جامعة الحسن الأول
UNIVERSITÉ HASSAN 1ER

كلية العلوم والتقنيات سطات
FACULTÉ DES SCIENCES ET TECHNIQUES SETTAT

Université Hassan 1er

Centre d'Études Doctorales en Sciences et Techniques & Sciences Médicales

Faculté des Sciences et Techniques Settat

THÈSE DE DOCTORAT

Pour l'obtention de grade de Docteur en Informatique

Formation Doctorale: MAI

Spécialité: Informatique

Sous le thème

Efficient Conceptual Modeling of varying Time Data and adaptative And Approaches into various environment : Object-Relational Database,XML and NoSQL

Présentée par :

Soumiya AIN EL HAYAT

Soutenue le: **02 Décembre 2021**

A la Faculté des Sciences et Techniques de Settat devant le jury composé de :

Pr.ZBITOU Jamal	**PES**	École Nationale des Sciences Appliquées, Tanger	Président
Pr.CHIHEB Raddouane	**PES**	École Nationale Supérieure d'Informatique et d'Analyse Des Systèmes, Rabat	Rapporteur
Pr. GHERABI Noreddine	**PH**	École Nationale des Sciences Appliquées, Khouribga	Rapporteur
Pr. EZZATI Abdellah	**PES**	Faculté des Sciences & Techniques, Settat	Rapporteur
Pr.MARZOUK Abderahim	**PES**	Faculté des Sciences & Techniques, Settat	Examinateur
Pr. BAHAJ Mohamed	**PES**	Faculté des Sciences & Techniques, Settat	Directeur de thèse

RÉSUMÉ

L'évolution rapide de développement des nouvelles technologies pour la gestion des données massives, et l'intérêt des entreprises à adopter des nouvelles techniques dans des différents environnements, constituent un enjeu majeur pour la réingénierie des données historisées. Plusieurs méthodologies sont mises en œuvre pour la modélisation, la migration et le partage des informations, afin d'améliorer les fonctionnalités des systèmes de base de données existants. Le processus de réingénierie consiste à identifier les différents composants d'un système. Cette étape cruciale et nécessaire dans le processus des migrations des bases de données.

Les bases des données temporelles ont attiré un ensemble important des chercheurs. Elles consistent à historier les données qui se reposent sur les concepts de temps dans le contexte d'analyses des données destinées aux managers pour élabore leurs stratégies décisionnelles. En Revanche, il y a un manque de littérature concernant l'utilisation des applications temporelles. À cet égard, l'objectif principal de notre thèse est d'examiner la flexibilité de la base de données relationnelle objets pour supporter les aspects temporels et valider la capacité d'adoption d'une nouvelle norme pour avoir un système solide de gestion de touts types de données.

Notre approche apporte une solution pour la modélisation, le stockage et la manipulation de données temporelles en utilisant des différents systèmes de gestion de données telles que RDB et NoSQL. Ce travail de recherche s'inscrive dans le contexte de la migration des bases de données relationnelles d'objets temporels. Pour éviter le risque de la perte d'une masse très importante des données, il est préférable d'enrichir et de convertir un tel schéma pour qu'il soit utilisé par de nouveaux systèmes comme NOSQL, et les partager à l'aide des méthodes de sémantique web tel que XML. Nous proposons un modèle qui préserve et améliore le schéma des bases de données relationnelles et objets relationnelles existants pour établir un pont technologique vers les différents applications. Ensuite, on définit un schéma de la transformation enrichi par des données sémantiques, pour qu'ils se convertissent d'une façon dynamique. Un prototype a été implémenté réalisant la migration automatique des différents types de bases de données prouvant l'efficacité de cette approche. Par conséquent, Notre méthodologie se base sur d'autres méthodes de la conception afin de développé une nouvelle modélisation de données temporelles en utilisant des mécanismes UML, y compris OCL, pour spécifier les contraintes et les restrictions relatives aux temps.

Mot clés: Temporal Database, TORDB, TRDB, UML, OCL, XML, Temporal Datawarehouse**,** varying Time Data, Migration

ABSTRACT

The rapid development in information technology to manage a large volume of data, and the interest of companies in securing benefits for the new environments has made information system re-engineering an active research area. Several methodologies are implemented for modeling, migrating and sharing the information; in order to improve the functionality of existing database system. The re-engineering process requires identifying and understanding all the component of such system.

The concept of temporal database has gotten much attention from researches. Meanwhile, there is lack of literature concerning the practical application of temporal database. This thesis investigate the examination the potential of object relational database to support Temporal features and the ability of the adoption of standard that is essential for increased portability , flexibility and constraints preservation.

Our approach contributes a solution for creating, storing and handling varying time data using Object relational database. This research work deals with the migration of temporal object relational database. Yet, it has a limitation to support complex types. Instead of throwing away a large amount of data, it is more appropriate to enrich and convert such schema to be used by new systems. We propose a solution that offers automatic migration of TORDB as a source into conventional and recent database technologies as a target such as NOSQL and sharing data into web semantic using XML files. Therefore, this solution provides a methodology for producing a new modeling of varying time data using UML mechanisms including OCL to express the constraints and restrictions dependent on the time. Thus, research on the migration is not fully developed.

Keywords: Temporal Database, TORDB, TRDB, UML, OCL, XML, Temporal Datawarehouse**,** varying Time Data, Migration

ACKNOWLEDGEMENTS

After all these years, I have finally the space to thank, after Allah the greatest and merciful, all those persons who were with me during this adventure.

First of all, I would like to thank the jury members for reviewing my thesis; your comments helped me to achieve this work.

I would like also to thank my adviser, Mohamed BAHAJ, who for all these years supported me as much as possible, even when things seemed to be very complicated to solve, He was there regardless of our own expertise. I will also thank all the members of the LMIET Laboratory.

I would like then to thank my family for their encouragement and prayers for me. Without support of my parents, brother as well as my sisters and their family, this work could hardly have been completed.

Finally, thanks to all of you, you know who I am, you know how you helped me, and you know that everyone of you has a special place in my heart.

TABLE OF CONTENTS

LISTE OF FIGURES

LISTE OF FIGURES

LISTE OF TABLES

LIST OF ABBREVIATIONS

TRDB	Temporal Relational Database
TORDB	Temporal Object Relational Database
OMG	Object Management Group
UML	Unified Modeling Language
OCL	Object Constraint Language
TXML	Temporal Extensible Markup Language
SGML	Standard Generalized Markup Language
OID	Object Identifier
TDW	Temporal Data Warehouse
W3C	World Wide Web Consortium
DTD	Document Type Definition
C_Schema	Class schema of UML
NVTM	New Valid Time Model
NBTM	New Bitemporal Time Model
TXSDM	Temporal XML schema Data Model
TX-OR	TXML Model for TORDB
S-TORDW	Star temporal Object Relational for datawarehouse
SW-TORDW	Snowflake temporal Object Relational for datawarehouse
TEER model	Temporal entity-relationship

GENERAL INTRODUCTION

I. GENERAL CONTEXT

Over the last decades, the Companies over the word are operating in very dynamic and complex areas that need from their managers the ability to make proactive decisions, in order to improve the quality and to increase the gain of their business. Meaningful data are required to achieve the specific goals, and are involving in analyzing and decision making. To ensure the availability of information generated by company's activities, it is necessary to use the various technologies. The database management systems (DBMS) are nowadays a common technology of everyday work in several environments where they are applied that allows it easy for enterprises to centralize the information, store data efficiently, and provide data access for application. They are among the most important tool for business applications. The growth of databases technologies day by day attracts organization investments due to its fast evolution and importance.

The emergences of novel databases systems have created challenges for companies and organizations to respond quickly to the demand impacted by this change take advantage of the benefits of new technologies in order to manage work more flexibly and efficiently. As a natural result of these evolution and requirements, Forward engineering has become a vibrant field for significant researchers and practical issue. It represents a method to transform the existing systems into new database systems to realize quality improvement in functionalities and operations that applies on the stored data. It can be defined as a process of discovering how a database system works. It involves a wide range of tasks to understand, convert, and redesign the existing system. This process starts by identifying the semantics structure, components and their relationships, and translating them into conceptual layer representing by entities, attributes and relationships. The obtained design-level can be used to provide new systems or redesign an existing database to meet new requirements. However, database systems cannot be easily replaced. It is very hard to re-write database applications every time the user wants to switch to the new technology. The system re-engineering is very complicated method. The main reason behind this solution is to solve the problem of the conversion from traditional to more recent environment in order to avoid throwing away a large volume of structured data

The most conventional databases are based on traditional relational database where they have been far successful in handling large amount of data.Today, the increasing popularity of new object relational, NoSQL systems, Business intelligent applications and XML technologies can be considered to be among the most significant recent revolution in information technology. These novel management database systems and information technologies have been dominant in the data management environments due to their productivity, flexibility and extensibility.Moreover, the conventional databases are only able of storing and querying the current perception of reality and relationship among objects, such database systems remove the old data values when the data is updated. Time is an important aspect of all real-world phenomena where the Events occur at specific points in time. The ability to model this temporal dimension of the real world is essential to many computer applications. Unlike the existed databases systems, a temporal database is capable of storing evolution of data, thereby allowing managersto maintain and examine complete object histories.

On the other hand, an Object relational database has more potential than traditional database because it has a relational technology base and appends object-oriented features. The ORDB concepts has the ability to support historical object by defining our temporal object in order to store all the changes dependent on the data value over the time. The efficient implementation of object-relational applications handling temporal data has received considerable attention, especially in business and commercial area. Improving object-relational applications performance has been a serious challenge for scientific and database researchers.On the other hand, Database migration is very necessary process in order to encourage organizations to move to new systems adopting temporal features.

II. SCOPE & MOTIVATION

Several reasons have led to the investigation described in this dissertation. Many companies have stored their data in conventional database and aspire to take advantage and adopt the databases systems that have emerged in the last years. In such databases, the recent data value is only recorded. When new data values are available through update insert and delete statement, the old data values are removed from these databases permanently. Although conventional databases serve some applications well, they are insufficient for those applications in which the data values historyare required to be retrieved rapidly. Hence, instead of discarding existing conventional databases on building a novel systems on top of them, it is generally suitable and beneficial to incorporate the varying time features and convert existing database into a new environment in order to discover which database is more appropriate to move to and which

system is able to support Temporal features to provide an opportunity for experimentation and comparison among alternative database technologies. Temporal information extraction and converting is complex subject that needs carefully designed.For this reason, the temporal Applications need a persistent storage of complex data and the capability to query the information arbitrarily and efficiently wherethese features are not supported by conventional database systems appear: user-defined Type and operators on the complex data. All these characteristics are specific to object-relational database systems.Then, there is need for methodologies that deal with integrating varying time data that are based on object relational model which combines concepts given by relational database and the oriented object features.

Therefore, implementing the migration between different database management systems, especially the database that manipulates the temporal features is far from easy. Several research questions need to be resolved before leading to systems that provide adaptation of temporal data storage and retrieval. Moreover, all research on the generation of ORDBs based on varying time is focused on diverse areas of handling historical data. But any work has covered a solution for the migration of object relational database associating time properties to the attributes into another Target schema. In addition, none of the existing proposals can be considered as a method for migrating from different models based on varying time management features. On the other hand, this could help further increase the acceptance of such newer and richer databases among enterprises and companies and to select the most effective strategies to achieve their objectives.

The main questions this research seeks to answer are:

- What are the main concepts of temporal databases?
- How temporal databases are currently implemented in different environments?
- How to manage and query the historical data effectively and efficiently?
- How Object relational database supports temporal database? Why?
- Which of the new Technologies is most appropriate for temporal Database?

III. SUMMARY OF THE MAIN GOAL AND CONTRIBUTIONS

The general aim of our research is to preserve the historical data that are associated to the time at attribute level. We are particularly interested in object relational database with temporal concepts.

Although many important insights and results have been reported to explain the storage and query processing using SQL: 2011 concepts, many research challenges still remain in temporal database management such as the migration method.

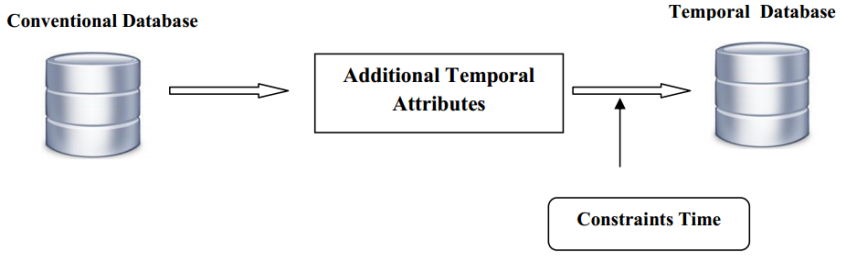


Figure 1. Migration from Conventional Database into Temporal Database

We propose a solution which will be offered to the problem of the migration of temporal database between different models such as TRDB, MongoDB and web semantic. Thus, solve different weaknesses and limitation that are outlined previously.

In this research, we exploit the semantic enrichment techniques in order to facilitate the implementation of our framework. We claim that that an integrated methods can be developed based on the definition of schema translation that enrich the object relational database and models with additional specifications and take into account the features and the characteristics of the target schema in order to evaluate which the model is more appropriate to support temporal data according to the required functionality, performance and suitability.Also, the proposed approaches help us to develop a framework for automatically convert an existing temporal databases into different target schema.

To achieve our goal, we propose the following objectives:

1. To survey existing researches related to temporal database storage and querying process based on Object relational Databases, by analyzing their capabilities and limitations
2. Produce the schema translation of the various management data systems enhancing with temporal features and using enrichment semantic method
3. Include temporal features in XML documents to share and exchange the historical data over the web.
4. Implement an algorithms and prototype to validate our solutions

IV. OUTLINE OF DISSERTATION:

The organization of this manuscript can be summarized as follows:

The first chapter will be the subject of a state-of-the-art presentation that addresses the concepts of temporal object database as well as different definitions of conceptual techniques, management data systems and web semantic. In the Next, A review is of existing works related to temporal databases and object relational database are analyzed to determine the problem to be resolved. The following chapters are the published and submitted works.

In the second chapter, the logical schema is produced and the rules of the migration are formalized using UML features with the definition of restrictions dependent to the time which are expressed by OCL language. The main goal is to develop a conceptual layer to design temporal data and it requirement in order to promote the understandability of different dimensions and advantages offered by temporal systems. The defined model will be adopted along of our thesis to facilitate the conversion process.

The focus of chapter 3 is the migration of temporal relational database into temporal object relational database. A categorization will be presented of selected works in the literature, involving the manipulation of temporal data concentrating on SQL: 2011 characteristics and object relational database advantages which are discussed and critically evaluated. In the next section, we will define a Database migration from the source (TRDB) into TRODB that introduce two basic phases, including the semantic enrichment and schema translation describing in detail how to identify TRDB model and TORDB constructs, and how to classify the relationships and temporal features between different entities.

In the fourth chapter, a review of existing approaches to include the temporal elements and data in XML files, considering their capabilities, weaknesses and limitations. After that, a solution is dealt with the problem of the creation and sharing temporal object relational database using TXML files, by providing schema translation and algorithm is implemented to facilitate the conversion process.

In the fifth chapter, we will produce the transforming model from temporal object relational database into new systems to handle a large volume of data. The Big amount of information cannot be processed by conventional databases. We will discuss the possibilities to create temporal model for data warehouse solutions by incorporating temporal object relational concepts. On the other hand, the Nosql present one

of the most popular technologies of big data which can make records and handle the data in the distributed environments. The main challenge is to find a good balance between characteristics of temporal database using object relational database management systems presented by oracle and opportunities offered by NoSQL database management systems. Our goal is to evaluate the ability and the capabilities of new applications in order to store and retrieve the history of data with efficient manner.

V. PUBLICATION

5.1 Published Articles

- ✓ S. AIN EL HAYAT & M. BAHAJ . "Converting UML Class Diagrams into Temporal Object Relational DataBase". International Journal of Electrical and Computer Engineering (IJECE). 2017. (Scopus).
- ✓ S. AIN EL HAYAT & M. BAHAJ. "Migration of the temporal RDB into temporal ORDB including Bitemporal Data: Phases". Transactions on Machine Learning and Artificial Intelligence, 5(4). 2017. (Index Copernicus).
- ✓ S. AIN EL HAYAT & F.Toufik & M. BAHAJ . "UML/OCL based Design and the transition towards Temporal Object Relational Database with Bitemporal Data".Journal of King Saud University - Computer and Information Sciences. 2019. (Elsevier).
- ✓ S. AIN EL HAYAT & M. BAHAJ. "Modeling and Transformation from Temporal Object Relational Database into Mongodb: Rules.Advances in Science" . Technology and Engineering Systems Journal (ASTESJ). 2020 . (Scopus)

5.2 Conferences

- ✓ S. AIN EL HAYAT & M. BAHAJ. "Migration of the temporal RDB into temporal ORDB including Bitemporal Data: Phases".ACMLIS(2017). Tetouan.Morocco
- ✓ S. AIN EL HAYAT & M. BAHAJ. "Converting Temporal Relational database into Temporal Object Relational Database".AIT2S (2017). Tanger. Morocco. (Springer)
- ✓ S. AIN EL HAYAT & M. BAHAJ. "Conversion of a TXML Schema to Temporal Object-Relational Database Using Bitemporal Data". ICITM (2018). Oxford.United Kingdom.(IEEE,Scopus)

- ✓ S. AIN EL HAYAT & M. BAHAJ. “A Temporal Data Warehouse Conceptual Modeling and its Transformation into Temporal Object Relational Model” .AI2SD.2018). Tanger. Morocco.(Springer, Scopus ,DBLP)

- ✓ S. AIN EL HAYAT & M. BAHAJ. “The Storage of Data from TXML document into Temporal Object Relational Database”.ICDS (2019).Marrakech. Morocco.(IEEE,Scopus,DBLP)

- ✓ M.RAJAALLAH, SA.CHAMKARA, S.AIN EL HAYAT. “Intrusion Detection Systems: To an Optimal Hybrid Intrusion Detection System”.AIT2S 2019.Mohammedia. Morooco. (Springer)

CHAPTER I:

BACKGROUND AND RELATED WORKS

I. INTRODUCTION

This chapter provided an illustrative background to the main concepts of different types of databases related to the topic of the thesis, in order to get a better understanding of this dissertation and the process of the temporal object relational database migration. It includes introductions to several technologies and tools, the advantages and disadvantages of each model. Finally, we presented the main challenges in the field of modeling and handling object relational database systems.

II. CONCEPTUAL MODELLING WITH UML AND OCL TOOLS

Data modeling is the important phase in the process of database design and the systems development. This step is considered to be a high-level and abstract design activity, also called a conceptual design. The conceptual model presents the specifications of software systems in the form of diagrams and relationships. It can be considered as an activity related to capturing the knowledge about the desired system. According to [1], "the conceptual schema of an information system is the specification of its functional requirements."

2.1 What is UML?

Unified Modeling Language UML is the de-facto standard in industry for designing software systems. It was developed in the mid-1990s as a collaborative effort by James Rumbaugh, and Jacobson. In November 1997, UML was accepted by the Object Management Group (OMG) as a standard modeling language. Although UML is most often associated with modeling Object-Oriented Software applications, it has a much wider system due to its inbuilt extensibility techniques. UML was designed to incorporate current best practice in modeling technologies and software engineering. It is important to realize that UML does not give us any kind of modeling methodology [2]. UML is not dependant to any specific

Methodology or life cycle, and indeed it is able of being used with existing methodologies. It has been playing an increasingly important role in software systems and dominates object-oriented modeling. Now, it is a very mature design language. UML can be used for business modeling, software modeling, and general modeling of any construction to describe both a static structure and dynamic behavior. It had proved its value in thousands of software development projects worldwide. On the other hand, The Object Management Group (OMG) defined several tools for helping the software engineer using UML and this list reflects the high reputation of UML as a modeling language. The UML specification defines a number of basic diagrams that are based on three aspects of the system in the form of classes, packages and their relations. They provide the high level design details of the system. The design process supported by UML starting by analysis data helps forward engineering as well as reverse engineering. Therefore, how to develop UML based environment for software development is a hot research issue.UML has always provided many options about how a particular model element may be displayed, and not all of those will be supported by every modeling language.

2.2 UML Diagrams:

UMLDiagrams are the graphs that design the contents of the system. They are used to specify the structure of the objects, classes and their components. Also UML diagrams promote the modeling of the connection between different entities and classes. UML offers several diagram types that are used in combination to produce all views of the application. There are two broad classifications of diagrams and they are divided as follow:

- ✓ Structural diagrams
- ✓ Behavioral diagram

In this work, we will focus on one of the important structured diagram that is called Class Diagram. A class diagram is the most diagrams used in the conceptual modeling. It represents the static structure of classes and their relationships in the system. Class diagram basically provides the object-oriented view of such application. Classes can be related to each other in a number of ways: Association, Aggregation, Composition or Inheritance. All these relationships described in a class diagram along with the internal structure of the classes in terms of attributes and operations.

2.3 OCL

The Object Constraint Language (OCL) was introduced at IBM as a language for business design. The developer's ability to use OCL is very important. It is considered as a formal language used to express constraints. OCL is an adopted standard of the Object Management Group (OMG) and mainly used for specifying properties of a model that cannot be expressed in the diagrammatic notations of UML. The OCL can supplement some of the shortcomings of UML modeling by providing the expression of textual and declarative requirements of conceptual schema. UML/OCL as OO Specification Languages is one of the most widely used diagrammatic object-oriented modeling languages in the industry, which can help formalize the semantics of the language itself and to facilitate UML users to express precise restrictions on the data and the structure of models. Constraints specified in OCL help to restrict UML models [3] but they also increase maturity level of a UML model [4]. Therefore, OCL does not create any new object in a class diagram but completes the meanings of the existing objects. Also, an OCL specification always conforms to the OCL meta-model.

III. VARYING-TIME MANAGEMENT DATA:

3.1 An Overview:

In the recent decades, temporal database is one of the most important parts of the information technology. The temporal database was developed in 1993 and implemented in 1994. It is generally known that temporal database stores the history of the objects or the database activity. A temporal database is a database that contains time-varying data and offers built-in support for modeling the temporal dimension of the data. The storage time is one of the most necessary properties to characterize an attribute. The varying time database is able to track when an event started and when it ended. Even today, a large number of database system based on time in nature to make a correct description of data by recording the database activity and their changes over time. Hence, the need to retain trace and audit the change made to a data and the ability to plan based on past or future assumptions are important uses cases for temporal data [5].Temporal databases capture the history of object or activity of database. The ability to model this temporal period of the real world is necessary to many computer applications in various domains, such as econometrics, banking, inventory control, accounting, law, medical records, land and geographical information systems, and airline reservations. Applications such as these rely on varying time database

that record time referenced data to make a history of all object changes.Therefore, Historical information can be stored systematically and in uniformed manner using temporal databases [6]. Historical which need past occurrences in an organization data is widely analyzed for a lot of purposes such as making data decisions.

On the other hand, Conventional databases provide the state of an enterprise at one moment of time. Although the stored data continue to change as new information is added, these changes are considered as modifications to the state, with the old out-of-date information being deleted from the system. In such applications the attributes involving time are manipulated solely. The temporal database management system (DBMS) manipulates dates as values in the base data types.When using a temporal database, retrieving information about the past is supported by included query functions which makes the development and databases more efficient and potentially increases the performance [7].

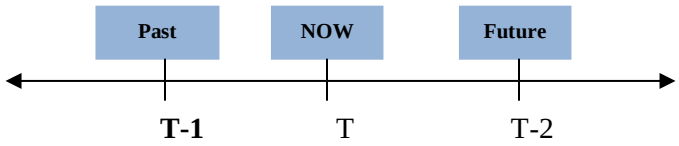


Figure 2. Temporal Database Interval

Therefore, data manipulation statements in temporal database (INSERT, UPDATE and DELETE) are handling in same manner as ordinary tables. However, in temporal systems, user has the permission to control dates and times of PERIOD columns and can refer to any date and time (past, present or future).To insert or update a value, additional syntax is defined without throwing the old value, which allows row splitting. That means if there exists record in the system, with predefined period and insert or update query have been executed, then the stored value will be updated from records end time to update query executing time . Also, the new additional row will be created in which the start attribute takes update statement beginning time in order to maintain integrity of data. On the other hand, the dalate query removes records that satisfy time period condition, but inserts in database additional two rows where, one update records beginning time by Delete statement beginning time and another update statement end time by records end time.

In the glossary of varying time management databases, the concept of temporal attributes is added to include multiple time dimensions as well as multiple data models. Temporal databases support 3

Dimensions called valid time and transaction time. These two dimensions are orthogonal and can be supported separately, or both can be supported in concert. Valid time and transaction time can be merged to provide bitemporal data.

3.2 Valide Time

Valid time presents the time at which an event was true in the real world. It can be in the future, if it is known that will be true at a specified time in the future.The temporal tables including valid time period are defined in SQL: 2011 standard and associated with the PERIOD clause statement. It consists of two predefined date-time columns, one for the beginning of a period and another for the end of a period. A valid-time period can be specified during the creation table or alter table process. In order to prevent the user to define two or more records with same value, the PERIOD can be added as primary key. Valid time tables are intended for meeting the requirements of applications that are interested in capturing time period during which the data is believed to be valid in the real world. A typical example of such applications is an insurance application, where it is necessary to keep track of the specific policy details of a given customer that are in effect at any given point in time [8].A primary requirement of such temporal applications is that the user has the ability to set the start and end times of the validity period of rows, and he is free to assign any time values, either in the past, current or in the future, for period attributes. On the other hand, user is permitted to update the valid period of the rows as errors are discovered or new information is made available. Also, Users can choose any name they want for the name of the period as well as for the names of columns that act as the start and end columns of the period .Any table that contains a period definition with a user-defined name is an application-time period table[9]. Additionally, one constraint is added, which disallows creating a record where end time is lower than beginning time. The data types of the period start and end columns must be either DATE or a timestamp type, and data types of both columns must be the same.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 from 11/11/2005 until now. All this changes can be recorded in temporaldatabase with valid time period columns in order to preserve history in such company.

3.3 Transaction Time

As Shown previously, valid time represents a period of the time when a fact is true in reality. Unlike valid time, Transaction time identifies when data was asserted in the database. It refers to the time at which a transaction has been processed to update the database. If transaction time is added to table, the states of the database at all previous points of time are retained. It models the database reality, storing exactly when row has been inserted, modified, and removed in the database. Also, the transaction time period cannot extend into the future, as it is impossible to update the past. This means, transaction time table must be created with two additional attributes, one for start period and another for end period. These new columns cannot be modified by the user. In other words, system can only add or change values of these fields [10]. The transaction-time start period is the time when the database became aware of a row, when the row was recorded in the database. With the insertion statement the transaction attribute start columns takes current time as value. On the other hand, the transaction time end period reflects when the fact was closed by an update to the row, or when the row was deleted from the database. This means, update or delete statements close end time of updated record and create new row with start time of current date.

