau, V.T.N. and S. Chittayasothorn. "A Temporal Object Relational SQL Language with Attribute Timestamping in a Temporal Transparency Environment" . Data & Knowledge Engineering. 2008; vol 67, p. 331-361.
[6] A.Hidayat,VG.Utomo, "Adaptive Online Module Prototype for Learning Unified Modelling Language (UML)".IJECE. 2016;vol6 N°6;pp:2931-2938.
[7] AR Ahlan, HT Sukmana, "An alternative method for determining critical success factors of information system project., *TELKOMNIKA Telecommunication Computing Electronics and Control*, vol. 12, no. 3, pp. 665-674, 2014.
[8] T. Chau and S. Chittayasothorn, "A Temporal Compatible Object Relational Database System", The IEEE Southeastcon. 2007.
[9] Noh, SY., Gadia, S.K. & Jang, H. J. Cent. "Comparisons of three data storage models in parametric temporal databases". Journal of central south university . 2013;vol 20; pp 1919_1927.
[10] ISO/IEC 9075-2:2011, Information technology - Database languages - SQL - Part 2: Foundation (SQL/Foundation).2011.
[11] K.Kulkarni, JE. Michels. "Temporal Features in SQL:2011" . SIGMOD Records. 2012;Vol. 41 No. 3.
[12] S. Vesić, S. Babarogić, N. Aničić. "Use of the Temporal Concepts in Transaction Database". PROC SYMORG. 2014;pp 850-857 .
[13] S. Radovanović, E. Milovanović, Nenad Aničić."Performance Evaluation Of Temporal Features Defined In Oracle 12C Database". PROC. SYMORG.2014;pp 858-866 .
[14] D.Taniar, E. Pardede , J. Wenny Rahayu. "Composition in Object-Relational Database". Encyclopedia of Information Science and Technology, First Edition IGI Global. 2005; pp488-494.
[15] SJ Putra, AR Ahlan, M Kartiwi, "A Coherent Framework for Understanding the Success of an Information System Project., *TELKOMNIKA Telecommunication Computing Electronics and Control*, vol. 14, no. 1, pp. 302-308, 2016.
[16] Lo, C.M., Hung, H.Y. "towards a UML profile to relational database modeling".Appl. Math. Info. Sci. 2014 ;vol 8(2);pp733-743.