The transaction time period uses also closed-open period interval. At any given point in time, a row in transaction table is regarded ascurrent system row if the start time period of that row defines as current time. The values of start and ad end columns are assigned and updated automatically by the database system.

3.4 Bitemporal Data

Bitemporal data support the transaction time and valid time periods. IT is like version control for your data. Bitemporal data are storing current and historical data and this means it not only shows records as they are now in the present, but also as they were at any point in the past in order to preserve data history that exists at different times. Rows in such bitemporal tables are associated with both the valid time and the transaction time period. Bitemporal tables are very useful for capturing both the period interval during which fact is believed to be true in the real world as well as the period interval during which this fact was stored in the temporal database.For example, an employee may change the bank. Typically the account number changes legally at a specific time but it is not changed in the database currently. In that case, the transaction period automatically records when a particular account number is known to the database and the valid time period records when the account was legally effective.

IV. RELATIONAL AND OBJECT RELATIONALDATABASE

4.1 Relational Database

Relational databases were first introduced in 1970 by E.F. Codd [11]. Relational databases are based on set theory and relational algebra to record related data in a structured manner. It is a database with a relational model that can organize data in the form of table .Each table contains one or more data categories in columns. Each row contains a unique instance of data for the categories defined by the columns. Each column in a table is called an attribute. Columns specify a data type (integer, char, date) which can be stored. Each row contains a unique instance of data for the categories defined by the columns. In Relational database tables, columns can also have constraints. Constraints can be that every stored data has a unique value, or if null a default value is utilized. The primary key of table is a data item, which must be atomic. There is a unique key for each row in each table and it is possible to connect the rows of the different tables the same row key. A relationship is created through a foreign key constraint, which requires a record with the same value of that column to exist in another table. Information is retrieved from a relational database by querying a table, and defining joins on these relationships to relate data between many tables. Relational databases have been built upon as major products for technology giants, such as Oracle, and IBM. New concept sets and improvements are still being made to them as these technology giants push to satisfy more needs.

Although Relational database is sufficient for managing the storage of an important capacity of primitive data types, where their SQL is easy to use, they are not strong enough to represent real world problems and new environments. For instance, Novel application often need persistence for nontraditional data structures, such as graphics, multimedia, or voice data. RDBs have some weaknesses in supporting complex structures and data operations. Moreover, they cannot handle applications such as temporal databases, and other systems that involve complex data inter relationships. The relational data model does not have the ability to provide user-defined data types that can be specified based on pre-defined data types. Different relations between tables may not represent entities in the real world, and the inheritance relationship is not supported. The relational database model is not scalable for systems and applications needing access to many related tables, which requires joins. Joining several tables leads to inefficient query processing times [12]. In addition, it is very difficult to include new operations to the system in the relational model since it is limited to the generic SQL operations queries. RDBMS developers and researchers have spent much time in producing methods for the mapping of

complex data structures into RDBs for persistence. All these limitations have led to the emergence of new extensions called as Object Relational DBMS (ORDBM). The most important features of RDB are:

- **Keys**

In RDBMS keys concept are very important features as they are used to identify data and make relationship between tables. One of the most important properties of relational model is uniqueness of rows or records which are also provided using the concept of 'Key'. There are mainly three types of keys which are specified for RDBMSs as follows:

- **Candidate Key**: is a key which can be used to uniquely identify any record in a table without referring to any other data. A candidate key can represent a single column or a combination of multiple columns. A table can have more than one candidate key.
- **Primary Key**: is a key by which any record can uniquely be identified from a table. A table can have multiple candidate keys, but only one from those keys can be considered or chosen as a primary key [13]. Primary keys ensure the uniqueness of recorded data in a table and reduce information redundancy. It can be defined as a single column or composed by multiple columns.
- **Foreign Key**: A foreign key represents a column or set of columns in a table which refers to the primary key of another table in order to uniquely identify a record of that table and define the joins between tables.

- **Constraint concepts:**

The constraints specify some restrictions on the data stored in RDBs with help of DDL. They are the rules enforced on the data columns of a table. In RDB, the constraints are represented by implicit, explicit, and semantic and data dependency restrictions. Implicit constraints are inherent in the data model for the characteristics of relations, relationships among tuples. Explicit constraints can be defined in the schema during the creation of the table that are called integrity constraints. Integrity constraints deal with data validation process and business rules that can be stored and imposed on relational data when applications or users manipulated data. The integrity constraints are: key, entity integrity, referential integrity, domain, null, and default value. Semantic constraints cannot be specified in the data schema. However, they are expressed in application programs or data content. In the last, Data dependencies test whether or not the RDB is designed perfectly using the normalization process. Normalization is a technique used in database

design to reduce redundancy, data anomalies and poor data integrity. These constraints should be enforced by RDBMSs at each instance of insertion, updating or deletion of data to or from the tables.

4.2 Object Relational Database:

The demand to represent complex data structures has motivated the development of the object Relational database systems. ORDBs have potential because they merge relational modeling and Object Oriented concepts. The main objective of their developpement was to combine both the robust transaction and performance management concepts of relational database, and the Object oriented technology features of scalability, flexibility, and support for rich data types. The OO models offer concepts that enable a better modeling of real world problems to conceptual schemas [14]. Researchers can work with tabular relational structures and DDL with the possibility of object management. The object-relational specification extends a relational model features to integrate object Oriented capabilities such as defining objects and complex data structure that are directly supported in database schemas and in the query language. These include pre-defined, structured and collection data types, primary keys and references, inheritance, and operations. Therefore, the table in ORDB is called typed table as it can be created based on Object aspect and using pre-defined type to identify the data. Each row in typed table has an object Identifier OID, through relationships among objects are established. The rules of RDB model have been ignored from the object-relational models, so that an attribute can be defined as collection of data types. On the other hand, one of the most important advantages of ORDBs is their huge scalability, the ability of reuse and sharing. Object relational database systems designed to have large storage capacities to satisfy large companies need to manage massive data.

The most important features of Object Relational Databases are described as follow:

- **Object:** An object is one of the most fundamental concepts of the object relational model where an object represents an entity of interest in a specific application. The objects are invented to overcome the limitation of relational database in supporting complex data and real world modeling. They are defined with a number of levels of complexity and an inheritance hierarchy. Each object has a state (value), behavior (operations) and unique identifier which is used as a reference to the object in order to establish a relationship between other Objects.

• **Inheritance:** Object relational database allows the inheritance between Objects this process is known also a generalization. This is a powerful mechanism, which lets a sub-type Object inherits the attributes and operations of a super-type Object defined parent class, with additional properties. Simple inheritance is supported in ORDB via the under keyword added in the definition of the table.

• **User defined types:** this concept allows to the users to provide a complex data type according to their needs, integrating object specification, attributes and methods. The UDT can be formed using predefined types. An UDT allows values in tables to be associated with methods (encapsulation). The attributes of user defined type or UDT can be used by different objects. Besides, a table can be defined based on an UDT.

• **Reference type:** A row in object relational model can be an object that is uniquely specified by a column called identity, containing an OID, to make a difference between objects. The type of this column is a **REF**, which is used for relationship participation. REF plays the same role as foreign key to express the relationship between tables.

• **Collection type:** Object relational database has the ability to store more than one value in the same row. It supports collection types which represent multi-valued attributes as a single type. A collection type is expressed by the keyword that determines the type of the collection and the element data type. ORDB support these two collection types:

- ✓ Varray: The varray is a collection type that allows the user to embed homogenous data into an array to form an object in a pre-defined array data type [15]. Also varray type, has a limited size to store the values where must be ordered.

- ✓ Nested table: this type allow duplicated values and accept unordered values. A nested table is a collection type that can be stored within another table. With a nested table, a collection of multiple columns from one table can be integrated into a single column in another table.

V. TEMPORAL DATA WAREHOUSE: AN OVERVIEW

Before the emergence of Data Warehouses, Developers created reports and analysis by straight queries into the operative systems. Data in these systems usually are stored in relational databases which

serve these needs. It is important when the database is directly queried with a real-time data. Moreover, straight query can causes some problems where Analysis operation of recorded information in the real time requires large volume of data from the operative system. Also, the retrieval data for reporting can be distributed between many environments. This can lead to the serious issues that have impact to the performance of the operative system .For this reason, Data warehouse is developed to support reporting and making analytics. A data warehouse can be defined as a database which stores data from several sources and presents them in an integrated structure that is suitable for effective decision-making support. A data warehouse (DW) collects large amounts of data from heterogeneous data sources and transforms them in order to make this information available and be used to analysis the behavior of company. The main goal of the analysis and reporting phase is to provide the management of an organization with information on trends and facts that are required for making a new strategy.Hence, Data warehouse is best described by Inmon [16] as: “a subject oriented, integrated, non-volatile and time-variant collection of data in support of management’s decisions”. Datawarehouseis based on the multidimensional model, which defineinformation as facts that can be analyzed along a set of dimensions, composed of levels conforming to aggregation hierarchies. The basic multidimensional model assumes that only facts evolve in time and this is materialized by the link(s) of the facts with the time dimension [17].

Therefore, temporal database and data warehousing are two separate environments that are strongly related: data warehouseis the commercial product that need temporal database features. Trend analysis can go along many dimensions, the most important of which is time [18]. It is used to identify different characteristics in the evolution of data, over time or over various geographic points or over product lines. With the temporal evolution data warehouse is very often required not only to hold a reformatted subset of current operational data, but also to maintain a history of this data.Furthermore, Temporal Data Warehousehas the same elements and components as non temporal one, called, dimensions, hierarchies, facts, and measures related on system requirements and the availability of this data in the source. The additional concept is that TDW associates a time period to facts, typically representing valid time, and makingtrack of the evolution of dimensions, facts, and measures. TDW supports the different dimension of time presented previously, namely, valid time, transaction time. In addition, a temporal DW should allow both temporal and nontemporal aspect and features in the same system.

VI. NOSQL DATABASE

The term NoSQL was invented first by Carlo Strozzi in 1998 for his Strozzi NoSQL opensource relational database. This database was relational, but the term has evolved and since 2009 it is more known for non-relational or non-ACID databases. Although, NoSQL is a vague term that also includes rich categories like key-value and Graph databases, it's commonly used to refer to systems that focus on simplifying the design to more easily achieve horizontal scaling and high availability, which usually means dropping the relational model. Besides, Th non-relational or NoSQL databases are schema-free [19], and allow storage of diffrent data formats without prior structural declarations [20]. To achieve scalability and availability of data, NoSQL database usually relaxes the consistency guarantee on data they store. Therefore, they were developped for massively scalable web applications. They are designed for distributed storage and parallel computing and so try to overcome the weaknesses of relational databases with management of massive amounts of data. The most interest characteristics advantages of NoSQL databases are efficient processing, effective parallelization, scalability and costs.

There are many types of NoSQL databases which might vary greatly in features. A way to categorize NoSQL databases is related to their data model:

- **Key-value Databases**

Key-value databases were the first and simplest Nosql databases, they are more basic. The data structure of the key-value stores is clear and easy to understand which does not need many schema restrictions. This kind of databases stores key/value pairs, which can be of different types and are organized in sets where Keys have to be unique . The exact data types that are supported depend on the properties of the object. Therefore, the key value database doesn't focus on how data is related to each other. Any new data, no matter what the structure is, has the possibility to be stored at any time without affecting existing data items and the availability of the database system. Also it can easily be distributed among different clusters and can be accessed very fast via keys. Key-value databases are based on the primary key access. But, they are efficient for relatively simple operations which can be done by the keys. However, it will take more time to develop complicated data structures and relations adopting this NOSQL model.

- **Column Databases**

Are also known as columnar databases that are based on the concept of grouping closely related data into one extendable column [21]. Logically, data is organized in tables with rows and columns. But the column data model considers this concept as a limitation in conventional databases. Instead of storing a single table's properties in one entity, Column database records the properties of many objects that belong to a column separately. In addition, a row acts as container for several columns. The rows in the column family store do not need to have the same number of columns and the columns in each row can be different. Therefore, Columns in a family are logically dependant to each other and are physically recorded in the same entity by grouping columns with similar characteristics into the same family. A column family is dealt with a row that includes many columns and can be identified by a unique row key, where each row can store any number of key-value pairs. This data model is mostly based on schema, and therefore the data types are predefined.

Examples of such databases: Accumulo, Cassandra, Druid, HBase, Vertica.

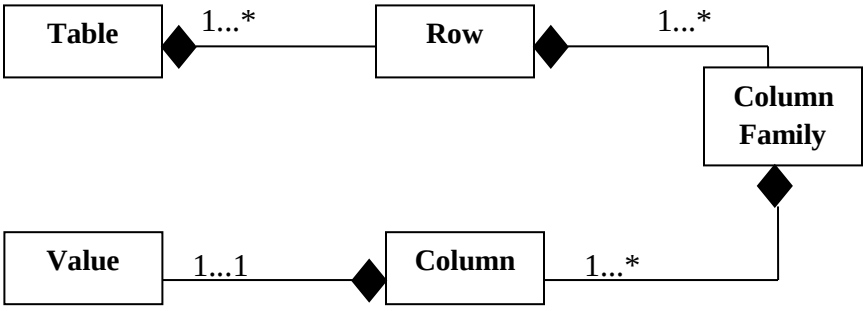


Figure 3. Column Database Meta-Model

- **Document Databases**

The document databases are not document management system. A document-oriented database is a modern technology to store data document rather than simple rows and columns. In NOSQL document store, the smallest element is a document. Document is similar to SQL record. It is a self-representation of a column. Document stores structured sets of key/value pairs in some variation XML or JSON. All the keys are identified as a unique in each document. For this reason a particular key is added in each document which allows storing a key reference to another document, establishing a relationship between the both. One of the advantages of this Nosql model is the ability to store a new document that contains a massive data, and to integrate new properties and information of any length into the existing document

that can be done easily. Hence, a set of documents are grouped in a collection. The document oriented databases are convenient for data integrating and schema migrating. As opposed to the column database, the document data is a schema-less and any data type can be defined in JSON or XML. This kind of databases is flexible and does not focus on the data structure. Therefore, they are efficient and helpful for rapid development and handling complex data structures.

Examples of such databases: MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB.

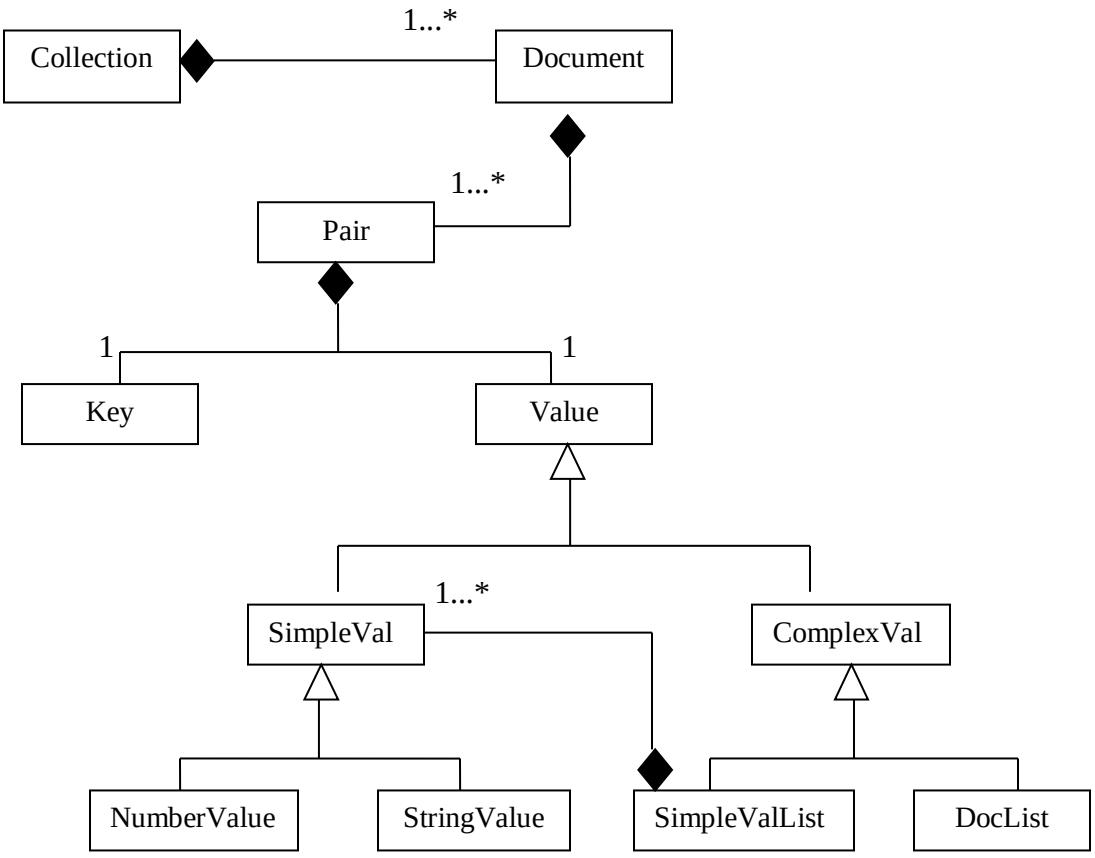


Figure 4. Document Database Meta-Model

- **Graph Database**

Graph Database is considered as a special type of database, where graph elements are used to represent interconnectivity or topology of unstructured data. Graph Databases provide a flexible graph model that can scale across multiple servers and applications. Typical of these kind applications are social networking and recommendations, network and cloud management, master data management, geospatial, bioinformatics, and security and access control [22]. Compared to other database Models, they are a little

different though by comparing the characteristics and how Graph databases preserve the unstructured data. The Graph database applies graphs, which consist of nodes and edges, to store and manage data, instead of storing data in tables in a relational database [23]. The nodes represent the entities in the graph which contain all the information about some object. And the vertices represent the relations between nodes. Therefore, the structure of graphs provides a high accessibility and scalability in distributed systems by partitioning the graph data into a number of separate environments.

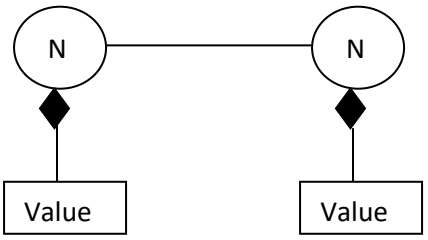


Figure 5. Graph Database Meta-Model

VII. WEB SEMANTIC:

According to the World Wide Web Consortium (W3C), The Semantic Web provides a common framework that allows data to be shared and reused across applications, enterprises, and community boundaries [23]. It can be defined as a collection of technologies that aim for representing semantic meanings in a language that computers can understand. The Semantic Web is an extension of the current Web in which information is given a well-defined meaning, better enabling computers and people to work in cooperation in order to facilitate their activities (research, e-business, e-commerce….). The Semantic Web designates a set of standards for describing content of W3 resources that is accessible and usable by adopting software or agents, through a system of formal meta-data, using in particular the family of languages developed by the W3C consortium. The word semantic means that the meaning of information in the Web can be known, not only by humans, but also by machines. Furthermore, The Semantic Web can enable the processing of the data to infer well defined meaning for a better communication between computers and people. It play an important role to make data to become Smart or intelligent through the use of ontologies. The intelligent data means the information of the web can be so richly interconnected which help the machine to be more able to infer as humans.The Semantic Web represents a novel solution in which both machines and men can search, read, understand and use data across the Web so that they can interpret Web resources and respond more appropriately to user requests. Therefore, The Semantic

Web was implemented by, Tim Berners-Lee, as an extension of the existing web for the purpose to formalize information and automated services in order to provide the content of the Web resources re-usable and sharable by automated techniques.

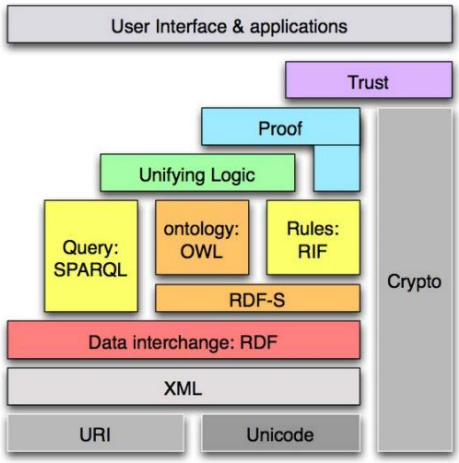


Figure 6. General Web Semantic Architecture

7.1 XML Model

Not all data can be handled in structured databases. Data might be collected in databases where they do not have to be constrained by the schema. These types of data are called semi-structured data, which they need a specific Model is called XML. XML has become a very important data representation model. It is considered as a tool for specifying the semantics of the data. The Extensible Markup Language (XML) is the universal format for structured documents and data on the Web [25].Is is a text-based markup language for structured documents, which is a subset of the Standard Generalized Markup Language (SGML). It is designed to facilitate interoperability with SGML and HTML .This meta-language concept is becoming the standard f for W3C in order to interchange the data over the Web. Therefore, XML document is composed by several constructs such as namespaces, elements, attributes, tags and values. An element represents a unit of XML data enclosed by tags which can group several elements. A tag is used to describe elements or data that is surrounded by brackets. An attribute is integrated on the tag to provide further information. Values occur as instances of elements/attributes. On the other hand, XML allow to the user to create his own markup language file for describing a specific purpose. Users have the ability to define their own tags elements and values. XML is self-describing according to which documents can be structured into complex levels of data and schema. However, XML files consist of schema and

data that can be combined in one document. XML is a powerful model because it extends simple defined tags to more complicated structures and relationships such as aggregation and inheritance. The schema in XML represents the data structure and restrictions using one of the XML schema languages. In XML, there are two predominant languages have been proposed by by W3C, and XDR by Microsoft: Document Type Definition (DTD) and XML Schema. The use of the one of the languages depends on its capability to fit the application requirements. Even though, DTD is less expressive than XML Schema specifications. XML Schema offers flexible name space support, which is a significant advantage over using DTD.

7.2 XML schema Language

The structure of an XML document can vary each time for this reason there are several languages are designed to express the same information. XML Schema language is a standard that provides a sophisticated means for describing the structures and constraints of XML schema and instance documents [26]. The W3C recommended the use of the XML Schema language in May [27,28]in order to overcome the weaknesses of DTDs and implement a more description of XML document content. It formalizes the definition of the structure, content and semantics of XML instance documents. With XML Schema concepts, it is possible to modelthe representation of datain XML instance document, and the relationship between the different elements. Also the XML Schema language has support to namespaces and has the ability to specify the data types of the information and is extensible because it is based on XML-syntax. On the other hand, An XML Schema offers an important of featuressuch as key and integrity constraints, and other concepts such as inheritance, references, data collections and user-defined data types can support RDB and ORDB models to exchange the data. Moreover, it implements a wider range of built-in data types, where offers to the users the possibility to define their own simple and complex data types including restrictions and extension keywords.

The essential components of XML Schema are used to model the data below:

- Elements and attributes
- Simple types
- Complex types
- Namespaces and annotations
- Inheritance
- Identity Constraints

VIII. OBJECT RELATIONAL DATABASE MIGRATIONAPPROACH

The emergence of the novel data management systems and the need to move into an efficient environment in order to overcome the limitation of the existing system in the company lead to involve the migration of database applications. Database Migration represents a process in which all components and properties of database source systems are converted into their equivalents into another database environment. Indeed, each migration has its specifications, prerequisites and data model used. These specifications lead to propose different mapping rules for the conversion models, which in turn affect the success and quality of the process.The migration process involves the extraction of the important properties and the structure, classification of the relationship, the translating of the source database schema into the target one and converting data into the target database format.

8.1 An Overview of the Proposed Approachs:

The migration of object Relational database and the use of temporal concepts to make records of data, represent a vibrant field for significant researchers and scientifics. These studies lead us to combine the most important features of ORDB and varying time aspect in order to develop our frameworks for simplifying the conversion and the migration of historical data between different environements. Now, we will cover several approachs that are devided as follow:

- **Temporal database Definition and Manipulation:**

As we already stated, there are several works dealing with the temporal data storage and query language. Atay presented a comparison between interval-based attribute and tuple time stamped with bitemporal data models, and evaluated their usability using the same data and same queries for both concepts [29]. According to this comparison, Petkoviç's work examined the performance implication for tuple timestamping and time varying attribute, his test stored data using two different forms, and examined the 36 query on both [30]. This work in [31] introduced a temporal object relational SQL language handling valid time dimension at the attribute level in a temporal environment. Comparison of three different data storage models (OODB, ORDB, and XML) for the parametric temporal data model in order to estimate storage costs are discussed in [32]. The ISO (international organization for standard) and IEC (International Electrotechnical Commission) committee, initiated a project to provide a language extension to support temporal database, is given in [33]. The most important features in SQL: 2011 to create and manipulate relational database including temporal data, which implemented by IBMDB2 is discussed in [8]. Slavimir Vesić presented a temporal concept and focused on temporal features defined

in SQL: 2011 in DB2 [34]. Sandro Radovanović evaluated the performance of traditional relational DBMS (RDBMS) and temporal databases using Oracle 12c DBMS [35].

- **Migrating Conceptual schema into database:**

We review the existing proposals for defining conceptual Modeling level, UML is assumed in most studies as a model language for data design. Transforming conceptual models into ORDB without tempral features have been studied extensively over the past years [36][37][38][39].A UML diagram, such as a class diagram, is typically not refined enough to provide all the relevant aspects of a specification. There is, among other things, a need to describe additional constraints about the objects in the model.For this reason, other approaches dealt with the expression of data constraints by adopting OCL annotation thatallowsenriching the conceptual model structurally and semantically. The work in [40] presented a systematic procedure for the transformation of a UML class diagram formed with OCL annotations into a constraint satisfaction problem (CSP) in order to provide a predefined set of correctness properties about the UML/OCL. Identification and specification of arbitrary constraints within the bounds of OCL-Lite in a conceptual level to ensure the completeness and correctness of reasoning are discussed in [41]. Gogolla and Hilken introduced the model validator of the use tool, a modern instance finder from UML and OCL based on implementation of relational logic in order to help developers to find fault in model description [42]. Maciaszek and Wong defined a design constructs needed for the development of an object Relational database based on UML and they proposed a mapping method from design models to an object relational implementation [43]. Golobisky and Vecchiesti proposed in [44] formalization mapping steps involved in the conversion of UML into Object relational database starting with the conceptual schema wich is presented by UML class diagrams.

- **Migrating RDB into ORDB:**

Here, we show some existing proposals for transforming RDB into ORDBs. The relationship Inheritanceis one of the important features given by ORDB, it is defined in [45]. Using foreign keys or ref types in Oracle 8*i* and the under clause in Oracle 9*i*/SQL3 A method of mapping and preserving collection semantics into an ORDB has recently been proposed [46].M. Castellanos and f. Saltor developed a canonical model that converts the schema of a federated system database. This involves a knowledge acquisition process to improve the semantic level of the schema. The approach presents a methodology that enriches relational schemas by converting them to an object-oriented data model, called

bloom. It is based on inclusion dependencies, but also takes into account exclusion and complementarily dependencies [47][48].A. Maatuk proposed a semantic enrichment concept of RDB and ORDB , it is done by improving a representation of an existing database structure in order to give more details of database schema. This approach holds an existing relational database as input, obtains a copy of its metadata and enriches it with as much semantics as possible, and builds an improved representation of the relational schema (RSR) [49]. The research work in [50] deals with the passage of migrating data from a relational database to an object relational database, by developing methods of selection and insertion which is based on optimizing the extraction of information in a predefined data model, which deals with the transition from relational to object relational.

- **Migrating ORDB into XML:**

In the last decades, significant research has focused on the problem of document history management combining the issues of XML document version and varying time data management for which a numerous technologies and models have been provided. There is much current interest in representing and exchanging temporal data in temporal XML document on the web. The approach in [51] proposed a concise and comprehensive review of the methods implemented for XML based semi structured semantic analysis, this solution is composed by four logical parts describing disambiguation techniques that can benefit from XML aspect including data clustering and indexing concepts. Rizzolo and Vaisman presented a new method for modeling and implementing temporal data in XML in order to track historical information in XML document and recover the state of the XML document as of any given time by developing an algorithm for validating this technique [52]. A research study developed a framework based on archIS system using XML techniques to manipulate temporal data and temporal xqueries which are supported via clustering and indexing techniques in order to manage the historical data in RDB is discussed in [53]. G.Qadah developed two new algorithms and the associated indexing structures to perform correctly in processing interlinked and independent XML documents, also another algorithm is introduced to minimize the storage requirements [54].an XML based implementation of temporal database system for parametric data model is discussed in [55].Ying and Yuyin presented a novel approach based on indexing technique wich is separated in two parts: path index and value index [56]. Qtaish and Ahmad implemented a new framework called XANCESTOR wich uses a path based techniques and composed by 2 algorithms (XtoDB) algorithm wich maps XML document to RDB reducing the storage space and XtoSQL algorithm converts Xpath queries into their corresponding in SQL [57]. The approach in [58]

introduced a general strategy for repairing detected inconsistencies that result from retroactive updates for currency data in a valid time and Bitemporal XML database.

- **Migrating ORDB into MongoDB**

A research work in [69] proposed a generic standard-based architecture that allows Nosql systems focusing on mongo db to be manipulated using SQL query and seamlessly interact with any software supporting JDBC. Ajit Singh demonstrated data conversion to mongo db, in order to make data more interactive and innovative using the data stored in cloud [60]. The model transformation and data migration from relational database into Mongodb taken into consideration the query characteristics of each model, in addition, an algorithm to automate the migration are discussed in [61]. Another approach presented Algorithm for automatic mapping of relational database to mongodb using entity relationship (ER) Model to provide the conceptual schema and modelling relationship between the different entities [62]. A framework for mapping MySQL database to mongodb by developing an algorithm that uses the metadata stored in relational system as input is discussed in [63]. The work in [64] described a migration process from Object relational database to Nosql document database end provided a review of different proposed approach. An overview of Nosql to evaluate the scalability and efficiency in storage of data in oriented document database case study in order to show the representational format and querying management process of Mongodb [65]. The work in [66]introduced a disciplined approach called Jschema(Temporal Json Schema) for the temporal management Json documents by creating a temporal json documents from conventional document that can vary over time, the generated document uses such features of temporal management data. The mapping Process of spatio temporal disaster data into Mongodb database using the data represented by the aspects of space and time is shown in [67].Boicea, Radulesu and agapin discussed the difference between oracle and Mongodb this comparison dealt with several criteria including theorical , Concept , restrictions and query processing of SQL database get a document oriented database Management [68].

8.2 Discussion

The investigations into the problem of Object relational database including temporal features migration demonstrates that the approaches proposed so far have had different goals. Each proposal has made certain assumptions to improve the performance of TORDB and facilitate the migration process, which might be a point of limitations or a drawback. However, there are still shortcomings in the implementation of temporal database based

on object relational model data translation in a more effective manner into more than one management data system. The use of a middleware can lead to slow performance, and make the process expensive at run-timebecause of the dynamic mapping and migration of temporal data in the different environements. Although most varying time concepts are present in these proposals, their focus has been on only the technical part of storage and retrieval of data of temporal data rather than on migration. We feel that some semantic concepts of varying data are not considered. Furthermore, these studies don't give solutions that will allow providing a mapping method between different data models, and gain from the offered advantages in temporal ORDB.

During our criticized analysis, some researchers focused on the designing and mapping the conceptual level using UML into Object Relational database without including the time features to make records of data history. We noticed that Articles about the transformation from UML class enriched with OCL to temporal database are not frequent. Therefore, many studies have been devoted to the analysis and correctness of UML formed with OCL notations. Furthemore, we noticed that temporal database features were overlooked and some semantic concepts of temporal data are not considered in many works, especially in the conceptual level. Therefore, Most of the previous works presented in the literature fail because they do not provide consistent migration formalization such that these mapping can be automated using different technologies such RDB XML and web semantic incorporating varying time properties into temporal Object relational database. We conclude these previous studies don't offer a solution that allows developing a comprehensive and precise modeling for temporal database and gain from the offered advantage of Object relational model, and Nosql techniques for time-varying data management design.

8.3 Semantic Enrichment of TORDB:

In order for information to be understood and workable by different systems, and adequate for migration between different models, techniques and tools that simplify the comprehension of the database component must be used. For this reason, it is important to resort to semantic enrichment, which can be described as the mechanism of providing metadata in order to facilitate understanding, integration, and processing of information. Semantic enrichment is a process of building process in which experts system formalizes rule sets specific to a type of database in order to capture the new facts of objects and their relationships. It allows analyzing and examining a database to determine its structure and identify definitions of data meaning. This is achieved by improving the representation of an obtained structure in order to make hidden semantics at the higher level. Therefore, the success of the process is related to volume of data that can be extracted from the existing database, and the method by which an enhanced

semantic representation is produced. To do that, additional concept must be taken into account such as the classification of different entities and their properties, the specification of relationships and cardinalities, and the different constraints and restrictions including Keys.

In our approach, the semantic enrichment process involves extraction of data semantics of an ORDB including temporal features by specifying its metadata representation and improving it with required information, and producing an enriched object relational schema. This method will be applied to generate the enhanced representation of source and target databases used in this work in order to capture the important characteristics and data structure. In the next chapters, the thinking behind the construct of database schema mechanism is explained in details the main benefit of using semantic enrichment techniques, including why they are needed, what purpose they serve, and their definitions. This method facilitates migration into new different databases based on temporal object relational model.

The following schema shows the important phases of semantic enrichment system described in our proposed approach:

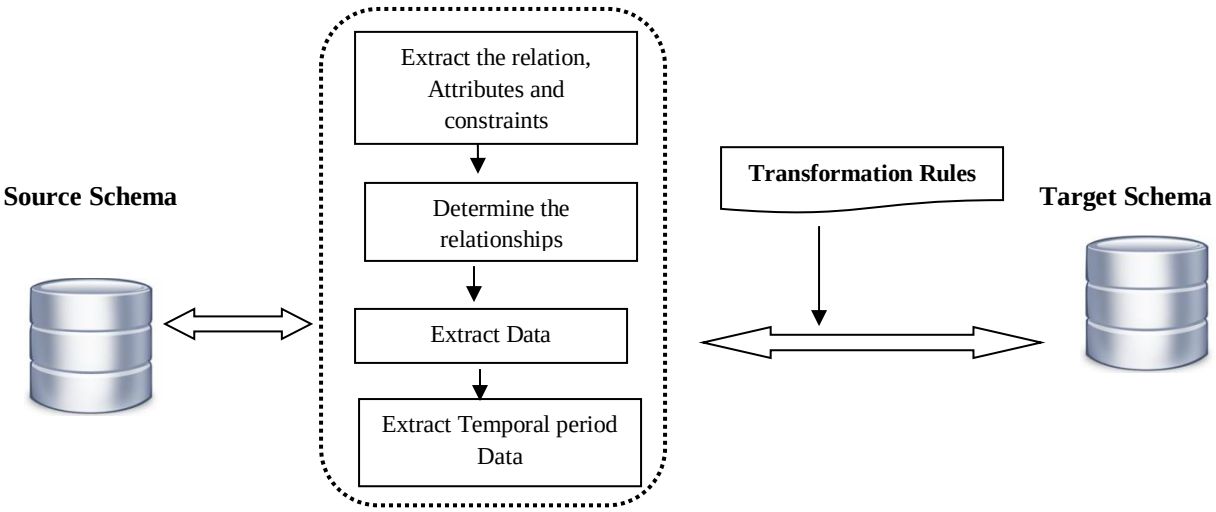


Figure7. Semantic Enrichment Process from database Source into Temporal Datbase

IX. CONCLUSION:

In this chapter we provided a survey of the basic concepts of our field of research, we focused on the notion of the different data management systems. We also presented the main goal of UML on the modeling level. In the next, we reviewed earlier approaches in Object relational database migration and some works introduced to handle time varying management data. We also detailed the challengesfor migrating and sharing temporal object relational data using XML or web semantic environment. The presented literature review was essential to understand the basics mechanisms and technologies used so far for migrating of TORDB. In addition, the aims have been to provide a comprehensive view of the problem of ORDB conversion including temporal features, to introduce various proposals in order to show how it has shaped current and future research in this area.

CHAPTER II:

CONCEPTUAL MODEL OF TEMPORAL DATABASE USING UML/OCL

I. INTRODUCTION

Nowadays, Software systems are vital part of our day life, the process of developing software becomes harder to manage and more complex to maintain. For the purpose to simplify the software building, the developers and researchers started using and introducing different mechanism to design the system. The modeling system is one of the interesting solutions to reduce the complexity of development process. The modeling system provides understandable description of the architecture and structure in a formal or semi-formal annotation and an involved activity follow which can be considered as reference of development process. The object oriented and relational systems exploit the concept of object technology and use conceptual design methods as a means of description for modeling of object relational applications.Today, UML is widely accepted as popular modeling language for different systems. UML (Unified Modeling Language) is a visual language for describing, specifying, constructing and documenting the artifacts of the system which can be applied to all applications and implemented platforms that are based in the time features in nature. In object-oriented software modeling, the UML has the ability to visually represent software models.

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 [69]. 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.

On the other hand, we assumed the most researchers don’t deal with the identification of conceptual schema for temporal systems. For this reason, we inspired of the recent and existing works for further

Investigation in this area, in order to describe the meaningful temporal information specifying the different dimension of time and give a comprehensive data model especially at the representation level.

This chapter contains definitions of a wide range of concepts specific to and widely used within temporal databases. We formalized the steps involved in the transformation from UML class diagrams into temporal Object relational database (TORDB) handling temporal attribute. In addition, we address the issues of seamless integration with UML meta-model, and the useful features of temporal query to represent the temporal database schema.

We define a mapping rules and present the essential steps involved in the transformation of UML/OCL into its equivalent in Temporal object relational database (TORDB) handling different dimension of time .We provide a precise conceptual schema (UML/OCL) for representing temporal database system handling bitemporal data. In addition, we will define syntaxes and semantics of OCL constraints for representing the complex constraints dependent on time. The proposed UML/OCL is the transformation at the conceptual level of either one of the temporal database constraints such as Triggers and constructors that can also be used to enforce complex integrity constraint. In addition, the creation of the prototype is carried out on Oracle to validate our solution.We deal with an expressive conceptual schema for temporal object-relational database representation using both UML specification and its constraints language OCL. We define the syntax of OCL expression representing the complex constraints which are dependent on the bitemporal dimension. The proposed constraints will be transformed to their equivalent in the database. In addition, our study proposes a UML/OCL design from UML class diagram annotated with OCL (UML/OCL) to define set of entities and restrictions, for the specification of the new characterization in order to manipulate the temporal attribute associated with class diagrams, and make an efficient description of a temporal object-relational database. A prototype has been implemented, we have developed an algorithm for generation of TORDB Model from UML/OCL Model, and the OCL specifications have been translated into integrity constraint, triggers and constructor to handle the complex restrictions.

II. CONCEPTUAL DESIGNING OF TEMPORALOBJECT-RELATIONAL DATABASE BY USING UML:

Conceptual modeling is the process of generating a description of the contents of a database in high-level terms and specifying its functional requirements. It can be considered as an activity for capturing knowledge about the desired system functionality. The process starts by taking as input information requirements for the systems that will use the database, and constructs a schema expressed in a conceptual design notation, such as UML class diagrams. The challenges in providing conceptual schema include designing informal and formal information requirements into a cognitive model, that describes unabiguously and completely the component of the database, and using the mechanisms of a data modeling language appropriately.

The conceptual schema represents an abstract definition of database tables and their relationships by using a human oriented natural language, independent of any implementation, respecting clarity and simplicity criteria [70]. It consists of taxonomy of classes with their attribute, their relationships and their set of constraint. Over the state of the domain which identifies conditions that each instance of schema must satisfy.

However, several methods for object-oriented analysis and design have brought a great number of languages for supporting many specifications of the applications development process. UML is one of the techniques the most used in different phases of system life cycle of. The main reason for its widespread use is that UML has a very rich notation for the modeling process. To ensure model quality, a conceptual schema is represented by UML diagrams tools with the graphical, and a set of complex constraints where are usually expressed in OCL as invariants object models. The primary goal of OCL is to augment a model with additional information and restriction that cannot be expressed in UML.

On the other hand, the concepts of transformation from the conceptual (UML) design to temporal Object-Relational models depend on the understanding of the data meaning and the structure of the database. It is required to use a conceptual model obtained and follow certain rules and phases to perform the creation of a temporal object-relational model.

2.1 Temporal Data Design with UML Mechanism

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. Therefore, 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. For representing time varying data from the real world into objects, and to overcome the complexity of designing temporal object and the navigability path between different entities, the Unified Modeling Language (UML) has become a standard of the Information Technology. It is the intention of this work to develop a general strategy for transformation mechanism from UML class diagram into objects in temporal database based on ORDB.

2.2 Improving UML by using OCL:

Several methods for object-oriented analysis and design have brought a great number of languages for supporting many specifications of the applications development process. The main reason for its widespread use is that UML has many Diagrams and tools for the modeling process. To ensure model quality, a conceptual schema is represented by UML diagrams tools with the graphical, and a set of complex constraints where are usually expressed in OCL as invariants object models. The primary goal of OCL is to augment a model with additional information and restriction that cannot be expressed in UML.

The proposed conceptual schema for temporal object-relational database uses both UML specification and its constraints language OCL. We define the syntax of OCL expression representing the complex constraints which are dependent on the temporal dimension. The proposed constraints will be transformed to their equivalent in the database. In addition, our study proposes a UML/OCL design from UML class diagram annotated with OCL (UML/OCL) to define set of entities and restrictions, for the specification

of the new characterization in order to manipulate the temporal attribute associated with class diagrams, and make an efficient description of a temporal object-relational database.

III. DEFINING THE TRANSFORMATION RULES FROM UML USING VALID TIME INTO TORDB:

The major task to be performed in varying management data is the conceptual layer. It plays an important role in interactive analysis of the temporal data in order to build a comprehensive system that dependent on the time. The conceptual models determine how to store the one or more facts from the reality to be modeled by temporal relations. Also, help us to handle the temporal data that requires specifications and more details of the temporal relations between different entities with an easier manner and can be envisioned as a sequence of data history. A Temporal relation consists of a set of classes which are related to each other. It has tree orthogonal time dimensions. In thissubsection, we will take a more general dimension to design the history states of data using valid time period, and define the rules of the transformation from UML diagram class into Temporal Object relational database.

3.1 UML Class Diagram with Temporal Data:

The most well-known language of UML is the language of class diagrams which describes the structure of a system in terms of classifiers, their behaviors, and possible relations between different objects. 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. Class diagram includes many elements to model corresponding to the system requirements. The class diagram represents the static view of an object-oriented application structure and includes many elements to model corresponding to the system requirement. It is composed of the classes, attributes, operations, constraints; and different relationships between classes can be established such as aggregation, association, composition, and inheritance. We have selected the more commonly used for database design.

Consider the class diagram in **Figure 8.** It illustrates a class diagram in banking system management including Bitemporal data in order to make records of the history of data and the transaction operations. This model will be used in the examples presented along this chapter.

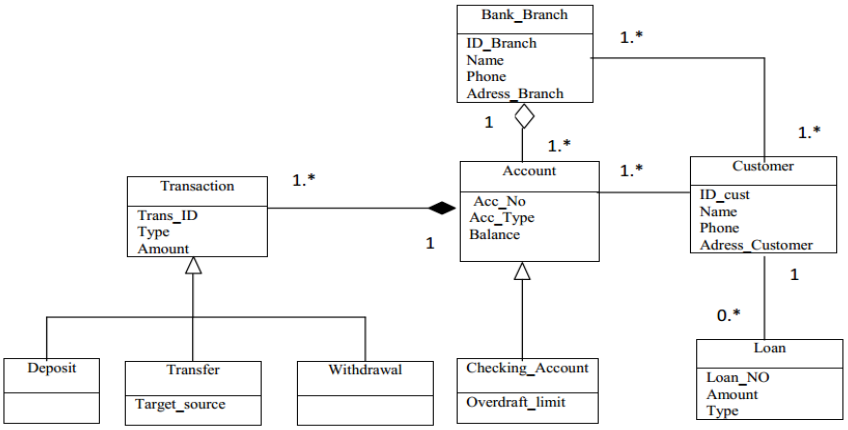


Figure 8. Class Diagram for a Management e-banking System

3.2 Meta-Model for Temporal Database:

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 langage and tagged values for describing a specific Domain. For this purpose, we concentrate on the UML class diagram to describe our system and to create the new logical design according to varying time aspects.

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 a temporal database, the time-varying of the customer, bank account and the transactionactivity are captured.For example: we can keep track of the specific period of time during a Customer has opened a new bank Account 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 customer hasrequest two Loans during fourth years, the First from 10/03/2001 until 10/11/2002 and the other from 15/09/2003 until 30/03/2004 and he received 6% increase when the type of the loan is changed. For this reason, all the financial transactions which have occurred within a given period of time on a bank account and the balance of the account at any point in time should be storedin temporal database to make a history of customer activities.

Although the class diagram is a general purpose of a conceptual and logical schema, it has several limitations for certain applications, which deal with temporal database features at the attribute level. 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 section 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.

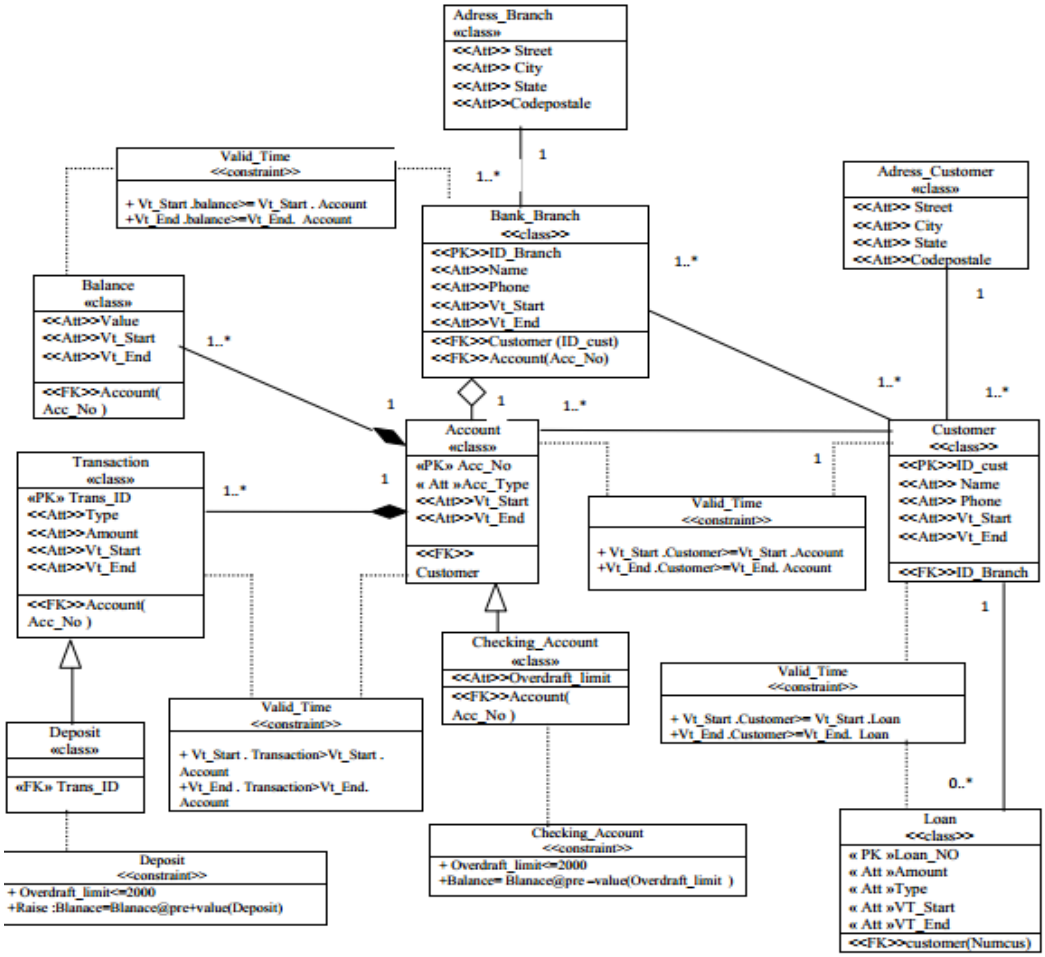


Figure 9. UML profile for Management e-banking System enriched with temporal data

3.3 Mapping Method from 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 time stamping. 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.

3.3.1 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 resultis:

- **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: Classes Customer and Loan have 1: N association relationship (see Figure 9). 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 loan history proved for the customer is stored. In these two tables, each valid time period is a closed-open interval [Vt-start, Vt-End).

Table1. Loan Table with Valid Time Period

Loan_No	Amount	VT_Start	VT_END
1	400000	01/08/2017	01/08/2018
2	23560	14/06/2014	30/05/2015
3	3000	01/01/2019	31/12/2020

Table2. Customer Table with Valid Time Period

ID_cust	Name	phone	Adress_History			VT_Start	VT_END	Loan_History		
			Adress_Ref	VT_Start	VT_END			Loan_Ref	VT_Start	VT_END
1	Nassima	06576890	1	10/05/2011	15/08/2015	10/05/2011	31/12/9999	1	01/08/2017	01/08/2018
			7	20/08/2015	31/12/9999			3	14/06/2014	30/05/2015
2	Amine	06342745	4	18/01/2014	27/02/2016	18/01/2014	27/02/2016	2	01/01/2019	31/12/2020

- **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 = [UDT1.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: Classes employee and project have N: N association relationship (see **Figure 9**). 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).

Table3. Branch-Bank Table includinh Historical data

ID_branch	Name	Phone	Adress_History			VT_Start	VT_END	Customer		
			Ref_Address	VT_Start	VT_END			Ref_customer	VT_Start	VT_END
FC456	CHI	05347865	1	15/07/2007	31/08/2015	15/07/2007	31/12/9999	1	10/05/2011	31/12/9999
			2	01/09/2015	31/12/9999			2	18/01/2014	27/02/2016

3.3.2 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: Type bank_branch is the aggregation of type Account (**see Figure 9**). The latter type can still exist outside type bank_branch, probably in another class. The aggregation will be mapped as a collection of UDT including Valid Time Period (Vt-Start, Vt-End) as attributes.

Table4. Bank branch table integrating Account data

ID_branch	Name	Phone	Adress_History			VT_Start	VT_END	Account_History			
			Ref_Address	VT_Start	VT_END			Ref_customer	Account_Type	VT_Start	VT_END
FC456	CHI	05347865	1	15/07/2007	31/08/2015	15/07/2007	31/12/9999	1	Checking_account	10/05/2011	31/12/9999
			2	01/09/2015	31/12/9999			2	Saving_account	01/2014	27/02/2014

Table 5. Account Table with historical data

ACC_NO	Account_Type	VT_Start	VT_END
1	Checking_account	10/05/2011	31/12/9999
2	Saving_account	18/01/2014	27/02/2014

3.3.3 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: *C*lass Account is the composition of class Balance (**see Figure 9**). The composition type will be mapped as a collection of multiple columns from balance history, can be placed into a single column in project table.

Table6. Account Table with balance History

Acc_No	Account_Type	VT_Start	VT_END	Balance		
				Value	Vt-Start	Vt-End
1	Checking_account	10/05/2011	30/10/2013	100000	10/05/2011	10/06/2012
				156000	11/06/2012	10/12/2012
2	Saving_account	01/11/2013	30/11/2016	50000	01/11/2013	30/10/2014
				1200	30/10/2014	30/11/2016
	Checking_account	01/12/2016	31/12/9999	4568	01/12/2016	31/12/9999

3.3.4 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, transfer class with additional attribute named target_source inherits class Transaction with temporal attributes. The details are illustrated in the following tables:

Table7. The Transaction temporal table

Transaction_ID	Amount	VT_Start	VT_END
1	3000	15/05/2006	15/05/2006
2	4000	30/10/2013	01/04/2014

Transaction_ID	Amount	VT_Start	VT_END	Target_Source
1	3000	15/05/2006	15/05/2006	Hajar
2	4000	30/10/2013	01/04/2014	Ahmad

Table8. Transaction table with Target Source data

3.4 Temporal ORDB QueryIncluding Varying Time:

This section describes the schema definition of the bank management system example **Figure 9**, 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 **Figure 10**:

TORDB Query

```
Create type t_Adress_Customer as object (
street  varchar(20),
City varchar(20),
State  varchar(20),
Codepostale varchar(20),)/
Create Type NT_AdressCustomer as object (
Adress_Ref  REF t_Adress_Customer
Vt_Start Date,
Vt_End  Date)/
Create type  Adress_cus_h  is table of NT_AdressCustomer;
Create type t_Loan as object (
Loan_NO  varchar(20),,
Amount  Number,
Type varchar(20),
Vt_Start Date,
Vt_End  Date)/
Create table Loan of t_LoanCONSTRAINT
Loan_NOPRIMARY KEY(Loan_NO);
Create type NT_Loan as object (
Loan  REF  t_Loan,
Vt_Start Date,
Vt_End  Date)/
Create type Loan_H is table of  NT_Loan;
Create type t_Customer as object (
ID_custNumber,
```

Name varchar(20),
Phone varchar(20),
Adress_History Adress_cus_h ,
Loan_History Loan_H,
Vt_Start Date,
Vt_End Date)/
Create table** Customer **of** t_Customer**CONSTRAINT
*ID_cust**PRIMARY KEY**(ID_cust), **NESTED TABLE***
*Adress_History **STORE AS** Adress_tab, **NESTED TABLE***
*Loan_History **STOREAS** Loan_tab;*
***Create type** NT_cust **as object** (*
*Customer **REF** t_Customer,*
Vt_Start Date,
Vt_End Date)/
***Create type** Customer_H **is table of** NT_cust;*
***Create type** NT_balance **as object** (*
Value Number,
Vt_Start Date,
Vt_End Date)/
***Create type** balance **is table of** NT_balance;*
***Create type** t_Account **as object** (*
Acc_NO Number,
Acc_Type varchar(20),
Balance_H balance ,
Customer_History Customer_H,
Vt_Start Date,
Vt_End Date)/
***Create table** Account of t_Account **CONSTRAINT** Acc_NO*
***PRIMARY KEY**(Acc_NO), **NESTED TABLE** Balance_H*
STORE AS** balance_tab, **NESTED TABLE
*Customer_History **STORE AS** Customer_tab*
***Create type** T_Checking_Account **UNDER** t_Account*
*(Overdraft_limit Number) **Final**;*
***Create type** t_Adress_Branch **as object** (*
streetvarchar(20),
City varchar(20),
State varchar(20),

```
Codepostale varchar(20))/
Create Type NT_AdressBank as object (
Adress REF t_Adress_Branch  ,
Vt_Start Date,
Vt_End  Date)/
Create type  Adress_h  is table of NT_AdressBank;
Create type NT_Account as object (
Account t_Account )/
Create type  Account_h  is table of NT_Account ;
Create type t_bank as object(
ID_Branch Varchar(20) ,
Phone Number,
Account_history  Account_h,
Customer_History Customer_H,
Adress_history Adress_h
Vt_Start Date,
Vt_End  Date)/
Create table Bank_Branch of t_bank CONSTRAINT
ID_Branch PRIMARY KEY(ID_Branch), NESTED TABLE
Account_history  STORE AS account_tab, NESTED
TABLE Customer_History STORE AS Customer_tab,
NESTED TABLE Adress_history STORE AS Adress_h;
Create type T_trasaction as object(
Transaction_ID Varchar(20) ,
Type Varchar(20),
Amount Number,
Vt_Start Date,
Vt_End  Date) Not Final/
Create type T_transfer UNDER T_trasaction(
target_source  varchar(30) ) Final;
Create table T_transfer of T_trasaction CONSTRAINT
Transaction_ID Primary Key ( Transaction_ID);
```

Figure10. Result of the migration into Temporal ORDB

IV. MODELLING AND MAPPING METHOD FROM UML/OCL INTO BITEMPORAL DATA

Several tools can be used to enhance the conceptual layer, andallow modeling the restrictions dependent on the temporal features in order to support the creation of code, reverse engineering, etc.

However, it is a well-known fact that the least used of all UML languages is OCL.Here, we describe how we can transform UML into UML/OCL as a new meta-model that offers a promising way for creating and analyzing the temporal system. This is done by adding some OCL constraint to our UML class diagram for providing some correctness properties to our conceptual model. In the next, required OCL constraints are defined to specify the restrictions and the business rules on the temporal attribute that cannot be expressed in UML model. In the last, we will show the translation of OCL expression into their equivalent in the TORDB. We will use Bitemporal data which combine valid time and transaction Time to capture the objects activity and the part of the reality to be modeled.

4.1 Employing UML/OCL for designing Temporal Database

As we see in the previous section, UML Model represents an explanation of the system as the whole, and show how the different entities are related to each other. UML permits the description of functional, static and dynamic models and the series of changes of the system over time.Now, it is necessary to develop a new model based on diagram class according to bitemporal dimension aspect and formed with restricted OCL constraints, in order to describe temporal database operation modeling, and to promote the visualization of the navigational path. It is able to define all the possible traces of an interaction between objects. This model is enriched by the use of additional constraints specified in the object constraints language. This model allows us to develop UML/OCL model adding bitemporal data which give the possibility to understand the structured of the temporal database and construct the schema translation that facilitates the converting process into a temporal object-relational database.

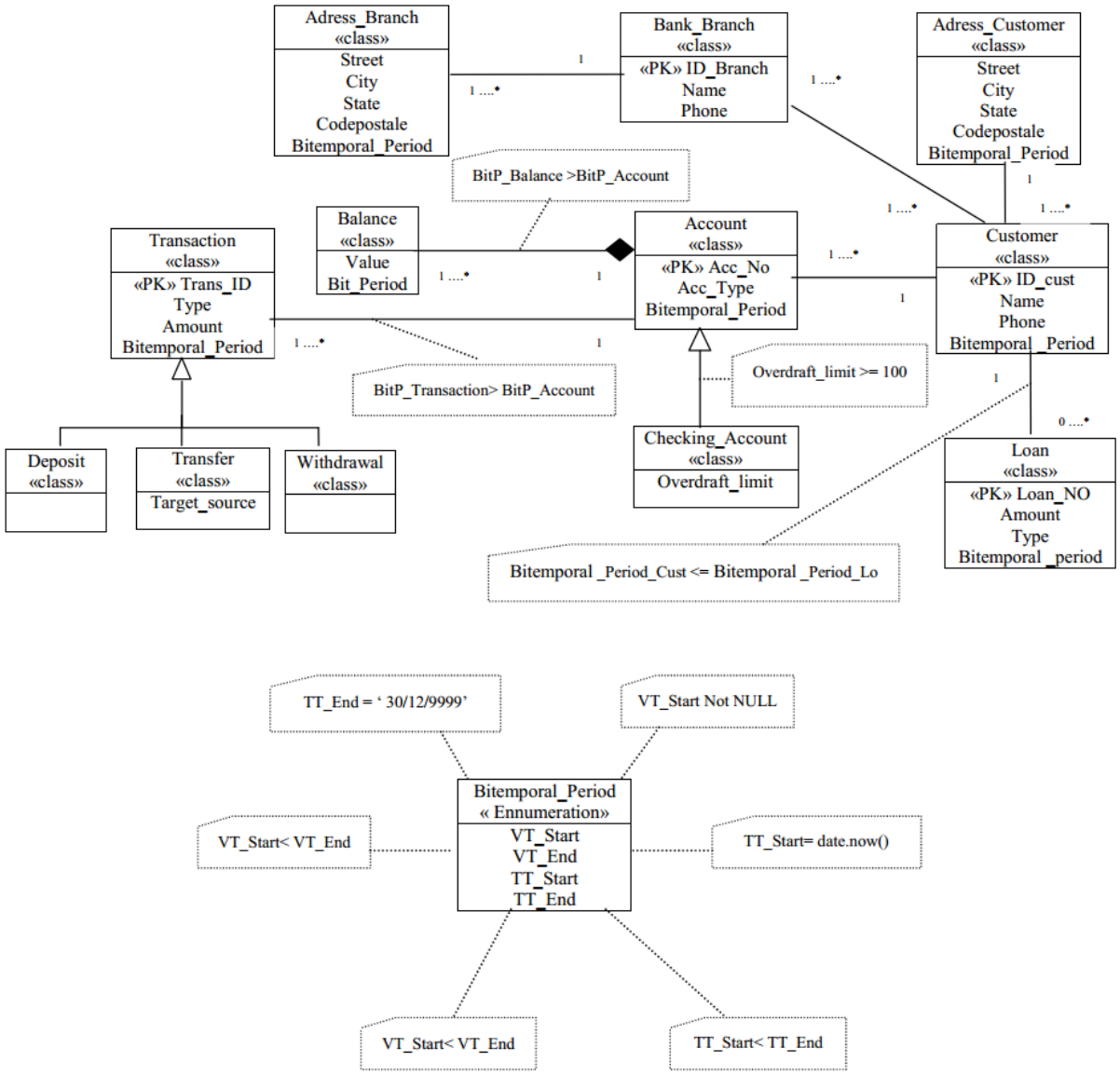


Figure11. UML/OCL conceptual Design for banking Management

4.2 The migration from Class_schema into TORDB Model

In this section, we outline the essential phases for translation method from class diagram into TORDB. In the fist, we define the class schema, corresponding in TORD model. The model extraction phase transforms class schema and their relationships to an equivalent set of structured type with bitemporal data.

4.2.1 Identification of Class_Schema

Figure 12 shows meta-Model of banking system; this model contains all concepts and information needed to achieve the goal of this work.

While class diagram has several techniques and elements for modeling temporal database system, we have selected the most commonly used in the database which can specify the relevant entities, their properties, and their relationships. The definitions of the elements of the UML class diagrams with the purpose of converting it into a temporal object relational model (TORDB_Model) are characterized by the following expression:

A class can be defined as:

C_Schema = {C|C=(Cn , A, S_class, REL, O , Bit_P,PK)}where

- **Cn**= Name of class
- **A**= denotes a set of class attributes
  A= {An, T} each attribute has a name An, and T is the type of attribute.
- **REL** = A Relationship where a class C is participating, where each Class C has a set of relationships with other class
  REL={REL | REL= (RelType,DirC,mp)} where
  RelType represents a type of relationship which supports four types: association, aggregation, composition and inheritance.
  DirC is the name of class C' interacts with Class C.
  Mp means a multiplicity to identify if the attributes can be simple or multivalued.
- **O:** defines a set of operations
- **S_class**= denotes name of super class in case of inheritance relationship
- **Bit_P**: Bitemporal element
  Each temporal class specifies bitemporal dimension by adding Bit_P which can have the following values:

Bit_P={(vt-start,vt-end,TT_start,TT_end)}.

- **Association:** Associations are links between two or more classes. It's particularly useful in order to specify navigability paths among objects.

A class with RelType= **"Associat with"** is defined as follow:

$$C\_A= \{ C|C=(Cn, AT, \text{“Associat with”}, C'n , mp, Bit\_P ,O, Pk)\}$$

Where:

Cn is the name assigned to the class; AT is a finite set of class's attributes, C'n is the name of class C' interacts with class C, O is a finite set of operations, mp means the multiplicity of the class C, Bit_P is the Bitemporal Period and PK is the primary key of Class C .

- **Aggregation:** An aggregation relationship is a binary association that specifies a whole-part type relationship [Fundamentals for the automation of] where the part is shareable and independent from the whole class. Therefore, the class with aggregation relationship can be defined by the following expression:

$$C\_AG= \{ C|C=(Wn, AT ,\text{“aggregation”}, Pn, mp, Bit\_P ,O, Pk)\}$$

Where:

Wn is the name of the whole class composed of 0..n objects of the part class P, AT is a finite set of attribute of the whole class, Pn is the name of part whose objects compose the class whole.

- **Composition:** It is a special kind of aggregation in which the part element is physically included in the whole. A composition relationship is a not-shared association that identifies a whole-part type relationship, but this relation is stronger than the aggregation, due to the part life depends on the whole existence. The composition can be defined as follow:

$$C\_C= \{ C|C=(Wn, AT,\text{“iscompposite”}, Pn, mp, Bit\_P ,O,PK)\}$$

- **Inheritance:** The inheritance is very important and easy type of relationship between more general classes of a hierarchy denoted super class and the lower class in the hierarchy called sub-classes. The inheritance relationship is formally expressed as:

$$C\_I= \{ C|C=(SCN, AT,\text{“ihnerit”}, SPC, Bit\_P ,O,PK)\}$$

Where: **SPC** means the super class name, and SCN is the name of subclass.

4.2.2 Definition of TORDB Model

Now, we will explain the different elements composed of a TORDB Model, which provides an efficient description of the temporal object-relational database. The TORDB model obtained is defined as a set of temporal typed table based on structured type ST and temporal structured type TST for storing data. Each TST is composed of a set of simple attributes and varying time attributes. The varying time attribute can be a bitemporal attribute, which actually stored in a nested table as a collection type.

The Definition of TORDB model is denoted as three-tuples:

$$TORDBModel=\{TT|TT=(TTs,STs,Tm)\}$$

Where:

TT s is set of temporal and non temporal typed table, STs is a set of temporal structured type or simple structured type , and Tm is a time-varying Period. The sets TTs, STs and Tm are defined as follows:

- **STs** ={Sn , S, AT},where Sn is the name of a structured type , S is the super type of ST, and AT is a set of structured type's attributes: $AT=\{A|A:=\{N,T,D,NL,BitT,M\}\}$,where

N: is the name of attribute, T: means data type which can be primitive, UDT or reference. NL: if the attribute accepts Null or not. D: default value. M: denotes if the AT is a single valued or collection valued. BitT: denotes if the attribute contains a bitemporal attribute is defined: $BitT=\{(AT1,AT2,\ldots,VTL,VTU,TTL,TTU)\}$.

- $TTs=\{typedtable|Ttable=\{TTn,STn,PK,Tp\}\}$

Where: TTn is the name of typed table, STn is the name of the structured type based upon which TT is defined, PK : primary key, TP: means if the TT is temporal or not.

- **Association:** For each relationship Rel where RelType= "associate with" is translated into:

$$TT\_Association = \{TT|TT=(TTn, STn, AT \cup NT (Ref (ST')) ,PK, Bit\_P)\}$$

In the relationship with an association class, we propose a method for maintaining the reference type of the structured type ST' referencing to the class which is related as a Nested Table collection with Bitemporal data dimension.

- **Aggregation:** each relationship Rel where RelType= "Aggregation" is mapped as a collection of UDT (User Defined Type) representing the C' class that participates in a relationship with class C.

$$TT\_Aggregation = \{ TT | TT = (TTn, STn, AT \cup NT(UDT(ST')), PK, Bit\_P) \}$$

- **Composition:** each relationship rel where RelType = "composition" is translated into an attribute typed as a nested table stores the attributes of the ST' corresponding to a class part C'.

$$TT\_Composition = \{ TT | TT = (TTn, STn, AT \cup NT(ST'), PK, Bit\_P) \}$$

- **Inheritance:** each relationship rel where RelType = "inheritance", the class C' inherits all the properties of its super class which is corresponding to ST'. For the creation of structured type that represents the inheritance relationship, we use the keyword UNDER during the creation of ST that corresponds to sub class C.

$$TT\_ihneritance = \{ TT | TT = (STn, ST.AT \cup Super\_T.AT, PK, Bit\_P) \}$$

4.3 Algorithm of conversion from Class_schema into TORDB Model

After classifying and extracting the basic information about the different classes, relationships and attributes, we propose an algorithm produces TORDB Model to automate the conversion process from C_Schema into TORDB Model.

- Class and attributes Extraction and transformation

Algorith to produce TORDB Model

Input : C_schema
Output: TORDB Model
Begin
ST,ST_C':STs:=Ø
TT:TTs:=Ø
TP:Tm:=Ø
M,TP,RelType:string:=' '
Foreach *class C ϵ C_schema* ***do***
If *C.BitP != Null then*
ST.AT.BitT=C.BitP
TP= 'Yes'
EndIf
ST=C.Cn+'_T'
TT.TTn=C.Cn
TT.PK = C.PK

```
Foreach Attribute A ϵ C. C_schema do
If  A.An!=C.PK then
ST.AT.T=mapType_ORDB(A.T)
EndIf
End For
```

Figure 12. Algrithm to extract the important component of the banking system

We begin the transformation process by taking the conceptual schema (CS) as input, the CS presented in our work represents an UML class diagram, which contains a set of classes and attributes connected by relationship and enriched by temporal data features.

The first step takes every class in the conceptual schema, extracting its metadata information including Bitemporal data, class name, attribute properties (Name, Type, Null or not, data Lengh) and the primary key which express the data dependencies and play a very important role during the schema translation process. After that, the algorithm generates their equivalent in TORDB model, and keeps the same name listed in C-schema adding '_T' to identify the table in temporal object relational database.

- Relationships Extraction and transformation

```
Algorith to produce TORDB Model
Foreach Relationship RlϵC.Rel do
C':C.C_schema
C'=getRel(Rel.DirC)
If  C.mp=(('1,1..N')||('1..N,1..N')) then
ST.AT.M= 'collection valued'
Else
ST.AT.M= 'single valued'
EndIf
If Rl.RelType= 'Associatewith' then
ST.AT= ST.AT∪{'NT('+defineRef(C')+ ')'+Bit-P+
AddConstraint(D,N)}
EndIf
If Rl.RelType= 'Aggregation' then
ST.AT=ST.AT∪{'NT(UDT_C')'+AddConstraint(D,N)+Bit_P}}
EndIf
If Rl.RelType= 'Is composite' then
ST.AT= ST.AT∪{'NT('+ST_C'.AT +')'+ AddConstraint(D,N)}
EndIf
If Rel.RelType= 'Ihnerit'  then
ST= C.SCN +'_T
```

```
TT.TTn=C.SCN
ST.Super_T= C.SPC
ST.AT= ST.AT∪{'UNDER('+getAT(ST_C'.AT.N)+ ')' +
AddConstraint(D,N)}
EndIf
EndFor
EndFor
Return(TT,ST,TP)
End Algorithm
```

Figure13. Algrithm of rules transformation

An efficient C-schema construction overcomes the complication and difficulties that can be occur during matching of keys in order to classify the relationships.

After finishing the first step of class and attribute extraction, we catch the part of the relationships; in our algorithm we cover all type of relationships (aggregation, composition, inheritance specialization and simple association). This step takes every relationship using the exported key, we concatenate the key with its references, for the purpose to determinate the classes which the current class interacts with them. Also, the cardinalities are defined to simplify the conversion process. In order to express as much semantics possible and to give a complete conversion process, the proposed model has taken into consideration the rules provided in our previous work, which formalize the necessary that can help us to transform each relationship into their equivalent in temporal object relational database.

V. MAPPING BETWEEN OCL SPECIFICATIONS AND TORDB:

In a high level of abstraction UML plays a key role in software and data modeling. Despite of the maturity and evolution of UML, it's not enough to present all characteristic of data specially the data behavior, that's why OCL (Object Constraint Language) is used. OCL is a natural comprehensive language and is a part of the UML standard and can be used with any Object Management Group (OMG) meta-model.

Using OCL to describe constraints and restrictions give us the possibility to enforce and enrich our UML class diagram and allow us to use one data model for many ORDBMS vendors. When using SQL directly with UML, here we merge the conceptual schema with the physical schema and we lose the power of making generic conceptual schema which represents different physical schema of different vendors (Oracle RDMS, Microsoft SQL Server).

This section presents some OCL expressions that must hold for the system being well modeled and its transformation into logical modeling expressed by temporal object Relational techniques. Furthermore, we will describe some restrictions with the matching OCL clause. The determinate restriction will be expressed by the different kind of constraint (triggers, integrity constraint, constructors….) in order to control access and storage of varying time data into temporal tables.

5.1 OCL specifications

OCL is a high level language for specifying contractual restrictions of object-oriented systems.It is a declarative textual language that expresses and describes the complex constraints on object-oriented models. It is associated with the UML standard for object oriented analysis and design operation in such common problem of complex systems. However, OCL offers the chance to explicit comparison of object attributes and automatic deal with business rules when building UML model for the object-oriented application. The most important advantages of OCL are navigation and attribute access which related to OCL expression with the values in a concrete model. More formally, OCL constraints can restrict the static aspect of UML standard through the different type of specifications.OCL expressions are connected to a context class. The proposed constraints are shown in the UML/OCL design with the proper syntax and are connected with the contextual element in order to express the restriction of varying time attribute and the requirement to maintain the system behavior. We extended our example UML model presented in **Figure 11** by simple and complex constraints which are defining as follows:

- **Enumeration For Bitemporal_Period**

Enumeration defined in UML Model has to be represented within OCL Constraints. Each enumeration is defined as a special data type and accordingly has a type name and the value name. We use the enumeration constraint to limit the content of an object to a set of acceptable attributes. The specification below defines an element called bitemporal_Period where the only allowed attributes are valid time start, valid time end, and transaction time start and transaction time end.

The following OCL expression defines a bitemporal data objects which are used to specify time-variant attributes:

contextBitemporal_Periodinv:

self. bitemporal_Period = bitemporal_Period::VT_startOr self.
bitemporal_Period = bitemporal_Period::VT_endor self.
bitemporal_Period = bitemporal_Period::TT_start or self.
bitemporal_Period = bitemporal_Period::TT_end

- **Valid_Time_Start NOT NULL**

The Figure 11 above, shows the banking system relation in the temporal Object-relational approach. Every Class has bitemporal dimension which contains Vt_Start and Vt_End attributes representing a valid period of a class. For convenience, we assume no null values in the valid time dimension. The NOT NULL constraint enforces a row to always store a value. The added restriction means that the insert or update operation of data not authorized without adding values in the vt_Start attribute of the bitemporal data object. To present this constraint with OCL, we create an invariant that uses the method ocllsUndefined() which return true if the value is NULL.

Consider the Vt_Start attribute of Adress_Customer Class as an example which must be Not Null.The respective OCL expression is given bellow:

Context Adress_Customerinv:

NOT self.bit_period.vt_start.ocllsUndefined ()

- **Default value for Transaction Time**

For any given temporal attribute, its transaction time may arbitrarily differ from its valid time where that transaction time is the time to indicate that the information is active in the system. The storage into a transaction time dimension automatically sets the value of system_start which a special value is associated with every insertion and sets the value of insertion of transaction time to the highest value (“31/12/9999”). In contrast to valid time, users are not allowed to assign or update the values of transaction period bounds. More precisely, the transaction time start store the default sysdate value during the insertion of new data.

Definition of OCL expression for Transaction Start(TT_Start) wich take the Sysdate as Default value .

Context Bit_period::TT_Start: dateTime

init :date.now()

Transaction End (TT_End) with default value “ 30/12/9999” identified by the following expression:

Context Bit_period::TT_End :dateTime
init : '30/12/9999'

- **OCL expression on Valid time interval**

A main requirement of such system that are interesting to make data historisy the data is that the user can be able to assign any time value, either in the past present or in the future for the start and end time. Therefore, another requirement of valid time a specification combines start time attribute and end time attribute to form a period. A period is used to refer to the interval of time, with upper and low values. When inserting the period, the user has to specify the valid time values indicating that the valid time start must be not null. Hence, the valid time end hold upper bound which is greater than valid time start value.

The following OCL expression defines the content restriction of valid time bounds (VT_Start<= VT_End).

Context Customer inv:
NOT self.bit_period. VT_Start.ocllsUndefined()
AND
(self.bit_period.VT_Start =<self.bit_period.Vt_End
OR
self.bit_period.Vt_End.ocllsUndefined())

- **OCL expression with ImpliesOperator**

Consider the class account in **Figure 11**. It shows the comparison relationship between the account and balance class, where the time-varying information of account and the balance that make records of historical Amount is captured and changed in each transaction. In these two classes, each time period represents a closed_open interval. The valid time start is included where the valid time is excluded. In this example, we deal with the problem of insertion of new balance record before any transaction operation. The pre amount (amount >= 2000) specifies the lower limit of the corresponding balance values. Each value is assigned in the valid time period in balance class must obey some restrictions that can be specified in OCL in the following way:

The OCL constraint below Identify the valid time restrictions of Balance and account tables. The valid time start of balance must be lower than the value of account, and the both don’t accept the NULL

value. On the other hand, the valid time end value of account and balance class obeys to the restriction described in the previous example.

```
Context Balance inv:
        self.bit_period.valid_time_start>=self.account.bit_period.valid_time_start
        AND
        ( self.bit_period.Vt_End<=self.account.Vt_End
                OR
self.bit_period.Vt_End.ocllsUndefined()
                Implies
        self.account.bit_period.Vt_End.ocllsUndefined())
```

5.2 Transformation of OCL Specification into TORDB

The examples below present a set of SQL queries and procedures to demonstrate the transformation process of the OCL specifications presented in the previous section into the object relational database with bitemporal data. For a better understanding of how our method works, we present some examples in the following way:

5.2.1 Creation of Bitemporal_Period Object

The constructor is used to initialize an object. It plays a very important role to define complex business rules to enforce the execution of complex constraint. In practice, there is a need to implement the constructor in our proposed solution for the reason to define the constraints, which are specified in the previous subsection to handle Bitemporal period. In this constructor, we identify the default values of the transaction Time period (TT-start,TT-End) , and specify the NOT NULL requirement to enforce a row to always store a value in the valid time dimension in order to avoid the NULL values . In addition, the constraint is identified to control the value of vt_End which must be greater than VT-Start during the insertion operation. Therefore, the constructor bitemporal_period is called every time the table is created, and during the insertion of the new tuple to verify the validity of the valid and transaction time periods, also to control the data access to temporal object-relational tables. This solution is the most universal and fulfills the requirement above:

Bitemporal Constructor

```
CREATE OR REPLACE TYPE BODY bitemporal_period AS
CONSTRUCTOR FUNCTION bitemporal_period (vt_start DATE, vt_end DATE,
        tt_end DATE)
RETURN SELF AS RESULT
 AS
 BEGIN
        IF vt_start IS NULL THEN
RAISE_APPLICATION_ERROR(-20343, 'The vt_start cannot be null');
                RETURN;
        END IF;

        IF vt_end IS NOT NULL THEN
                IF vt_start>vt_end THEN
RAISE_APPLICATION_ERROR(-20344, 'The vt_start cannot be greater than
vt_end');
                        RETURN;
        END IF;
        END IF;
        SELF.vt_start := vt_start;
        SELF.vt_end := vt_end;
        SELF.tt_start := TO_DATE(sysdate,'DD/MM/YYYY HH24:MI:SS');
  RETURN;
 END;
END;
/
```

Figure14. Bitemporal Type constructor

The query below defines object type bitemporal_Period which is used to specify the time-variant attribute. All temporal object-relational tables contain Bitemporal features are created with four additional columns (The VT_start and VT_end columns define the lower and upper bound of the valid time data, while TT_start and TT_end express the lower and upper bound of the transaction time data, respectively). The Bitemporal_Period uses the previous constructor to define the system behavior.

Figure 15. Bitemporal Object called the Bitemporal Constructor

Bitemporal Object

```
CREATE OR REPLACE TYPE bitemporal_period AS OBJECT
(vt_start DATE,
vt_end DATE,
tt_start DATE,
tt_end DATE,
CONSTRUCTOR FUNCTION bitemporal_period(vt_start DATE, vt_end
        DATE, tt_end DATE)
  RETURN SELF AS RESULT);
CREATE TYPE bit_period  IS TABLE OF bitemporal_period;
```

We use Oracle's concept of nested tables to create the temporal Object Relational tableswith bitemporal column. The example shows the creation of the Customer, Loan, and account tables and of all necessary auxiliary (object) types using the Bitemporal_period Object and including Bitemporal_period Constructor.

TORDB Query with Bitemporal Data
Create or Replace type *Customer_T* ***as Object***
(id_cust Number,
name_cusvarchar(20),
phone Number,
Loan_History Loan_H,
customer_hbit_period);
Create Table *customer* ***OF*** *Customer_T* ***NESTED TABLE*** *customer_h*
STORE AS *customer_tab,* ***NESTED TABLE*** *Loan_History* ***STORE AS*** *Loan_Tab*
Create or Replace type *Loan_T* ***As Object****(*
Loan_NO Number,
Amount Number,
Type varchar(20),Loan_Hbit_period)
Create Table *Loan_table****OF*** *Loan_T* ***NESTED TABLE*** *Loan_H*
STORE AS *loan_tab ;*

Figure16. Example oF Tables including Bitemporal Object and the constructor

5.2.2 Creation of trigger before inserts (transformation of Implies operator)

The OCL specifications are applied in balance class will be transformed into Triggers, which would be used to implement tracking the state of balance tables in object Relational database and make records of the system behavior that can enable to manipulate data upon insertion of the new values. We create the trigger to be automatically fired when some events occurred before the insertion in the balance table. Therefore, it is necessary to define all possible triggers as OCL expression in object-relational database to handle the varying time data, since triggers merge between SQL queries and procedural code implemented in Oracle. The following example shows the transformation of the OCL expression presented in the previous subsection:

Trigger before Insert
CREATE OR REPLACE TRIGGER *insertNewBalanceRecord*
BEFORE INSERT
ON *Balance_table*
FOR EACH *ROW*
WHEN *(new.amount>= 100)*
DECLARE

```
        account_balance NUMBER
BEGIN
Selectaccount_bl into account_balance from account_table
        where ( id = :new.id);
        IF ( :new.amount>account_Balance ) THEN
RAISE_APPLICATION_ERROR( 10266, 'Insufficient funds');
        END IF;
        Updateaccount_table
        set balance = account_bl - :new.amount;
END;
```

Figure17. Creation of trigger before insert into balance table

5.3 Exprimental study

The conventional INSERT statement provides sufficient support for setting the initial values of valid and transaction time period columns. The following Examples show an INSERT statement, which inserts a tuple in the account table. This statement is here just to show how temporal rows are inserted where each table can have as many tuples as needed to represent bitemporal attribute.

Also, we use the select statement to show how to retrieve all the information stored in account table in order to validate our solution. We can express the query as (See the Examples Below):

Description	TORDB statements
Insertion statement into account table	***INSERT INTO*** *account_table* ***VALUES****(1,'xxxx', (select REF(c) from customer c where c.id_cust =1), balance_T(Balance_Type(2345,bit_period(bitemporal_period(TO_DATE(' 03/05/201708:30:25','DD/MM/YYYYHH24:MI:SS'),TO_DATE('03/01/2018 12:22:10','DD/MM/YYYYHH24:MI:SS'),null), bitemporal_period(TO_DATE('04/06/201811:49:26','DD/MM/YYYYHH24: MI:SS'),TO_DATE('03/07/2019 10:01:36', 'DD/MM/YYYY HH24:MI:SS'),null))*
Select balance value between two dates '03/10/2017' and '03/05/2017' of a given customer:	***SELECT*** *c.id_cust,c.name_cus,c.phone, b.** ***FROM*** *customer c, table(c.customer_h) b* ***WHERE*** *TO_DATE('03/05/2017 08:30:25','DD/MM/YYYY HH24:MI:SS') >b.vt_start and TO_DATE('03/05/2017 08:30:25','DD/MM/YYYY HH24:MI:SS') <b.vt_end;*
Select created accounts between two date ' 03/05/2011' and '03/05/2011' of a specific customer	***SELECT*** *ac.acc_no, ac.acc_type, b.value_b, bh.** ***FROM*** *account_table ac, table(ac.balance) b, table(b.balance_h) bh* ***WHERE*** *TO_DATE('03/10/2017 08:30:25','DD/MM/YYYY HH24:MI:SS') >bh.vt_start and TO_DATE('03/05/2017 08:30:25','DD/MM/YYYY HH24:MI:SS') <bh.vt_end and ac.customer = (select REF(c) from customer c where c.id_cust =1);*

Select valid created account of a customer between two date '03/05/2011' and '03/05/2011'	***SELECT*** *ac.acc_no, ac.acc_type, ah.** ***FROM*** *account_table ac, table(ac.account_h) ah* ***WHERE*** *TO_DATE('03/05/2011 08:30:25','DD/MM/YYYY HH24:MI:SS') >ah.vt_start and TO_DATE('03/05/2011 08:30:25','DD/MM/YYYY HH24:MI:SS') <ah.vt_end and ac.customer = (select REF(c) from customer c where c.id_cust =1);*

Table 9. Example of Queries with bitemporal data

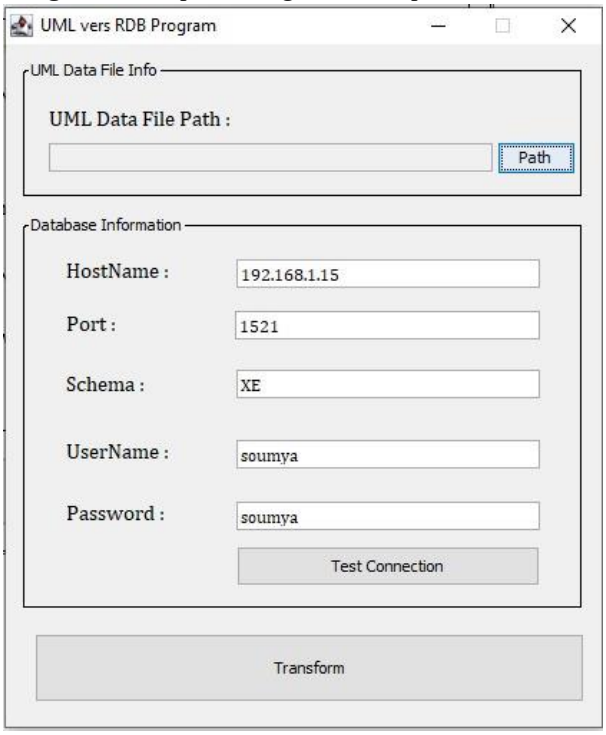
VI. IMPLEMENTATION:

We introduce the prototype that has been implemented to demonstrate our proposed approach. This application manages the migration between UML and temporal object relational database. The solution consists on 2 main steps:

- Upload the file which contains the Class schema with the definition of diffrents entities and properties, and the specification of the relationships.
- The "Transform" button allows converting the File into their equivalent by extracting the necessary component and replace the relationship names generated automatically for TORDB with meaningful names. This prototype is based on the schema translation algorithm and the proposed rules that are provided previously.

The following Figure illustrates the process of the mapping:

Figure18. Graphic Design of the implementation



VII. CONCLUSION

In this chapter, we described an approach for providing a conpetual design for varying time data based on UML. The solution is considered as a complete study that shows how we can maintain the collection semantics stated in the conceptual level into the implementation using temporal object relational database. We use the class diagram, their relationships, and properties as input enriched with semantic data to provide class_schema, which will be transformed to TORDB model which characterize the temporal and non-temporal tables. After that we formalized the necessary rules and steps for the migration into temporal object relational database starting with UML class diagram enriched by OCL, to express restrictions that must be applying on data during the storage of data. The study is based on a set of methods and rules depend on the UML class specifications. These new methods can gain benefit for specific cases such as tables with varying time aspects and temporal queries, etc. Our subsequent chapter, we will present the migration process from the temporal relational database into temporal Object relational database based on the obtained conceptual model and their rules to help on the schema conversion methods.

CHAPTER III:

MAPPING METHOD FROM TRDB INTO TORDB

I. INTRODUCTION

The relational database model has emerged a long way and it has become the dominant model of database that is based on traditional management systems. Today, in the majority of the management database systems, the time of event presents one of the important rows in its application. However, a need to shift time manipulation from application to RDB is implemented in SQL: 2011 standard, which adds period definition as metadata to tables on the relational database. A period is defined as table component, identifying a pair of columns in order to capture start and the period end time. The temporal relational database with SQL: 2011 features have been accepted as a solution for kept data changes over time with insertion of new records or update old records. Although the temporal relational database is designed to serve many applications which need only store the recent state of data, many problems have been emerged. They are insufficient for those who need to retrieve past as well as current and future data.The weakness of such Temporal RDBMSs in supporting complex data structures, user-defined data types and data persistence required by temporal object relational database. The weakness of such Temporal RDBMSs in supporting complex data structures, Object types and data persistence required by the temporal object-relational database. Furthermore, the reconstruction of temporal complexes objects split across relational tables is costly because of its causes many joins. Several Companies developing the largest database systems have updated their relational database systems by including object-relational technology. It allows developing databases for very complicated objects and ensuring efficient retrieval of data by the use of extended SQL language [71]. ORDBs with time-varying features have addressed these problems, which have a relational base and append object concept, to enhance ORDB performance and give the correct description of data, we associate time at rows. Temporal ORDB overcomes the disadvantage of data redundancy introduced when using temporal RDB to store the information.

The aim goal of this chapter is to provide a method for migrating RDB based on SQL: 2011 standard into Temporal ORDB integrating time-varying data. The migration is prominent in several different areas of temporal data management, including database design, information integration, and the use of temporal dimension. The solution comprises four basic steps. In the first, the method takes the temporal relational database as input and stores it in a temporal structured table that contains several parameters, attributes, class, relationships, cardinalities, integrity constraint and Bitemporal period, in order to enrich and realize the schema translation , also we will create TRDB queries using Period Clause to express valid time dimension. The schema translation so obtained is converted into a temporal ORDB model (TORDB), which handles complexes object and data semantic that can be expressed in its metadata in the second step. The third step deals with the transformation of the ORDB design based on TORDB Model, which holds the necessary for the correct description of Object associated with time, to TORDB tables. In addition, we will present algorithms which have been produced to automate the conversion process and the ability to manipulate temporal tables which are associated with one or more temporal period dimension.

II. AN OVERVIEW Of TEMPORAL RELATIONAL DATABASES:

2.1 Comparison between TRDB and TORDB

Several Scientifics, e-commerce and e-business systems have a temporal elements related with them integrating applications that involve time series analysis. Therefore, all companies needs to track the changes of data values at any point of the time. The time designed within the standard relational model cannot be done in straightforward manner. Due to the importance of storing and handling temporal data, there is a need for novel consistent extension of the standard relational model to manipulate such data In practice, the implementation of temporal database management systems (DBMS) using the varying time attribute approach is rarely available. On the other hand, implementing temporal features in Relational database systems requires development of additional varying time features to track changes dependent on the time for each record. For this reason, a recent specification of a temporal extension of SQL, termed SQL: 2011 has been emerged to support temporal relational database.The temporal tables are defined in SQL: 2011 standard and they are related to valid time. Even Oracle 12C, which now offers to RDB users new clause called PERIOD that support valid time dimension and it can be used daring the creation of the table to specify the opened-closed interval period. These kinds of tables are created with PERIOD statement with two predefined

attributes, one to store the initial time of period, and the other to store the end time of period. The relational table includes PERIOD clause, whereby the user specifies the name of the period and the values to be stored. On the other hand, to define the transaction Time in temporal Table, two additional columns where start time is set by system and the end time records '9999/12/31' as a default Value. The users are not allowed to supply values for these columns. The drawbacks of such TRDBMSs in supporting complex data structures, and temporal data persistence required by many application that are based on the storage time such as e-commerce and e-health systems , have led to the development of temporal object-based database systems.A temporal OR database TDB is an OR database DB that supports some aspect of time. The temporal relational model (TORD) has been applied in a number of areas and accepted as a one the efficient solution for storing and retrieving temporal data due to their maturity. The users can provide additional type of data by defining both the structure of the data and the ways of operating DB. The temporal ORDB offers the possibility to specify what the users want rather than how to achieve what they want. Using the proposed language for temporal data management allows manipulating temporal complex data of any data type and issuing temporal queries based on OR technology with ease and intuition. In terms of tables, a temporal Object Relational database is based on a collection of 1NF, non- 1NF, and typed tables. At least one of obtained tables has some temporal aspect which is called temporal table. If a temporal table is a typed table defined on a structured type and includes temporal Object, such a structured type is called a temporal structured type.the temporal object is declared as user defined type to present the varying time features in Object relational database. This Object contains four attributes, two attribute to store the valid time interval and the others to specify the transaction time dimension in order to express bitemporal data concept.

In the following subsection, we concentrate on comparing TRDB features with Temporal Object relational database concepts. The table below presents the main techniques of TORDB methodology and TRDB:

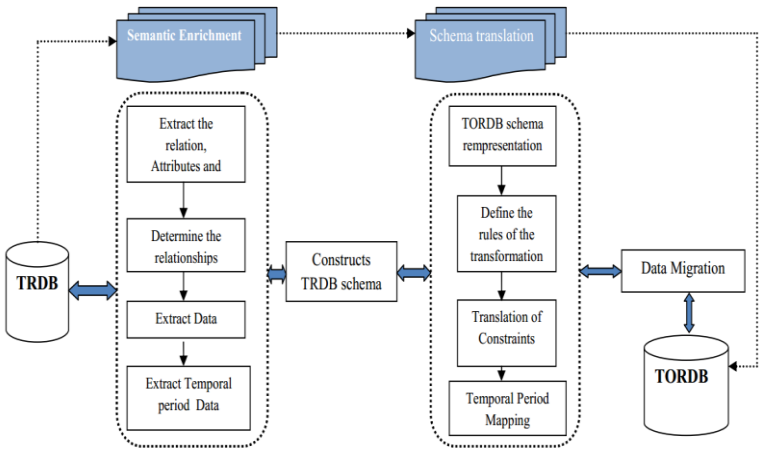
TRDB	TORDB
Varying period	User defined Time Object
Transaction Time	
Bitemporal data	
Primary Key	Primary Key
Foreign Key	REF
	UDT(User Defined Type)
	Collection(Nested table \varray)
	Under

Table10. Comparison between TRDB and TORDB

2.2 Strategy of the Migration from TRDB into TORDB:

The main question that needs to be answered is how to transform Temporal Relational Database technology into Temporal Object-Relational Database technology. Development of a method for the migration is the aim of this chapter. There are several steps is based on semantic enrichment technique required to achieve the goal. Semantic Enrichment is a process of analyzing databases to understand their structure, and to make hidden semantics explicit [49]. To enrich the semantic of temporal RDB schema, we have to extract its data semantic and converted into a much enhanced TRDB schema. Our process starts by extracting the basic information about an existing temporal RDB schema, integrating relations names, Periods time, and Attributes properties. We assume that data dependencies are represented by primary keys and foreign keys. The next step is to identify the TORDB Model constructs based on classification of data, the relationships, structured type and valid time period, which may be performed through data access. In the last step, the TRDB schema so obtained is converted into their corresponding in TORDB Model and the prototype algorithm is developed to validate the solution.

Figure19. Process of the Migration from TRDB into TORDB



III. MAPPING PROCESS OF TRDB INTO TORDB:

We present through this section the steps required to perform the migration of the temporal relational structure into temporal object relational structure, by identifying the schema conversion starting by more general dimension valid time. Also, we will describe the transformation model based on Bitemporal data to provide more details of temporal table .To do that, we give a formal definition of the schema conversion of TRDB and TORDB. The proposed schema is a source of valuable semantics producing an enriched and well structured data model, which can be translated flexibly into the target database. Besides considering the characteristics of the temporal Object relational model, the TRDB schema preserves all data semantics and varying time concepts that could be extracted from a RDB based on temporal features and the integrity constraints imposed on it. The obtained TRDB data Model plays an important role to simplify the migration of the data and the reallocation of attribute values to the appropriate values in temporal Object relational tables. In addition, we cover the technical level of temporal data, by implementing the data definition, Data manipulation and data retrieval statements. Each part of the translation process is described with examples.

3.1 Semantic Enrichment of Temporal Relational Database Using Valid time:

3.1.1 Definition of the New Valid Time Data Model

The NVTM is a representation of Relational Database contains valid time dimension, which is enriched with semantic data in order to provide a new kind of tables representing the classes extracted from temporal RDB, with the data necessary for the creation of temporal Object Relational Database.

This phase provides a data reference model that is designed to allow the exchange the temporal schema and the sharing of information to reuse.

The NVTM is defined in our approach as a set of element:

NVTM :={ C\|C :={ Cn, ANVTM, RelNVTM, Clas, PK-NVTM, FK-NVTM}}

Where:

- Each class has a name **Cn**
- **ANVTM** = means a set of Attributes of class C

ANVTM ={a|a:=(NA,TA,L,NL,D)} , where NA : attribute name , T : attribute type, L: data length, NL: if the attribute accepts the parameter null (N/ NoN), D: Default value.

- **RelNVTM**: Relations NVTM

  each class C has a set of relationships with other classes ,where rel is defined in C with another class C' ***RelNVTM={rel| rel:= RelType,DirC,Crd }*** , where each class C has a relationship RelType with other classes, DirC is the name of C' that interacts with C,Crd means cardinalities describing the relationship.

  RType offers five types of relationships:

  – "Ass" : Association
  – "Agg" : Aggregation
  – "comp" :Composition
  – "inher" and "inherBy": Inheritance

- **Clas**: classification

  Classification divides classes into two different kinds of categories:

  – Temporal class (TCls): class contains a varying time period.
  – Simple class (SCls): class without temporal data.

- **PK-NVTM**: each class C has a primary key PK.

  PK= {P|P:=PKn, NB}, where PKn is the primary key name, and NB is a number of Pk in case of a composite key.

- ***FK-NVTM:*** denotes foreign key of Class C, FK= {F|F:=FKn, NB} where: FKn is the Foreign key name, and NB is a number of Fk.

3.1.2 Generation of the NVTM from TRDB:

The NVTM presents the first step of the migration process from TRDB into TORDB. Consider the TRDB Example shown below. That example presents an RDB includes valid time features. In **Table 11**, we will generate the NVTM of the TRDB:

EmpNo	Name	Birthday	VT-Start	VT-End	dept	D-Start	D-End
1	Hajar	10/04/1986	15/02/2007	31/12/9999	1	15/07/1998	31/12/9999
2	Amine	24/08/1976	03/05/2004	31/11/2011	2	03/05/2004	31/12/9999
3	Ahmed	24/08/1980	03/05/2008	31/12/2013	4	21/12/2010	31/12/9999
4	Ilyas	30/01/1979	20/12/2005	31/12/2010	2	03/05/2004	31/12/9999
3	Ahmed	24/08/1980	10/12/2013	29/11/2016	3	20/12/2005	29/11/2016
5	Imane	31/05/1975	03/05/2006	31/12/2007	3	20/12/2005	29/11/2016

Empno	NumProj
1	2
5	1
3	17

NoK	Kname	sexe	Numemp
10	sarah	W	17
15	Mehdi	M	1

DeptNo	deptname	VT-Start	VT-End
1	Computer	15/07/1998	31/12/9999
2	Accounting	03/05/2004	31/12/9999
3	After-sales	20/12/2005	29/11/2016
4	Marketing	21/12/2010	31/12/9999

Empno	Grade	salary	Vt-Start	Vt-End
1	engineer	7000	15/02/2007	31/12/9999
1	Manager	8000	15/02/2015	31/12/9999
3	commercial	5000	03/05/2008	31/12/2013

NumProj	Nameproj	Details	VT_Start	VT_END
1	Payment Management	Creation of payment management application web	15/05/2006	01/01/2007
2	HR Management	Integration of a module in an erp source	30/10/2013	01/04/2014

Figure20. Sample Input representing TRDB tables

Cn	Clas	ANVTM					RelNVTM			PKNVTM		FKNVTM	
		NA	TA	L	NL	D	RelType	DirC	Crd	PKn	NB	FKn	NB
employee	TCLS	EmpNo Name Birthday Vt_start Vt_end dept D-Start D-End	Number Varchar Date Date Date Number Date date	25	NoN NoN NoN NoN NoN NoN NoN NoN		Ass Ass IhnerBy Agg	Department Works-on Salaried-emp Kids	1..N 1..N 1..1 1..N	EmpNo Vt_start Vt_end	1 2 3	dept D-Start D-End	1 2 3
department	TCLS	Deptno Deptname Vt-Start Vt-End	Number Varchar Date	25	NoN NoN NoN NoN		Ass	employee	1..1	Deptno Vt-Start Vt-End	1 2 3		

			date										
Kids	SCLS	NoK Kname Sexe Numemp	Number Varchar Varchar Number	25 10	NoN NoN N NoN		Agg	employee	1..1	NoK	1	Numemp	1
Project	TCLS	Numproj Nameproj Details Vt-start Vt-End	Number Varchar Varchar Date Date	30 255	NoN NoN N N N		Ass	Works-on	1..N	Numproj Vt-start Vt-End	1 2 3		
Works_on	SCLS	Empno Numproj	Number Number		NoN NoN		Ass Ass	Project Employee	1..1 1..1	Empno Numproj	1 2	Empno Numproj	1 2
Salaried_emp	TCLS	Empno Grade Salary Vt-Start Vt-end	Number Varchar Number Date Date	25	NoN NoN NoN NoN NoN		ihner	employee	1..1	Empno	1	Empno	**1**

Table 11. Result of the generation of NVTM

3.1.3 Translating NVTM into TORDB design schema

After obtaining the NVTM, we focus on the use of UML notation for creating the TORDB design scheme, which can facilitates the transition towards the object by a set of rules for transposition, and promotes the description of the complex type and possible navigation paths. The model of navigation introduces the logical links between object of the type ref or nested table in order to Decrease the redundancy of the temporal data.

The main goal behind developing a TORDB design is to simplify the comprehension of essentials information stored in temporal data. The TORDB modeling plays a pivot role between the conceptual schemes and implementation object.

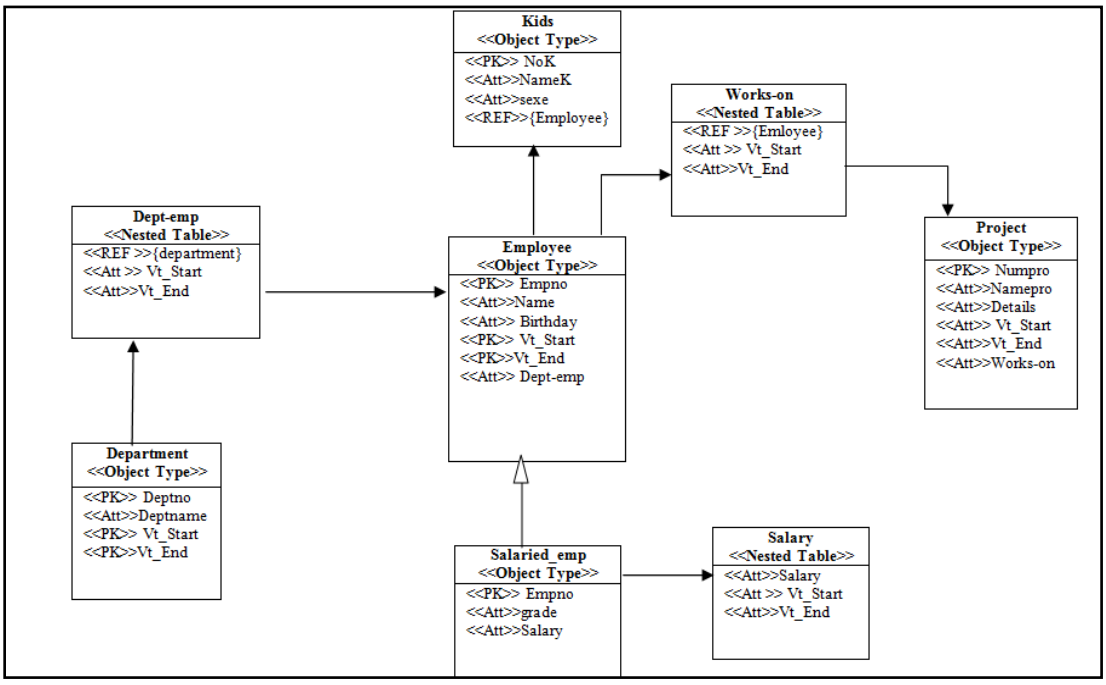


Figure21. Temporal ORDB design schema

3.1.4 Translation of the TRDB design schema to TORDB Query

This section describes the schema definition of the previous prototype, using the commercial database oracle 12C. Through the studies presented in the previous sections, it can be able to produce temporal ORDB queries for relationships. The temporal and no temporal queries formed as shown in **Figure 22:**

TORDB Queried with Valid Time
Create type *t_employee* ***as object(***
NoEmp NUMBER ,
Name VARCHAR(25),
Birthday DATE,
Vt_Start Date,
Vt_End Date,
Department dept_emp) Not Final /
Create table *employee* ***of*** *t_employee (****CONSTRAINT*** *emp-PK*
PRIMARY KEY*(NoEmp, Vt_Start ,Vt_End)),* ***NESTED TABLE***
Department ***STORE AS*** *dept_tab;*

Figure22. TORDB Querie

3.2 Temporal Relational Database Queries with Bitemporal Data:

This subsection describes the schema definition of the previous example. It can be able to implement TRDB queries with oracle 12c by using PERIOD FOR clause. The PERIOD clause is used in the creation statement of the table to specify valid time intervals. There is also another form of time-variant data, called Transaction Time. It represents the time period during which a data is stored in the database. The insertion into a transaction time period automatically sets the value of transaction time start which stores the default sysdate value during the insertion of new data and sets the value of insertion of TT_end to the highest value ("31/12/9999").

TRDB Queries with Bitemporal Data

```
Create table employe ( Empno NUMBER PRIMARY KEY, Birthday date,
KidsNo Number,
Start_vt date, End_vt date,
TT_start date DEFAULT sysdate,
TT_end date DEFAULT to_date('9999-01-01','YYYY-MM-DD'),
PERIOD FOR priod_vt ( Start_vt, End_vt));
Create table department(
DeptNo NUMBER primary key, deptname varchar(25),
Vt_start date, vt_end date,
TT_start date DEFAULT sysdate,
TT_end date DEFAULT to_date ('9999-01-01','YYYY-MM-DD'),
PERIOD FOR period_VT(vt_start , vt_end));
Create table Emp_Dept(
empNo NUMBER, DeptNo NUMBER ,
vt_start date,
vt_end date,
TT_start date DEFAULT sysdate,
TT_end date DEFAULT to_date ('9999-01-01','YYYY-MM-DD'), PERIOD
FOR period_VT(vt_start , vt_end));
```

Figure23. Example of TRDB creation Queries

3.3 Semantic enrichment of Temporal Relational Database:

3.3.1 Definition of the New Bitemporal Data Model

The NBTM is a representation of Relational Database contains Bitemporal dimension, which is enriched with semantic data this phase provides a data reference model designed to allow the exchange the temporal schema and the sharing of information to reuse based on valide time and transaction time in the same Time.

The NBTM is defined as a set of element using the same properties shown in the previous section, with a description of Bitemporal period:

$$NBTM = \{C|C:=\{Cn, ANBTM, RelNBTM, Clas, PK\text{-}NBTM, FK\text{-}NBTM, BT\_P\}\}$$

- Each temporal class specifies Bitemporal dimension by adding BT-P where:

$$BT\text{-}P = \{VT\text{-}start, VT\text{-}End, TT\text{-}start, TT\text{-}end\}$$

3.3.2 Generation of the NBTM from TRDB:

The NBTM presents the first step of the migration process from TRDB into TORDB. Consider the TRDB Example shown in **Figure 20** wich will be enhanced with Transaction Time Periods. The following example presents an RDB includes Bitemporal features. The NBTM is considered as an important phase for the conversion process which in the end generates the target schema.

Cn	Clas	ANBTM					RelNBTM			PKNBTM	FKNBTM	Bit_P
		NA	TA	L	NL	D	RelType	DirC	Crd	PKn	FKn	
employee	TCLS	EmpNo Name Birthday	Number Varchar Date	25	NoN NoN NoN		Ass Ihner By Agg ASS	Departm ent Salaried- emp Project	1..N 1..1 1..N	EmpNo		Vt_start Vt_end TT_start TT_end
department	TCLS	Deptno Deptname	Number Varchar	25	NoN NoN		Ass	employe e	1..N	Deptno		Vt_start Vt_end TT_start TT_end
Dept_Emp	TCLS	EmpNo Deptno	Number Number				Ass			EmpNo Deptno	EmpNo Deptno	Vt_start Vt_end TT_start TT_end
Project	TCLS	Numproj Nameproj Details	Number Varchar Varchar	30 255	NoN NoN N		Ass Comp	Employe e Budget	1..N 1.N			Vt_start Vt_end TT_start TT_end
Budget	TCLS	Numproj value	Number Number		NoN NoN		comp	project	1.1		Numpr oj	Vt_start Vt_end TT_start TT_end
Works_on	TCLS	Empno Numproj	Number Number		NoN NoN		Ass Ass	Project Employe e		Empno Numpr oj	Empno Numpr oj	Vt_start Vt_end TT_start TT_end
Salaried_emp	TCLS	Empno Grade Salary	Number Varchar Number	25	NoN NoN NoN		ihner	employe e	1..1	Empno	Empno	Start_vt End_vt TT_start TT_end

Table 12. Result of NBTM generation

3.4 Semantic Enrichment of Temporal Object Relational database:

3.4.1 Definition and Identification of TORDB Model:

As shown in the chapter II, The TORDB model is defined as a set of typed table based on structured type ST and temporal structured type TST which include varying time features for storing data. Definition of TORDB model is denoted as a set of element:

TORDBModel = {TTs, ST, TTm}

3.4.2 Translation of the TORDB design schema to a TORDB Queries:

In the previous chapter, we defined the temporal object to make records of temporal dimension periods by including a constructor which is used to initialize an object. We provided temporal ORDB queries which are formed by the temporal and no temporal queries. We use Oracle's concept of nested table and reference to store the varying-time attributes and to express the relationships between the different tables. The following Example shows the creation of the TORDB tables including bitemporal object of all necessary auxiliary (object) types.

Noemp	name	Birthday	Employee_H				Department					Project				
			VT_start	Vt_end	TT_start	TT_end	Dept	EmpDept_H				Proj	WorksOn_H			
								Vt_start	Vt_End	TT_start	TT_end		Vt_start	Vt_End	TT_start	TT_end
1	Hajar	10/04/1986	15/02/2007	31/12/9999	15/07/2009	31/12/9999	1	15/07/1998	31/12/9999	15/07/2009	31/12/9999	2	15/07/2013	15/11/2015	17/07/2013	31/12/9999
												1	22/04/2012	09/12/2014	30/02/2013	31/12/9999
2	Amine	24/08/1976	03/05/2004	31/11/2011	03/05/2004	31/12/9999	2	31/12/2014	31/05/2013	07/12/2009	31/12/9999	1	18/08/2012	18/08/2012	23/04/2012	31/12/9999
3	Ahmed	03/05/2008	03/05/2008	31/12/2013	03/05/2008	31/12/9999	4	22/12/2010	31/12/2014	30/12/2010	31/12/9999	2	30/10/2013	01/04/2014	02/11/2013	31/12/9999

Table13. Example of Temporal Object Relational table with bitemporal Data

TORDB Query to create Employee Table using the construction and including Bitemporal period:

Creation Statement

```
Create Type Employee_T as Object
(NoEmp Number,
Name varchar(25),
Birthday Date,
Emplyee_H bit_period,
Department Emp_Dept ,
Project Works_on
) Not Final ;

Create table Employee of Employee_T (Primary Key (NoEmp)) NESTED
TABLE Emplyee_H STORE AS employee_tab, NESTED TABLE
Department STORE AS Department_Tab(NESTED TABLE EmpDept_H
Store as EmpDeptH_tab ), NESTED TABLE Project STORE AS
project_Tab( Nested table WorksON_H store as WorksONH_tab ;
```

Figure24. Example of Creation statement for employee table

3.4.3 Algorithm for Translating NBTM to TORDB Model

After classifying the different classes, relationships and attributes, we propose an algorithm produces TORDB Model to automate the translation process of NBTM into TORDB Model as shown in **Figure 25**:

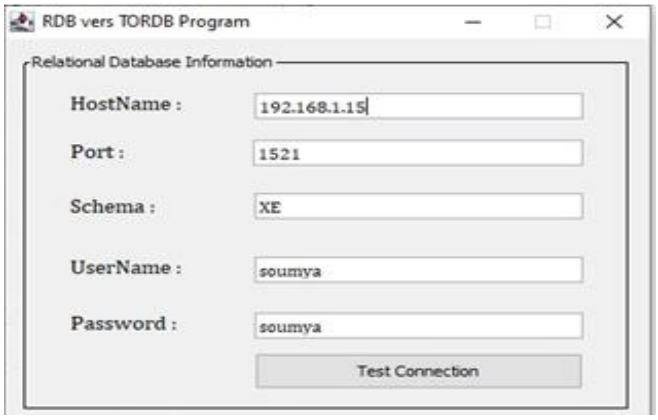
```
Algorithm Produces TORDBModel
  Input: NBTM
  Output: TORDBModel
  Begin
  ST: STs :=Ø
  TT: TTs :=Ø
  TP:Tm:= Ø
M,,Clas,TP,Crd:string:=''
Foreach class C ϵ NBTM do
     If C.clas='TCLS' then
       ST.AT.BT_P= BT_Period
       TP= ' Temporal class'
EndIf
       ST=C.Cn+'_Type'
       TT.TTn=C.Cn
       TT.PK=A.PK_NBTM
 Foreach Attribute A ϵ C.A.NBTM do
  If A.NA != A.PK_NBTM.PKn then
       ST.AT.N=A.NA
       ST.AT.NL=A.NL
   ST.AT.T=mapingType('ORDBType',A.T)
  Endf
 End For
Foreach Relationship Rel ϵ C.RelNBTM do
       C':C.NBTM
  If C'. FK-NBTM != NULL then
     C'=getRel(Rel.DirC)
    If Rel.Crd=(('1,1..N')||('1..N,1..N')) then
  ST.AT.M= 'collection valued'
    Else
  ST.AT.M= 'single valued'
    EndIf
    If Rel.RelType= 'Ass' then
  ST.AT=ST.AT∪{'NT('+defineRef(C')+')'+BT-P}
    EndIf
    If Rel.RelType= 'Agg' then
 ST.AT=ST.AT∪{'UDT('+getAT(C'.NA)+')'}+BT-P+AddConstraint()}
  EndIf
     If Rel.RelType= 'Comp' then
     ST.AT= ST.AT ∪{'NT('+C'.A+')'+BT-P+ AddConstraint ()}
  EndIf
  If Rel.RelType= 'Ihner' then
  ST.AT=ST.AT∪{'UNDER('+getAT(C'.A.NA)+')'+BT-P+ AddConstraint() }
  EnIf EndFor EndFor
  Return(TT,ST,TP)
End Algorithm
```

Figure25. Algorithm to produce TORDB Model

IV. IMPLEMENTATION

The following prototype is implemented to automate the transformation of temporal relational database into their equivalent in temporal Object Relational database, in order to simplify the migration with an efficient manner and gain of the main advantaged giving by Object relational model. The Java language is used for basic coding in our approach. We have chosen JAVA for its ability to connect to TRDBs and the reusability of code wich makes it preferable for use through JDBC. As described previously, our solution is based on 3 important phases:

- Connexion to the source database and the extraction of all possible information about tables, their attributes and keys, and temporal period from a given TRDB.
- the construction of TRDB schema from the obtained component by defined functions and methods
- the generation of the TORDB Model, which includes the classification of constructs and the identification of relationships and cardinalities . This function is responsible of TORDB Model generation by using the defined rules and the algorithm of the translation described in the previous subsection.
- The connexion will be established to database Target in order to create the TORDB tables.



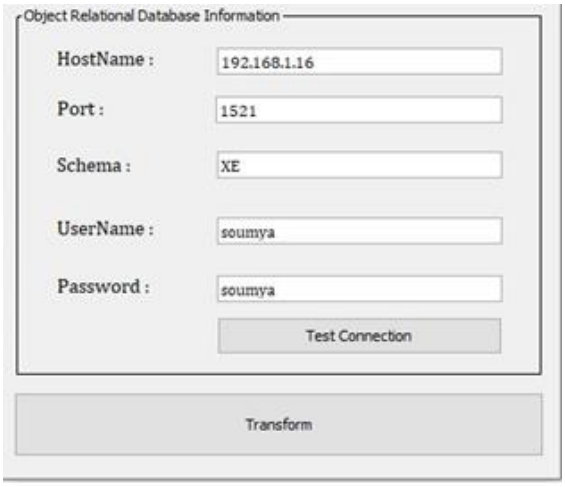


Figure26. Graphic Design of the transformation from TRDB Into TORDB

V. TEMPORAL DATA MNIPULATION

Here, we outline the more difficult parts of temporal database operations. The data manipulations, especially insert, delete and update operations to handle varying time data in TRDB and TORDB. With the consideration of temporal events in the temporal database, the three manipulation operations need deep enhancements. Temporal insertion is straightforward. Indeed, the INSERT statement should allow the introduction of valid time and transaction time, and the UPDATE statement must carry out of the non-destructive updates of any data. Furthermore, the DELETE statement must allow the non-destructive delete of rows in order to maintain the history of data, using bitemporal period. The UPDATE query can be used to modify the rows of temporal tables including Bitemporal period in order to define changes that are effective within a specified period. This is provided by both UPDATE and DELETE statements that let users specify the period of interest. We further detail the enhancements that we propose in the following three sub-sections.

5.1 Insert Statement:

The conventional Insert statement provides sufficient support for the insertion of the initial values of Bitemporal period columns. The “Insert” allows the user to introduce valid time period values. The transaction time columns are not specified in the insert query, default values referred at the creation of temporal database. For example, the following insert statements store one row into a project table in TRDB and TORDB:

Description	TRDB Queries	TORDB Queries
Insert into Project table	*Insert into project (Numproj ,nameproj,Details ,vt_start,vt_end) Values (1,'carrefour','ecommerce', date '2016-09-01',date '2022-12-31');*	*INSERT INTO project VALUES(1,'Carrefour project', 'e-business project' , Budget_T (Budget_NT(2345,bit_period(bitemporal_period(TO_DATE('03/05/2017 08:30:25','DD/MM/YYYY HH24:MI:SS'),TO_DATE('03/01/2018 12:22:10','DD/MM/YYYY HH24:MI:SS'),null), bitemporal_period(TO_DATE('04/06/2018 11:49:26','DD/MM/YYYYHH24:MI:SS'),TO_DATE('03/07/2019 10:01:36', 'DD/MM/YYYY HH24:MI:SS'),null))*
Select Budget values of all projects in '03/10/2017'	*SELECTproject.nameproj,budget.value,project.vt_start , project.vt_end FROM project AS OF PERIOD FOR period_VT TO_DATE ('01JAN-2014','DD-MON-YYYY') JOIN budget on budget.numproj= project.numproj ;*	*SELECT p.nameproj,b.value, ph.* FROM project p, table (p. Project_H)ph, table (p. Budget)b WHERE TO_DATE b.vt_start = ('03/05/2017 08:30:25','DD/MM/YYYY HH24:MI:SS') ;*
Select employee works on a given project between '1and 2-Jan-2013' and '06-JAN-2020'	*Select * from proj_emp VERSIONS PERIOD FOR period_VT BETWEEN TO_DATE ('12-Jan-2013','DD-MON-YYYY') AND TO_DATE ('06-JAN-2020','DD-MON-YYYY') JOIN project ON project.numproj=proj_emp.numproj JOIN employe ON proj_emp.empno=proj_emp.empno WHERE proj_emp.empno=2;*	*SELECT ep.numrpoj, e.empno, wh.* FROM employee e , table(e.projectt)ep, table(ep. WorksON_H) wh WHERE TO_DATE('12-Jan-2013','DD/MM/YYYY')>wh.vt_start and TO_DATE('06-JAN 2020','DD/MM/YYYY')<wh.vt_end and e. NoEmp = 2;*

Table14. Insertion statement with temporal data

5.2 Delete Statement:

Deletion Operation is a challenging task. Delete statement on Bitemporal Table does not actually delete the rows from temporal relational or temporal object-relational tables. For This reason, we provide a new tables Log that has the same schema and properties of the previous tables in TRDB and TORDB, in Order to preserve the deleted data.

Creation of table LOG

```
Create table  employe_log ( Empno NUMBER PRIMARY KEY,  Birthday
date,KidsNo Number, Start_vt  date, End_vt date, TT_start date DEFAULT
sysdate,TT_end date DEFAULT to_date('9999-01-01','YYYY-MM-DD'),
PERIOD  FOR priod_vt ( Start_vt, End_vt));
```

Figure27. Creation of table LOG for employee table

Therefore, we perform a trigger that fires for the row selected for deletion. The created trigger inserts the selected row into the table log and changes the transaction time-end value to date of Delete query is executed.

For example, suppose the current system row in the employee table with Bitemporal Period is shown below:

The following delete statement simplifies modifications of the transaction Time end columns of the current system row (31/12/9999) for the Employee to the transaction time in which the delete statement was executed. That means the transaction Time End of the current row will be inserted into Employee log, and the transaction Time End will take the system date value when the row is stored in table Log. A trigger is used to control the insertion into the table log after every deletion statement.

Figure28. Trigger to control historical data after delete statement

Delete Trigger

```
CREATE OR REPLACE TRIGGER delete_time
After  DELETE
  ON employe
    REFERENCING OLD AS old
 FOR EACH ROW
BEGIN
```

```
 INSERT INTO employe_log (Empno,Birthday ,Start_vt ,End_vt ,TT_start
,TT_end) VALUES(:OLD.Empno,:OLD.Birthday ,:OLD.Start_vt ,
:OLD.End_vt ,:OLD.TT_start ,sysdate);
END;
```

The above query outputs the records of employees who's still active in the current system and the deleted rows that are stored in table Logs :

```
SELECT * from employe

UNION

SELECT * from employe_log;
```

5.3 Update Statement:

Although traditional databases store data in its current versions only, the varying time databases make records of the informative on which were visible in the system at every point in time. The update operation in the temporal database is implemented as a modification of the current values, and the insertion of the old version in the table log with the modification of the transaction time start which stores the current date when the update statement was executed. Therefore, in the temporal database, the user is unable to change the content of historical rows, more precisely, the Bitemporal period interval associated. As deletion operation, a trigger is performed to restrict the insertion of the selected rows to be modified into a table log before executing the update statement.

Update Trigger

```
CREATE OR REPLACE TRIGGER update_data
BEFORE  UPDATE
ON employe
FOR EACH ROW
BEGIN
 INSERT INTO employe_history (Empno,Birthday ,Start_vt ,End_vt
,TT_start ,TT_end) VALUES(:OLD.Empno,:OLD.Birthday ,:OLD.Start_vt
,
:OLD.End_vt ,sysdate ,:OLD.tt_end);
END;
update employe
set birthday = date '1989-05-02'
where empno=1;
```

Figure29. Trigger to control historical data after Update statemen

VI. CONCLUSION

This chapter introduces the basics phases to convert an RDB based on SQL: 2011 standard into ORDB, which contains valid time and Bitemporal data features, with a simple and practical method to capture the different relationships between classes, association, aggregation, composition, as well as inheritance. We started by extracting data model from RDB including temporal data and implemented by SQL: 2011. We used it as an input enriched with semantic data, which is translatable into any of the target database schemas. After that, we have defined TORDB model including the schema and data by exploiting the range of powerful concepts provided by time varying data management. Therefore, we have provided a TORDB design to capture the characteristics of temporal and non-temporal SQL query. Finally, we proposed an algorithm for mapping method from TRDB with varying time period features into ORDB including temporal data.

CHPTER IV:

CONVERSION AND STORAGE OF DATA FROM TXML DOCUMENT INTO TORDB

I. INTRODUCTION

With the emergence of novel systems and the increasing complexity of Web applications and user requirements, there are different needs to shift from the data level to the human semantic interaction. Hence, semantic representation level becomes very interest in order to maximize the semantic data structure that requires these representations become increasingly explicit. For this reason, the users must learn how can deal with the ambiguity of language by understanding the context of information in which terms are used. Therefore, The World Wide Web was shifted from the semi-structured Internet to a more structured Web called the Semantic Web. The information stored by WWW is intended for human use. The contents on the web must be readable and comprehensible by the users. Information linked up in such a way to be process-able by machines in a mesh network defines Semantic Web where it develops languages for articulating information in a machine process-able form [72]. The new generation of Web provided many intelligent and mechanism that specify explicit semantics for data and enable knowledge sharing and publishing among knowledge-based systems. For this reason, ontologies are used to define and relate aspects and concepts that describe Web resources with a formal method. On the other hand, the dominant standard for information exchange on the Web is XML. It is used as a base syntax for other languages produced for the upper layers of the Semantic Web. XML and its related standards, such as DTD and XML schema are used to form a common means to structure and semi-structured data on the web. XML provides the syntactic layer for the Semantic Web. XML offers rich documents because of its ability to use meta-data and focuses on the meaning of documents rather than the presentation. However, the modeling for XML including temporal data and its transformation into Object relational database schema has not been widely investigated.

II. TEMPORAL XML DOCUMENT MIGRATION

XML is emerging as a standard format and Meta language for interchange data and structured document on the web. The data in XML file can be organized into hierarchies so that the relationships between data elements are visually obvious [73]. Various kinds of applications that use the XML format have been developed. This is why, XML needs to have databases system to store all data which will be reused and published later in the different environment. Nevertheless, Time is present in almost any real-world application especially on the web where XML data changes over time with the creation, modification, and deletion of XML documents. Temporal information is the nature and basic description for the development and changes of real word objects, and almost everything has explicit or implicit temporal features [74].Several databases applications need to keep records of changes of data in past present or in the future. Today, there are many database applications support a data type features dependent on the time. The most common examples of such applications involve flight reservation, the banking system, e-health, e-business, which require a full history and retain trace of data for these reasons: Avoiding loss of data after schema changes, maintenance of legacy data formatted according to past schema, reuse of legacy application and auditing purpose[75].Hence, XML provides excellent support for temporally grouped data models which have been accepted as a better solution to represent and to share temporal information. For this reason, it is certain to use ways needed to describe temporal XML schema formats in the temporal database using Object Relational concept based on bitemporal data dimension. Hence the acknowledge required from the combination two research area, management varying time and XML standard, led to the emergence of TXML document.XML provides excellent support for the temporal database in order to represent and to share temporal data.

III. SEMANTIC ENRICHMENT OF TXML

3.1 Definition of TXSDM

TXSDM is a description of XML schema with historical data which can define the relevant entities, their attributes or elements, and their relationships. The Model is enriched with semantic information that is extracted by in-depth analysis of TXSD document and takes into consideration features and object that are provided by object based model. Therefore, TXSDM is a data reference model that is designed as a method to extend and exchange the schema.

The TXSDM is defined in our approach as a set of complex type:

TXSDM = {CT|CT:= [N_{CT}, CLS, SE\A, REL, BitE, Key, KeyRef]} where:

- **N_{CT}** : name of Complex type
- **CLS**: each Complex type CT is classified into two different kind of categories as follow:
  1. Temporal **CT (TCT):** Complex type CT contains a historical data
  2. Simple **CT (SCT):** Complex type CT without temporal data.
- **SE\A**: denotes a set of Element or attribute of complex type CT.
  SE\A= {E\A:= $N_{E\A}$, $T_{E\A}$,MinO\MaxO, Use} each attribute or element has a name $N_{E\A}$, $T_{E\A}$ is the type of the element or attribute , MinO is the minimum of occurrence, MaxO is the maximum of occurrence ,and Use means if the element or attribute is required or not.

- **REL**: denotes relations TCSDM
  Each complex type CT has a set of relationships with other complex types.

  REL= {REL: = RelType, DirC}.

  RelType means a type of relationship where RelType supports four types:

  1. “Ass” for Association
  2.“Agg” for Aggregation
  3. “Comp” for Composition
  4. “inher” and “inherBy” for Inheritance

DirC is the name of complex type CT’ interact with CT.

- **BitE**: Bitemporal element

BitE= {N_{BitE},T_{BitE} ,MaxO\MinO} where

N_{BitE}: Bitemporal Element name

T_{BitE}: Predefined Type of BitE which accepts as a value Bit-Period.

Bit-Period is a bitemporal complex type specifies the lower and upper bound of valid Time (VT) and transaction time (TT) : ***Bit-Period = {VT-LB, VT-UB, TT-LB, TT-UB}.***

MaxO\MinO= Max and Min of occurrence

- **Key**: primary Key of CT

  Key= {N_K , Selector, Field}

  Each key has a name, element Selector as scopes within which the key is defined and a collection of related sub elements field are specified by selector to be unique.

- **KeyRef**: data dependencies are represented by KeyRef which is a reference to key of another complex type.

  KeyRef= {N_{FK} , Selector, Field, Refer}

  Each KeyRef has a name N_{FK} , Selector , Field and Reference constraint Refer.

3.2 Generation of TXSDM from TXML schema File:

The TXSDM presents the first step of the migration process from XML schema into TORDB, which in the end generates the target schema. Let Consider the XML schema file shown in **Figure30**, modelling a part of the purchase Orders system in a business company.

TXML Document Extraction

```
<?xml version="1.1" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xs:complexType name="Bit_Period">
<xs:sequence>
<xs:element name="VT_LB" type="xs:date"/>
<xs:element name="VT_UB" type="xs:date"/>
<xs:element name="VT_LB" type="xs:date"/>
<xs:element name="VT_UB" type="xs:date"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="Dis_history">
<xs:sequence>
<xs:element name="percent" type="xs:integer"/>
<xs:element name="Discount_H" type="Bit_Period"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="customer">
<xs:sequence>
<xs:element name="customerName" type="xs:string"/>
<xs:element name="Customer_H"type="Bit_Period" />
<xs:element name="phone" type="xs:integer"/><xs:element name="Address_cust" type="address"
maxOccurs="unbounded"/>
<xs:element name="order" maxOccurs="unbounded">
<xs:complexType >
<xs:sequence>
<xs:element name="orderID" type="xs:integer" />
<xs:element name="Order_H" type="Bit_Period"  />
</xs:sequence>
</xs:complexType >
```

```
</xs:element>
</xs:sequence>
<xs:attribute name="customerId" type="xs:integer" use="required"/>
</xs:complexType>
<xs:complexType name="address">
<xs:sequence>
<xs:element name="street" type="xs:string"/>
<xs:element name="city" type="xs:string"/>
<xs:element name="zipCode" type="xs:integer"/>
<xs:element name="Adress_H" type="Bit_Period" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
<xs:element name="Person">
<xs:complexType>
<xs:complexContent>
<xs:extension base="customer">
<xs:sequence>
<xs:element name="Discount_H" type="Dis_history"/>
<xs:element name="Bitemporal" type="Bit_Period" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:element>
<xs:element name="customer" type = "customer" >
<xs:key name="customerId">
<xs:selector xpath="customer"/>
<xs:field xpath="@customerId"/>
</xs:key>
<xs:keyref name="Purchase_Order_Ref" refer="orderID">
<xs:selector xpath="customer"/>
<xs:field xpath="@order"/>
</xs:keyref>
</xs:element>
<xs:element name="PurchaseOrder">
<xs:complexType>
<xs:sequence>
<xs:element name="shipping_date" type="xs:date"/>
<xs:element name="Adress_shipping" type="address"/>
<xs:element name="Orderline" maxOccurs="unbounded"/>
<xs:element name="Orderlineitem" >
<xs:complexType>
<xs:sequence>
<xs:element name="quantity" type="xs:integer"/>
<xs:element name="productId" type="xs:integer" maxOccurs="unbounded"/>
<xs:element name="bitemporal" type="Bit_Period"></xs:element>
</xs:sequence>
<xs:attribute name="line" type="xs:ID" use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence><xs:attribute name="orderID" type="xs:integer" use="required"/>
</xs:complexType>
<xs:key name="orderID">
<xs:selector xpath="PurchaseOrder"/>
<xs:field xpath="@orderId"/>
</xs:key>
</xs:element>
</xs:schema>
```

Figure30. Example of TXML schema document

Now, we generate the TXSDM of the TXML scheme described in the example above (see **Table15**)

Table 15. Result of TXSDM Generation

N_{CT}	CLS	SE\A				REL		BitE		Key	KeyRef
		$N_{E\A}$	$T_{E\A}$	MinO\MaxO	Use	RelType	DirC	N_{BitE}	T_{BitE}		
Person	**TCT**	Discount	Dis-history	unbounded	Required	Inher	Customer				
Discount	**TCT**	Percent	xs:integer	unbounded		Aggr	Person	Discount-H	Bit-Period		
Address-Cust	**TCT**	Street City zipCode	xs:string xs:string xs:integer			Agg	Person	Adress-H	Bit-Period		
Customer	**TCT**	CustomerId CustomerName Phone Address-Cust Order	xs:IDxs:stringxs:integerAdress	unbounded unbounded		Ihnerby Ass	Person PurchaseOrder	Customer-H Order_H	Bit-Period Bit-Period	CustomerId	OrderId
Purchase-Order	**TCT**	OrderId Address-shipping OrderLine	xs:ID address	unbounded	Required	Ass comp	Customer OrderLineItem	Shipping_date	Bit-Period	OrderId	
OrederLineItem	**TCT**	Line Quantity ProductId	xs:ID xs:integer xs:integer	unbounded		comp	PurchaseOrder	OrderLineIT-H	Bit-Period	Line	

3.3 Algorithm for Translating TXSDM into TORDB Model:

In This subsection, we present an algorithm to produce TORDB Model in order to automate the translation process of TXSDM into TORDB Model shown in **Figure 31**. After classifying and extracting the basic information about the different complex types, the algorithm goes through the main loop to TXSDM constructs as input and generates their equivalents in TORDB Model as output.

In the second chapter, we defined TORDB as a reference model which provide complete description of temporal Object Relational Database. The TORDBM obtained is presented as a set of temporal table based on structured type and temporal structured type thatcontain varying time attributes.Definition of TORDB model: is denotes as three-tuples:

TORDB Model = {TTs, STs , Tm}

Algorithm To Produce TORDBModel

Input: TXSDM

Output: TORDBModel

Begin

TST: STs :=Ø

TT: TTs :=Ø

TBit:Tm:= Ø

Min,Max,RL,CoL,SV,NL,:string:= ' '

Foreach** ComplexType CT ϵ TXSDM **do

If CT== Bit_Period Then

TBit= getBitPeriod(Bit_ Period)

EndIf

ST:StructType

Ttable:TypedTable

ST.Sn=CT. N_{CT} + '_t'

Ttable.TTn= CT. N_{CT}

Ttable.PK=getKey(Key.CT)

Foreach** Element E\A ϵ CT.E\A **do

If E\A .MinO= '0' Then

ST.AT.NL= 'Null'

End If

If** E\A .MaxO= 'unbounded' **Then

ST.AT.M= 'CoL'

Else

ST.AT.M= 'SV'

End If

ST.AT.N= E\A. $N_{E\A}$

ST.AT.T= E\A. $T_{E\A}$

ST.AT.BitT=CT.BitE

EndFor

Foreach Relashionship** RL ϵ CT.REL **do

CT':CT_{TXSDM}

CT'=DefineRelation(RL.DirC)

CT.RLT=getRelType(RL.RelTye)

If** RLT= 'Ass' **then

```
   ST.AT={ST.AT, DefineCoL(CT.KeyRef+BitE) }
EndIf
   If RLT= 'Agg' then
   ST.AT= { ST.AT, 'UDT('+ CT'.NCT + ')'}
EndIf
   If RLT= 'Comp' then
   ST.AT={ST.AT,defineCoL(CT'.E\A)}
  EndIf
   If RLT= 'ihner' then
   ST.SS=CT'.NCT + '-T'
    Ttable.STn= CT'.NCT
 EndIf
  EndFor
    TST=TST ∪{ST}
    TT=TT∪{Ttable}
EndFor
Return(TST,TT, TBit )
End Algorithm
```

Figure31. Algorithm of the transformation

IV. STORAGE AND PUBLISHING DATA FROM TXML DOCUMENTS INTO TORDB:

First, we outline the interest phases for modeling temporal data in XML documents. The process starts by defining a graph model for temporal temporal XML documents which track the historical data and then extended it to tree graph based on indexing techniques tree, where each node is identified by Dewey encoding concept. Through this model, we proceed to the next step; we will give the generated tables showing the Txpath according to the proposed models. In this section, we first outline the interest phases for modeling temporal data in XML documents. The process starts by defining a graph model for temporal temporal XML documents which track the historical data and then extended it to tree graph based on indexing techniques tree and each node is identified by Dewey encoding concept. In The next, we will give a representation of TORDB including varying time data enriched with additional semantic data. Through these two models, we will give an algorithm explaining the general process of the migration.

4.1 TXML documents Modelling and Dewey Numbering Schema:

Temporal XML database is built by a collection of TXML documents. XML provides the opportunity to have a sequence of element and attribute. On the other hand, a temporal xml document can be modeled as a rooted, structured and ordered graph. This work adopts the tree graph which itself is a connected acyclic schema in the Graph theoretical term in order to determine the relationship between the nodes. The tree will be consisting of the tags and content represented in the nodes of the tree. The stored data in the tree can be tracked by its path. The mechanism we use for integrating the varying time dimension to the proposed model consists in adding a new element to capture the changing value of the data over time. A temporal element is defined as H_element. Each H_element has an open –closed interval time (valid time start and valid time end).

Therefore, this data model leads to simplify the search of temporal data and reduce the storage space of TXML files. The tree diagram help us to convert and map the data from TXML documents into their corresponding in temporal object relational database (TORDB) where high scalability is needed to store different size of TXML files. In this proposed approach, we use a new index mechanism based in Dewey encoding feature. We assign to each node a Dewey ID to specify vector that define the path from the root to the leaf and identify the absolute position of the node. Furthermore, Ancestor–Descendant relationship between the nodes composed our proposed graph tree can be determinate and identify the number of the fragment, for this reason we divided the tree graph into several sub trees denoted ST (i) in order to simplify the migration of data. Figure1 shows the TXML files describing the history of a customer in banking managements and their system using a temporally representation.Figure1 shows the TXML files describing the history of a customer in banking managements and their system using a temporally representation.

Figure32. An example of tree schema for TXML files with Dewey ID

Let Consider the XML file shown in **Figure 33**, modeling a part of the customer and its relationship with the account element in banking system. Other elements like Balance_H table can be modeled in similar way and its child elements represent the grouped history of attribute values.

TXML schema Extraction

```
<customercustomerId="1">
<customerName> Soumiya </customerName>
<phone>0522435688</phone>
<Address_cust>
<street>Farah1</street>
<city>Settat</city>
<zipCode>26000</zipCode>
<Adress_H>
<vt_Start>2010-05-04</vt_Start>
<vt_End>2013-03-31 </vt_End>
</Adress_H>
</Address_cust>
<Address_cust>
<street>elqods</street>
<city>Casablanca</city>
<zipCode>26000</zipCode>
<Adress_H>
<vt_Start>2013-04-02</vt_Start>
<vt_End>2999-12-31 </vt_End>
</Adress_H>
</Address_cust>
<Customer_H>
<vt_Start>2013-05-04</vt_Start>
<vt_End>2019-05-04 </vt_End>
</Customer_H>
</customer>
<account Type= "Saving Account">
<Account_NO> F23
<vt_Start>2010-05-04</vt_Start>
<vt_End>2013-03-31 </vt_End>
<Balance_H>
<Balance Value= "349087">
<vt_Start>2010-05-04</vt_Start>
<vt_End>2012-03-31 </vt_End>
</Balance>
</Balance_H>
<Customer>
<CustomerRef>1</CustomerRef>
<vt_Start>2014-05-04</vt_Start>
<vt_End>2020-05-04 </vt_End>
</Customer>
</Account_NO>
</account>
```

Figure33.Example of TXML schema document

4.2 Definition of TXML Model for TORDB (TX-OR)

For our purpose, we proceed to define the structure of TX-OR index model, which is based on complex index key to reduce the storage space and to handle the records of TXML files with efficient manner. Each index item has to store different Dewey ID corresponding element.

TX-OR is a description of tree graph enriched with additional semantic data for modeling historical data by defining the structure, the relevant entities, their elements or attribute, and the relationships. This model is extracted by in-depth analysis of tree graph produced and shown above, considered TXML files as input. It takes into account concepts and feature offered by varying time data management and object relational model.

The TX-OR is defined in our approach as follow:

TX-OR= {(Node| N= Doc_ID, Parent_ID, Node_Tag, ST(i), Pos, Node_name, Node_val , VT_H, PathID)}

Where:

- **Doc_ID**:Document Identifier
- **Parent_ID:** Parent label of the node
- **Node_Tag:** denote the Dewey ID of the node.
- **Node_Name:** represents the name of the element or attribute
- **Node_value:** node representing values (text or numeric).
- **Pos:** position of the node in the tree graph
- **ST (i):** Subtree in the level i wich represents the fragment where the node belongs.
- **Path-ID:** simple path expression identifier
- **VT_H:** Valid_Time history label defined by two attributes VT_start (Valid Time start)and vt_End (Valid Time End) wich can be denoted:
  VT_H= {Key, vt_start, vt_End}

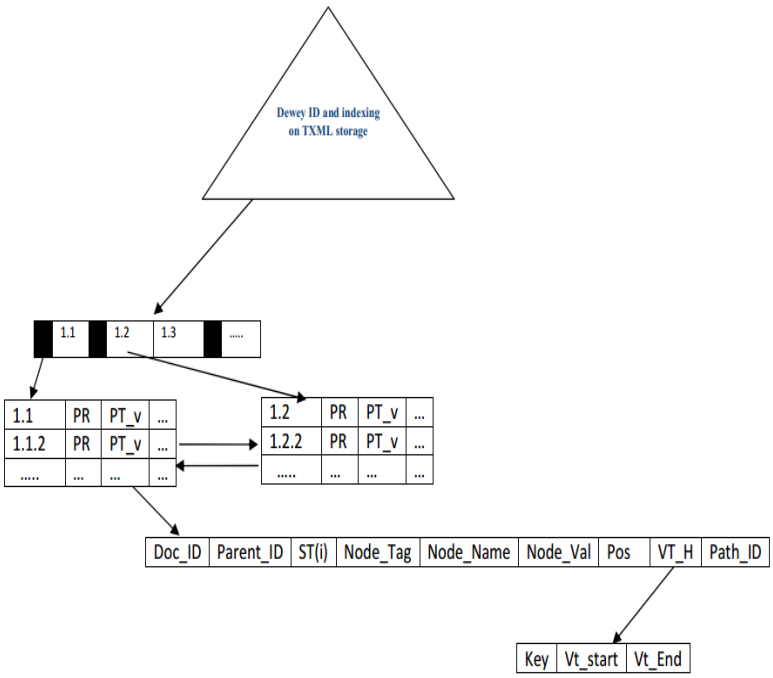


Figure34: TX-OR Index tree

Where:

- **PR:** Path Index pointer
- **PT_V:**value index pointer
- **VT_H:** valid Time period pointer

In TX_OR index tree, the node Number is a Dewey ID. The leaf node in the proposed graph represents a set of related nodes having a same prefix of the Dewey IDs. Each node contains a Dewey ID, path Index pointer (PR) which maintains the path of the each node, and value index pointer (PT_V) points to the node information so that allows us to access to the content of the element. A VT_H point is used to get the history of the Node expressed by valid time dimension.

4.3 Representation of TORDB schema for storing data:

This subsection, presents the different elements composed a TORDB schema by providing a general description of temporal OR database. The TORDB schema is defined as a set of temporal tables based on user-defined type UDT which contains varying-time attributes. Generally, a UDT can be denoted as tree tuples:

$$TORDB\_Model = \{UDT | UDT = UDT\_name, SA, VT\_P\}$$

Where:

- **UDT_name:** name of UDT
- **SA:** is a set of UDT's attricutes

SA= {A|A= (AN,AT,NL,MC,val)}where AN: is the name of attribute, AT: means data type which can be primitive , UDT or reference. NL: if the attribute accepts Null value or not.MC denotes if the Attribute is a single valued or Multi-valued, and val means the value of this attribute.

- **VT_P:** valide time period which composed by two Attribute:

$$VT\_P = \{VT\_start, VT\_end\}$$

4.4 General Algorithm for the conversion:

The **Figure 35** shows how the algorithm maps a TXML document into the TX-OR schema (Path_Index_Table and H_index_Table).this algorithm uses a temporal path based by loading TXML file, parseing it using DOM parser, and decompose the structure into different fragment. This is done by implementing several methods and function to store the data into proposed model TX-OR. In the Next, some methods and functions are done to get the UDT matching the input elements and convert the TX-OR schema into TORDB Model.

Algorithm Produce TORDBschema
Input:TX_OR ,XMLDoc
Output: TORDB schema
Begin
UDT: UDT :=Ø
CurrentNode: Node :=Ø
VTP:VT_P:= Ø
MC:string:=''
\\ Parse and Read XML document
T=Tree. Parse("DocXML")
For each *Sub tree ST ϵ TX-OR* ***do***
While *Node is not a leaf and Node ϵ TX-OR* ***do***
\\Check if Node ID (Dewey ID) exists in TX-OR table
If *Current Node.Tag==TX-OR.Node_Tag*

```
    CurrentNode.name=TX-OR. Node_Name
    CurrentNode.value = TX-OR. Node_val
    CurrentNode.parentID=TX-OR.ParentID
  If CurrentNode.VT_H !=Null then
 If      VT_H      ==      History_Index_Table.key      then
CurrentNode.VT_P.vt_start=History_Index_Table.vt_start
CurrentNode.VT_P.vt_end=History_IndexTable.vt_End
End if
End If
End while
End For
\\ store the data into TRORDB schema
\\ check if the UDT is already created in Temporal database
For each UDT ϵ TORDB schemado
If UDT.name == Element.NameThen
For each attribute ATϵ TORDB schema do
If UDT. AT.AN== CurrentNode.name Then
UDT.AT.Type=Nodeval.getType (CurrentNode)
UDT.A.MC=='Collection' then
UDT.A.MC= getAttributesfromCollection(AT)
If  AT.MC.AN== CurrentNode.name then
AT.MC.val=currentNode.Val
End if
If UDT.AT.MC=='Single valued' then
If UDT.AT.Type==Node val.getType (CurrentNode) then
UDT.AT.val= currentNode.Val
EndIf
UDT.VT_P.vt_start=curentNode. vt_start
UDT.VT_P.vt_End=curentNode. vt_End
EndIf
EndIf
EndFor
```

Figure35. Algorithm to convert the TX-OR into TORDB Model

V. CONCLUSION

In this chapter, we explained the basics phases of translating XML schema including varying time features into Temporal Database based on Object relational concepts. Currently no approach has proposed as a solution to extract temporal data model from TXML schema. Our method is done by providing a TXSDM from TXML schema file, and we use it as input which is enriched by additional semantics data. In the Next, we provided an algorithm to automate the translation process into TORDB. In addition; we discussed the issue of modeling TXML documents including temporal dimension and the migration of data into TORDB. We proposed a new solution takes the advantage of XML path, indexing mechanisms and Dewey encoding to identify the nodes. Through this model, we can define the Ancestor–Descendant relationship for each sub-tree rapidly. Another interesting feature of this approach is can keep the necessary data by capturing the temporal XML data and varying time element semantics, which can help to migrate the temporal data into temporal database.

CHAPTER V

MODELING BI-TEMPORAL PROPERTIES INTO BIG DATABASE: DATAWAREHOUSE AND NOSQL

I. INTRODUCTION

Companies are operating in a dynamic area that requires the ability to make proactive decisions in a complex situation to maintain or improve their business. The available data provided and handled by their activities is exploding due to the increasing use of various technologies. In consequence, there is a variety of decision support systems for managers to help to elaborate an analysis strategy rapidly. Before making a decision, the analysis phases can go along many dimensions, the most important of which is time. For this reason, historical information is used to identify certain characteristics in the evolution of data. Data analysis and Big Data originate from the longstanding domain of database management. They are based heavily on the storage, extraction, and optimization techniques that are stored in varying time management data. On the other hand, NoSQL and data warehousing are considered the core components of Big Data. They present the systems of modern data analysis as we know it today, using well-known techniques such as database statements, online analytical processing, and standard reporting tools. In recent times, the increasing use of temporal data has initiated different researches and growth efforts in these two environments in order to handle a huge amount of historical data in distributed systems.

One of disadvantages of the temporal object Relational model is the complexity and associated increased costs to store and to manipulate the varying time data. For this reason, we have chosen the temporal data warehouse and Nosql technologies to evaluate the ability to preserve the historical data.In this chapter,we attempt to propose a new conceptual modeling for temporal data warehouse that can be adopted as a reference in order to adopt a novel database management system in order to ensure the correctness of data and handle complex information that are dependent on the time.However, this works contribute with the use of Nosql technology by adopting Mongodb to store the huge amount of temporal data. We

Formalize the rules of the transformation from TORDB into Json documents. This chapter is structured as Follow:

1. Section I describes a general strategy for transformation process from UML class diagrams into temporal snowflake object relational schema (S-TORDW).this model based on snowflake structure using temporal object relational concepts and based on nested approach. We create a meta-Model that defines a new modelling of TDW including bitemporal data aspects. This is accomplished by using UML specification, stereotype, constraint and tagged values.

2. Section II presents a reliable, reasonable, and efficient method to convert the schema and migrate the temporal data from the implemented temporal object-relational database into MongoDB system. Our proposed approach provides a new model transformation from object-relational tables including Bitemporal data features towards documents oriented databases based on JSON files. Several rules are defined to facilitate the migration process.

II. TEMPORAL DATAWERHOUSE MODELLING USING TEMPORAL OBJECT RELATIONAL FEATURES

A Data warehouse is a multidimensional database that is used to store and provide access to large volumes of historical data, based on indicators for supporting the strategic decisions of organizations. Data in data warehouse has features of being a collection of subject-oriented, integrated, non-volatile and time-varying data. The two last characteristics allow changes to the data values without overwriting the existing values. Furthermore, data in a DWs must be stored in a way thus is secure, reliable, easy to retrieve and to manage [76]. It is mainly used only for using querying and consequently, it is important that querying technique performance is as high as possible. The structure of DWs is based on a multidimensional view of data usually represented as a start or a snowflake schema, consisting of fact and dimension tables [77]. In the recent decades, another active research, temporal database, deals with recording the history of the objects or the database activity..Even today, a large number of database applications based on time in nature, to make a better description and clearly some tasks of database systems.

The literature on temporal database offers three dimensions of time for temporal data support: transaction time, valid time and Bitemporal data which support the both. Bitemporal

data model our changing knowledge of the changing world, hence associate data values with facts and also specify when the facts are valid. The Knowledge acquired from these two research area, Management varying time data and data warehouse, led to the emergence of temporal data warehouse (TDW). A TDW is considered as a repository of historical information associated with time, and originating from multiple and heterogeneous sources, for the purpose to analyze, plan, react to changing business conditions and identify the relevant trends fast as possible.

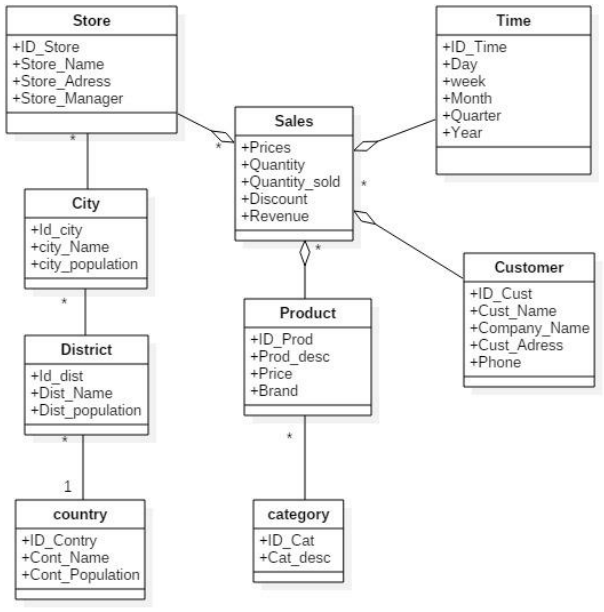
2.1 Process Of medelling and transforming UML into Logical Model:

In this section, we outline the important phase for modelling the conceptual and logical design schema. The process starts by defining a meta-model for temporal data warehouse, which is based on UML diagram class specification integrating bitemporal data features. In the Next, we propose EER representation of Temporal Meta-model which allows a better description of the entities, their attributes and their relationships. Through this model, we preceede to the last step, we will develop S-TORDW and SW-TORDW as a logical design uses the star and snowflake structure according to the specifications provided by Object relational Models such as Structured Type, References and nested table (NT) to manipulate the complex data.

2.1.1 Creation of Meta-Model for TDW:

Figure 36 presents the class diagram modeling data of store department as an example to be model with datawehouse structures.

Figure 36. Class Diagrams for department Store System



We need in this work to develop a meta-model which task is to record the changes of data in past, present or in the future. Although a class diagram is a general purpose language for system modeling, the temporal data warehouse has not been addressed. For this reason, it is necessary to propose a novel model for TDW modeling operation. A meta-Model is involved in system engineering to understand the data semantic meaning because it describes the elements, the relationships, constraints and attributes.

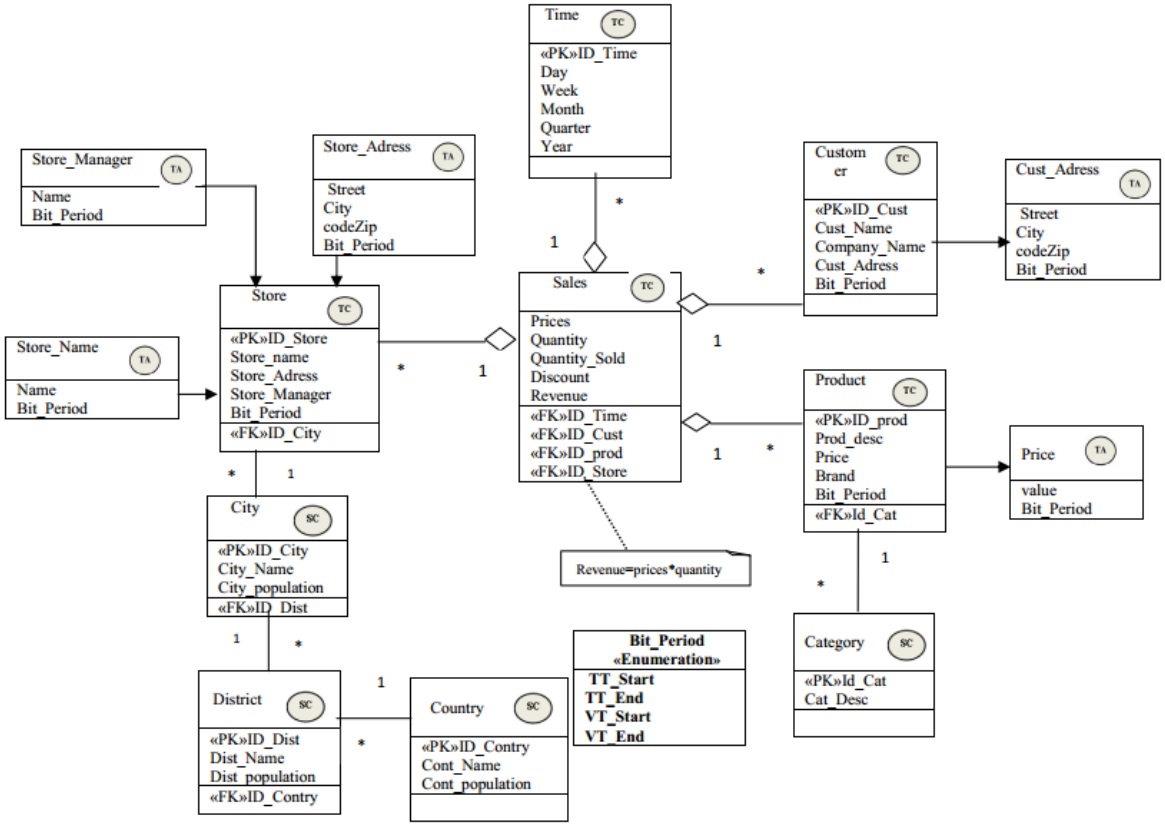


Figure 37. Meta-Model for TDW based on UML notation

2.1.2 Identification and definition of TEER Model

We choose the enhanced entity-relationship (EER model) since it is well known and widely used conceptual data model for database design. In this work, the proposed EER model is implemented by transforming the previous meta-Model and its specification into object Relational model. Therefore, this migration requires additional attributes and concepts for supporting time varying data to make a better description of temporal data. The aim goal of the EER model enriched with object relational features and bitemporal data is to simplify thecomprehension of the essential information stored in temporal data warehouse, and facilitate the implementation of the logical model. The TEER modeling plays a pivot role between the conceptual schemes expressed by UML and implementation of temporal data warehouse schema.

Consider the TEER model example as shown in **Figure 38**. In This example, we produce a TEER model of sales, product and category class described in the previous conceptual model.

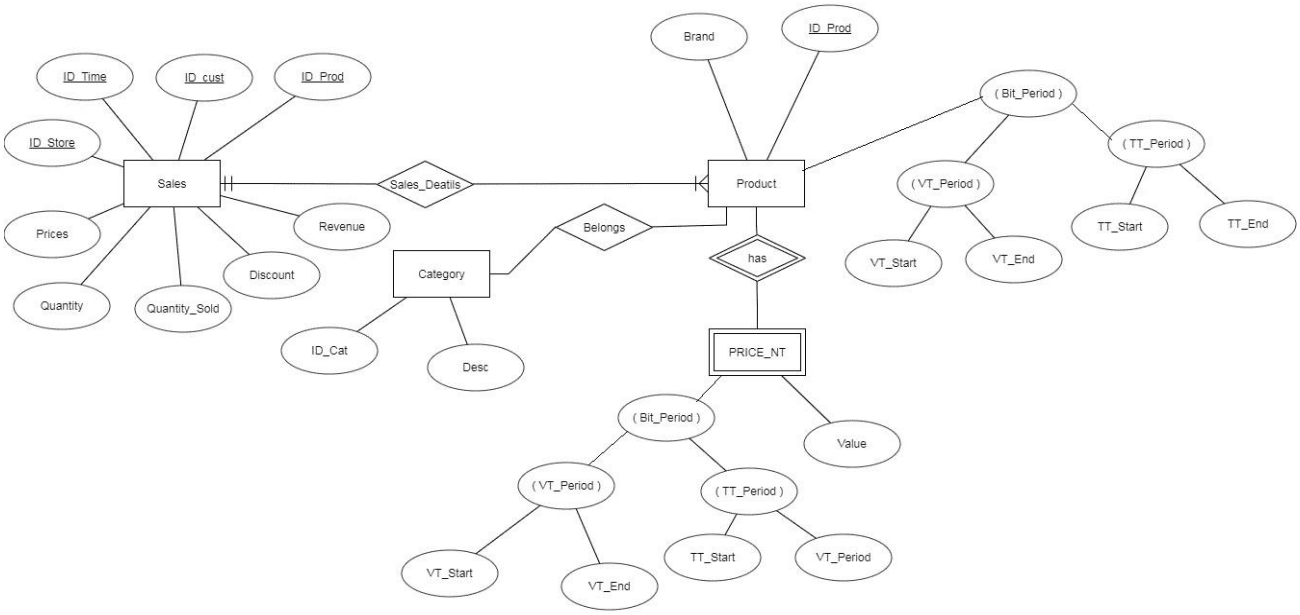


Figure38. The TEER model of sales, product and category tables

2.1.3 S-TORDW and SW-TORDW Model:

The SW-TORDB Model and S-TORDBW uses the snowflake and star schema for representation of DW structure including the specifications provided by the Object relational model to handle the complex data and objects. It consists of the fact table connected to several Dimension contain varying time data which are called temporal dimensions. To provide a history of the data and store their changes, the Bitemporal period should be kept at the attribute level. Attributes can be temporal or nontemporal. SW-TORDW and S-TORDW are based on Nested Approach to express hierarchy levels by the clustering of data in nested tables.

In the **Figure 39**, the schema of S-TORDW is represented in star schema at the logical design. In the S-TORDW a dimension's hierarchy is expressed as a structured type and Nested Table.

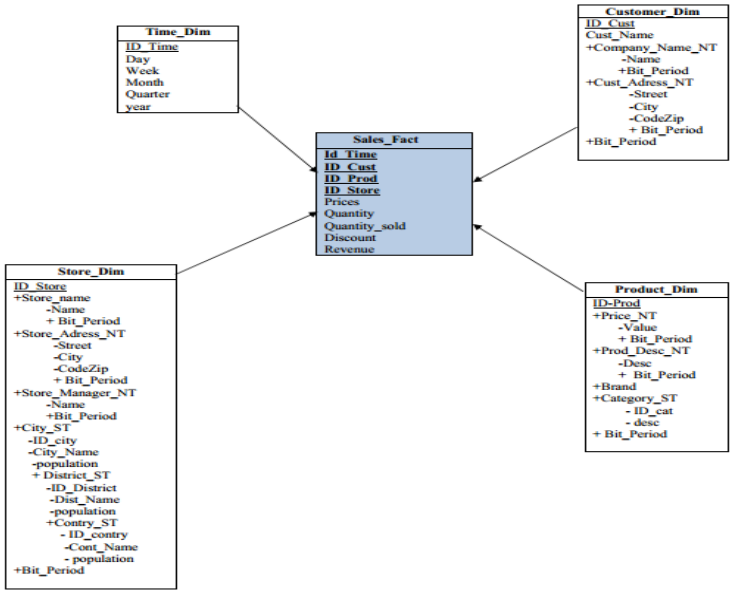
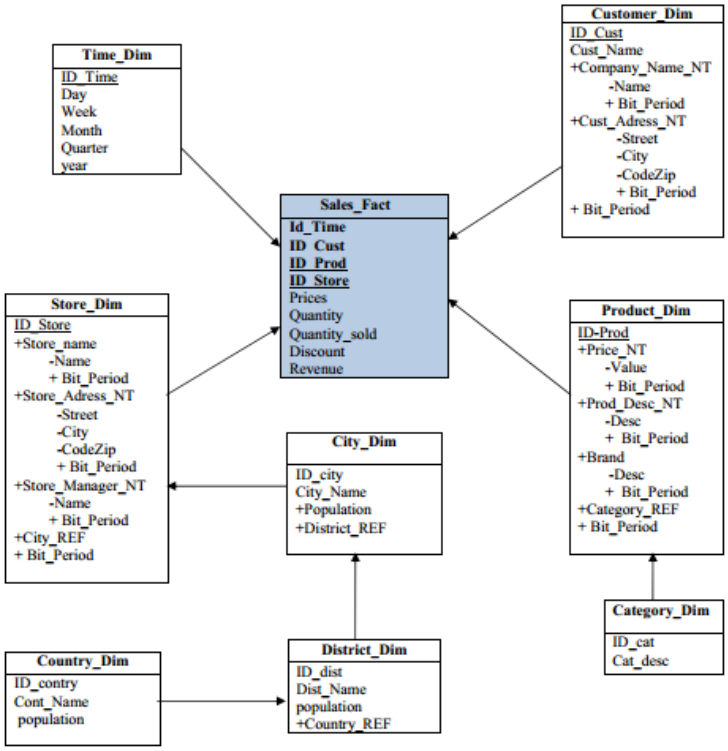


Figure 39. Temporal Star OR DW (S-TORDW) model

The SW-TORDW model which uses a snowflake schema is described in **Figure 40**. Each dimension table has a hierarchical attribute which is referred by a REFERENCE feature to express the foreign key attribute.

Figure 40: Temporal Snowflake OR DW (SW-TORDW) model



2.2 TORDB Queries Implementation:

We can translate the S-TORDW and SW-TORDW models into temporal relational queries (for the sake of simplicity, we will create TORDB queries only for Product Dimension) using Oracle 12 C.

TOR-QUERY

```
CREATE TYPE BIT_PERIOD AS OBJECT(
TT_START DATE,
TT_END DATE,
VT_START DATE,
VT_END DATE
)/
Create price_Type as object (
Value Number,
Bit-NT Bit_Period
)/
Create price is table of price_Type;
Create Type category_T as Object(
Id_cat Number,
Cat_Desc varchar(20))/
Create table category of category_T;
Create Product_Type as Object (
ID_prod,
Price_NT price,
Prod_desc varchar(20),
Brand varchar(20),
Category  REF category_Type)/
Create table Product_Dim of product_Type Constraint
Prod_Key Primary Key (Id_Prod), Nested table Price_NT
Store AS Price_Tab.
```

Figure 41. TORDB Queries for product Dimension in S-TORDW based on Star schema

III. MODELLING AND MIGRATING METHOD FROM TORDB INTO MONGODB

Today, we notice the advent of big data. There is a revolution going on in databases system management. With the development of data acquisition technologies, the information to be stored expands strikingly in volume and velocity. NoSQL database have evolved intensely in

the last years due to their flexible structure, less constrained than relational ones and offering faster access to information. Nosql is now used in many fields of industries and companies to support applications and systems not well served by relational and object relationaldatabase. NOSQL is released and widely used in many domains. Nosql database provides a mechanism for storage and retrieval of instructed data other than tabular or object relation used in relational

and object relational database respectively [78]. The Nosql model fulfils the scalability problems. Nosql databases are mostly open source , non relational , distributed and designed for large volumes of data across many clusters supporting replication and partitioning , parallel processing and what is usually called horizontal scaling [79].

One of the post popular and leading Nosql system management is Mongodb. Mongo db is an open source database based on distributed document released in 2009. It stores data as JSON-Like document with dynamic schemas (The format is called BSON) [80]. Mongodb is a document oriented database which holds a set of collection that are similar to relational database Tables where each collection contains a set of documents. One collection can hold different document with number of fields, content and size are not similar. It is a schema less that means the mongodb can be a distributed database and does not have a predefined schema so that allows providing additional data type and inserting new fields. Mongodb document is a set of key value pairs. Key values databases provide a hash table where the unique key and a pointer to a specific set of values are stored and data can be retrieved using the key. Mongodb is suitable tools for managing a distributing data between instances while using replication to improve the level of availability. With the information industry dependent on the time, developing rapidly in recent years, the dataset using in different system are becoming extremely large in volume with a high variety of data. For this reason , Mongodb has been invented to overcome the limitation of relational and Object Relational Database , and provides new mechanisms for managing huge amount of data that are different from the typical relational and object relational Model. In addition, the mongodb is adopted to handle the huge evolution of temporal data in distributed environments in which continue to rise in complex applications and social network.

In this section, we present the most necessary phases for translation process. We are going to propose several rules that allow developing the determined schema. In the fist, we present a comparison between temporal object relational database and Mongodb. After that we will

explain the different elements composed TORDB design, and then we will define the mongodb schema including bitemporal data. In this approach, we will not explain the rules for translation of TORDB model into their equivalent in the TJson_schema.

3.1 Comparison between Object relational database and Mongo db features:

Mongodb is an open source Nosql database based on oriented document structure. It was developed during 2007 by software called 10gen Company. Mongodb documents are stored in binary form that are similar to Json document model called BSON format, that supports such primitives data type (String, integer, date, Boolean, float and binary).the main features in mongodb are collections and documents. Mongodb documents have a flexible schema in which the collection dos not impose the necessary document schema. However, the temporal object relational databases require a table schema to be declared and created before inserting the data. Any temporal object table has a certain design that shows the relationships between them. Although mongodb does not support join operations as SQL databases, the relationship between documents can be represented using either the referenced or embedded concepts. The relationship in mongodb define how various documents logically dependant to each other. The relationship in mongodb can be expressed via embedded or referenced concepts. Where, embedded documents maintain all the related data in one document. These renormalized data representation allows applications handle and retrieve data from a single document.

In the following subsection, we focus on comparing mongodb to Temporal Object relational database. The table presents the main techniques of TORDB methodology and Mongodb.

TORDB	**Mongodb**
Database	Database
Temporal Object\Temporal Table	Temporal Collection
Row	Document
Column	Field
Data Type	Data Type
Primary Key	Id_Field
Simple UDT| REF | Nested table | Array | nested table(REF)	Referenced Document \ Embedded Document

Table16. The Differences between TORDB and Mongodb features

On the other hand, the referenced document stores the relationships separately between document by adding id-Field that references or links from one document to another. This

approach designs the normalized relationship. Actually, the difficult part of transformation process is how we can convert the relationships of the existing temporal object relational database into Mongodb document.

Each User Defined Time or UDT is converted to a MongoDB collection, in this example the collection name is customer contains Bitemporal period object. The customer_table holds data rows of customer_type objects. Also the customer collection will store customer objects with the same attribute as documents in BSON (Binary encoded JSON) format including varying time attributes. Then statements below shows the creation of customer collection and how we can generate documents with bitemporal Object in Mongodb :

Document1

```
db.createCollection("Customer") \\ Creation of Customer
Collection
db.Customer.insert(
{
"Customer_Id":23
"Name": "Soumiya"
"Bitemporal_data":
{
"Vt_start": "2020-02-03"   \\ Valid time start
"Vt_End":  "2028-02-28"   \\ Valid Time End
"TT_start": new Date()   \\ Transaction Time Start takes
the sysdate as Default value .
"TT_End":  "9999-12-31" }} \\ Transaction Time End
sets the value of insertion of transaction time to the highest
value ("31/12/9999")
```

Figure42. Example of Json file with bitemporal data

3.2 Temporal Json schema (TJSON-schema):

Tjson-schema is a representation of Json document with historical data that is enhanced with additional semantic data to offer a new description of mongodb document. We have chosen the most commonly used in the oriented documents that able to identify the relevant collections, their documents and their relationships. The model constructs a data reference design in order to facilitate the understandability of metadata stored in Json document integrating Bitemporal Data. Also, it overcomes the complications that occur during the transformation process.

Tjson-schema can be defined as follows:

Tjson-schema ***={TJ|TJ={ Col_name , Tdoc , RELCol}}***

Where:

- **Col_name** : each collection has a name , where the collection can be defined as a set of temporal and simple documents.
- **Tdoc**: denotes temporal Json documents including varying Time fields. Generally, the document in mongo db is a set of a key value pairs:

TDOC= {doc_ID, Fields, Bitemporal_Period, Primary Key}

The Json document uses Object Id as a default id which is generated during the creation of mongodb document.

- **Fields** = means a set of fields and can be identified by the following elements:

Field={ Field_Name , Field_Name} Where:

Field_Name is the name of the field. Field_Type= means data type (integer, string, date) .Bitemporal_Period means the embedded document in Json document which composed by:

Bitemporal_Period ={VT_start,VT_end,TT_start,TT_end}

- **Relcol**=Relations Tjson-schema. Each collection has a set of relationship between documents and can be identified as follow:

Relcol ={RelType,RelName,Dircol Where:

Each collection or Document has a relationship Type with other documents. The Reltype offers 2 types of relations: Embedded document of referencing document. Dircol is name of doc' that is related to the document Doc.

3.3 Transformation Rules From TORDB into mongodb:

- **Association**

R1: For the two UDTs Namely Customer_Type and Account_Type contain Bitemporal data and are related with association (1...N) relationship, using reference mechanism which allow retrieving data rapidly without using join between tables. The transformation of association in Mongodb consists on generating a new document where the Customer_Type will be referenced in Account_type document and the object type will be embedded in the both document.

Col1={NameCol1,NameDoc,Doc(Fields),Embedded(Bitemporal_Period)}

Col2={NameCol2,NameDoc,Doc(Fields),Object_ID(Doc),Embedded(Bitemporal_Period)}

The example below shows the structure of customer and account document, also the TORDB query Statement:

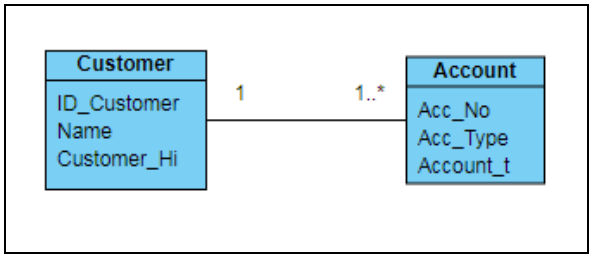


Figure43. An example of Association 1 to N relationship between Customer and Account

The creation will proceed according to the following syntax:

Col1={Customer,Customer_Json Document , (_id, Id_Cust, Name), Customer_h (VT_start, VT_End,TT_Start, TT_End)}

Col2={ Account, Account _Json Document , (_id, Num_Accout, Type_Account), Customer (_id),account_h (VT_start, VT_End,TT_Start, TT_End)}

Document 2:

```
{
   "_id":1345672
    "Num_Accout": 1
    "Type_Account": “Saving_Account”
    "account_h":
          {
           "VT_start": ISODATE(“2010-01-01”)
           "VT_End":  ISODATE(“2012-11-30”)
"TT_Start": ISODATE (“2010-01-03”)
            "TT_End": ISODATE(“2012-12-04”)
             }
"Customer":{
          "_id": “145673” }}
```

Figure44. Account_Document extraction with Mongodb

The defined TORDB Query for the corresponding example:

Query1: creation statement for account table

```
CREATE TYPE account_t AS OBJECT
(acc_no NUMBER,
acc_type varchar(20),
Customer REF customer_T,
account_h bitemporal_period);
```

```
CREATE TABLE account_table of account_t NESTED
TABLE  account_h STORE AS accounth_tab ;
```

Figure45. Account_Document extraction with Mongodb

On the other hand, in the case of association many to many (N,N) , the both document representing account and Branch_bank(see the example below), will be mapped in composition between the both document where each document integrates referenced document of another. The transformation result is:

Col1={NameCol1,NameDoc,Doc(Fields),Object_ID(Doc),Embedded(Bitemporal_Period)}

Col2={NameCol2,NameDoc,Doc(Fields),Object_ID(Doc),Embedded(Bitemporal_Period)}

The example shows the structure of branch_bank and customer Json document representing N to N Relationship:

Document 3:

```
Banch_bank document:
  {
     "_id":45679
      "branch_pk":1
      "city": “Casablanca”
      "phone": 06453278
      "Bitemporal_Period":
            {
             "VT_start": DATE(“2013-03-01”)
             "VT_End":  DATE(“9999-12-31”)
             "TT_Start": DATE (“2013-03-03”)
               "TT_End":DATE(“9999-12-31”)
               }
  "Customer":{
           "Customer_id": “ 2334” }}
  Customer_Json Document :
  {
     "_id":2334
      "Num_Cust":
      "Name_cust":
      "Bitemporal_Period":
            {
             "VT_start": DATE(“2010-01-01”)
             "VT_End":  DATE(“2012-11-30”)
             "TT_Start": DATE (“2010-01-03”)
               "TT_End": DATE(“2012-12-04”)}
  "Branch_bank":{
                        " _id": “45679” }}
```

Figure 46. Extraction of branch_bank Json file

- **Composition:**

In ORDB, the composition relationship is represented by declaring Nested table in the whole class which stores all attributes of the whole part. This relationship will be converted in a strong composition in mongo db between the account and balance documents. The transformation model generates one document account contains embedded collection of balance documents. The result of composition relationship:

Col={NameCol, NameDoc, Doc (Fields), Embedded (Bitemporal_Data), Embedded(Collection (Part_ Fields))}

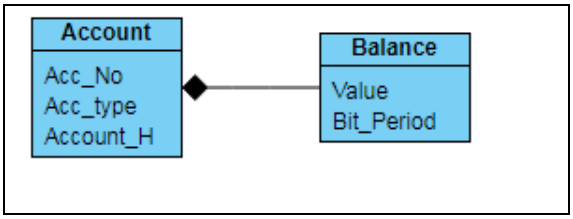


Figure 47. An example of Composition Class diagram

The corresponding TJson document of the example above:

Col={ Account, Account _Json Document , (_id, Num_Accout, Type_Account), account_h (VT_start,VT_End,TT_Start,TT_End),Balance{Value,Balance_h(VT_start,VT_End,TT_Start , TT_End)})

Account document embedded the balance document:

Document4:

```
{
   "_id":65789
    "Num_Accout": 2
    "Type_Account": “saving_account”
        "Bitemporal_Period":
          {
           "VT_start": “2015-01-01”
           "VT_End": “9999-12-31”
           "TT_Start": “2015-01-01”
            "TT_End": “9999-12-31”
             }
"Balance":[ {
          " value":324561.098

           "Blance_h":
          "VT_start": “2015-01-01”
            "VT_End": “9999-12-31”
            "TT_Start": “2015-01-01”
```

```
            "TT_End": "9999-12-31" }
{
          " value": 4356.098

           " Blance_h ":
{
     "VT_start": "2019-07-24"
          "VT_End": "2019-09-23"
          "TT_Start": "2019-07-25"
           "TT_End": "2019-09-27"
 }]}
```

Figure 48. Extraction of balance Json File for composition

- **Aggregation:**

An aggregation represents a binary relationship. It is a weak form of composition where the part is shareable and independent from the whole, and its properties can be linked with more than one whole class component. For example, if all the composites (whole) are deleted, the sheared part can be existed.

For the aggregation Relationship, the relation is identified a collection of (UDT) in addition of bitemporal data. Then, the branch bank can be composed by one or more than one shareable and existence independent collection. In mongodb, the Aggregation relationship will be considered as a referencing collection of _ID document represented the account document:

Col1={NameCol1,NameDoc,doc(Fields),Embedded(Bitemporal_Period)}}
Col2={NameCol2,NameDoc2,Doc(Fields),Referencing(collection(Doc1)+BitemporalPeriod),Embedded(Bitemporal_Period)}}

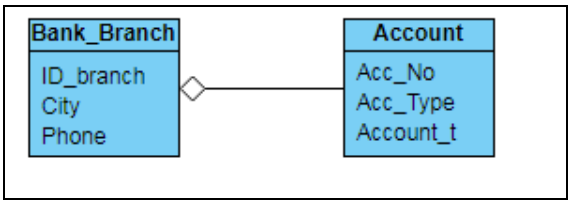


Figure 49. Aggregation relationship Class

The transformation will be defined as follow:

Col1={ Account, Account _Json Document , (_id, Num_Accout, Type_Account), account_h (VT_start, VT_End,TT_Start, TT_End))

Col2={Branch_Bank, Banch_bankdocument, {_id, branch_id, city,phone},Account(collection(_ID , account_h (VT_start, VT_End,TT_Start, TT_End)} , Banch_bank_h(VT_start, VT_End,TT_Start, TT_End)}

The following TJson document presents the example above:

Document5:

```
{
   "_id": 678
   "branch_id": 9
   "city": "Casablanca"
              "phone":074467543

            Accounts :[ {
              "Account":56432
              "Account_h":
                   {
               "VT_start": "2019-08-22"
                    "VT_End": "9999-12-31"
                    "TT_Start": "2019-08-25"
                     "TT_End": "9999-12-31" }}
            {
             "Account":980654
             " Account_h ":
                   {
              "VT_start": "2018-07-24"
                    "VT_End": "2019-09-23"
                    "TT_Start": "2018-07-25"
                     "TT_End": "2019-09-27" }
        }]}
```

Figure 50. Temporal Json File for Balance_Bank

The example below presents the TORDB Query Creation for Branch Bank and Account Tables:

Query 3: creation statement of the aggregation relationship

```
Create type Account_NT as object
( Account Account_T,
Account_h bitemporal_period) /

CREATE TYPE Account  IS TABLE OF  Account_NT ;
Create type branch_bank_type as object (
Branch_Num number,
Address varchar(40),
City varchar(20),
Accounts  Account,
Branch_h bit_period)

CREATE TABLE branch_bank_table of
branch_bank_type NESTED TABLE  accouns STORE AS
accounth_tab, NESTED TABLE branch_h  STORE AS
branch_tab ;
```

Figure51. TORDB for aggregation Relationship

- **Inheritance**

Is called also a generalization is a relationship between two classes or more , where one entity represent a parent or super class and the other one is considered as a child or sub class . The child inherits the behavior and all the properties of the parent.

The inheritance is a very important In ORDB. For the creation statement of UDT that represents the inheritance, we add the keyword under for the sub class. This model will be transformed in Mongo db by generating two documents separately modeling transaction and transfers Types. The transfer document maintains the same structure of transaction type with Additional properties of subtype is defined in the usual way with time varying features.

Col1={NameCol1,NameDoc,Doc(Fields),Embedded(Bitemporal_Period)}}

Col2={NameCol2,NameDoc,doc(Fields+doc1(Fields)), Embedded(Bitemporal_Period)}}

The details are illustrated in the following example:

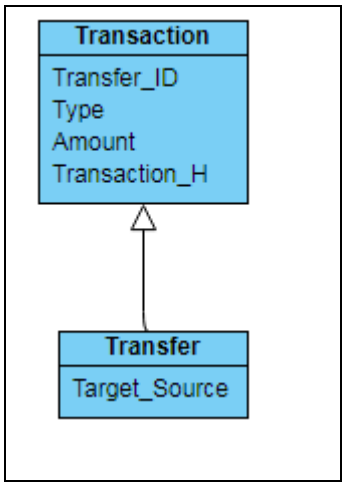


Figure52. Inheritance Relationship Example

Col1={Transaction, Transaction_Json Document doc(_id, trans_ID, Type_Trans, Amount), Transaction_h h (VT_start, VT_End,TT_Start, TT_End)}

Col2={Transfer, Transfer_Json Document ,(_id, trans_ID, Type_Trans, Amount, target_source), Transaction_h h (VT_start, VT_End,TT_Start, TT_End)}

The following TJson document presents inheritance relationship:

Document6: Temporal Json File for Transfer class

```
{
   "_id":87654
    " trans_ID ": 34
    " Type_Trans": “transfer”
     “Account”:9876
    “Amount”:  23450
    "Transaction_h":
          {
     "VT_start": “2018-04-02”
          "VT_End":  “2018-04-05”
          "TT_Start": “2018-04-02”
           "TT_End": “2018-04-04” }}
“target_source “: “soumiya}
```

Figure 53. Temporal JSON Document For Ihneritance Relationship

Ihneritace Creation Query in TORB using Under keyword:

Query 4: creation statement for inheritance relationship

```
Create or Replace type  Transaction_Type as object
(trans_ID number,
Type_Trans varchar(20),
Amount number,
Account REF account_type,
Transaction_h  bitemporal_period) NOT FINAL ;
Create table transaction_table of Transaction_type
NESTED TABLE Transaction_h  STORE AS
transactionh_tab;
Create or Replace type Transfer_T  UNDER
Transaction_Type (target_source varchar(20)) ;
```

Figure 54. Temporal Queries FOR Ihneritance Relationship

IV. CONCLUSION

This chapter described two solutions to the problem of modeling andtransforming of massive data based on temporal object Relational database to handle a large volume of historical information. In the first, we have presented a novel approach for conceptual design of TDW and the rules. This study simplifies comprehension of the transformation process of UML class diagram into TDW using ORDB and Bitemporal data. From UML class diagram we have extract data behaviour, relationships between class and constraint rules in order to provide our meta-Model in which contains varying time attributes. In the Next, We have described the

basics phases of modeling and converting Temporal Object relational including User defined time with Bitemporal data into oriented document database. To do that, we formalized the rules by specifying the basics steps involved in the temporal object-relational database design, in order to capture the relationship's type between objects. Furthermore, Temporal Json document design has defined including the varying time data by exploiting the range of powerful concepts provided by Nosql database. Currently, any work deals with the transformation process and its functionalities from TORDB into Mongodb.

CONCLUSION

In the last decades, with the emerging popularity of Object oriented applications, it is becoming easier than ever to find methodologies for migrating into different environment. The topic receiving less attention, though, is how to convert the temporal database adopting Object Relational Model. The temporal databases are naturally good sources for knowledge discovery. They could also be used to record all changes through time. This concept helps organizations in better decision making and to function in a well formulated manner. Therefore, the proposed methodologies describe howto migrate in great details with comprehensive examples. Our approach seeks to make the gap smaller by easing the effort to the transformation of TORDB. This is a major concept of this thesis. The methodology itself contains three major parts are:

1. implement a Conceptual model for temporal databases using UML and OCL
2. Extract a schema translation of each model for each stage of the migration
3. Formalize the rules of the migration which comprehensive examples

In our research, we presented an approach to generate ORDB with varying time features from class diagram schema. We use the class, relationships, and properties as input enriched with semantic data to provide class_schema, which will be transformed to TORDB model which characterize the temporal and non-temporal tables. To do that, we have developed an algorithm to automate the translation schema conversion. Therefore, we have presented a class diagram based on temporal concepts and incorporate OCL specification that can be useful for early identification of undesired problems, in order to simplify the comprehension of the system interaction and help for database migration and evolution using temporal object-relational concepts.

The next chapter covered the migration of Temporal object relational database based on SQL: 2011 standard into temporal object relational database. The main goal of this conversion is to overcome the lack emerged after combining TRDB with temporal features. Our contribution offered a precise description of a solution for migrating a TRDB into temporal object systems. We presented the basics phases to convert an RDB based on SQL: 2011

standard into TORDB, which contains varying time features, with a simple and practical method to capture the different relationships between classes, association, aggregation, composition, as well as inheritance.

In the last, we described the modeling the the transformation of massive data based on TORDB as database source into Nosql databases and the web semantic, in order to store , reuse and share the data on the web .We outlined the basic rules of the transformation by specifying the basics steps involved in the temporal object-relational database design.

The investigation of the relevant literature has shown that viewing the objects on top of temporal database and establishing gateways to the migration of existing temporal data between different systems. Besides, it seems that existing work does not produce a solution for integrating temporal features in different environment in general and the migration mechanism between different databases in particular.

REFERENCES

[1] A. Olive. “Conceptual Modeling of Information Systems”, Springer. Heidelberg , (2007) .

[2] J.Arlow. “UML 2 and the Unified Process Practical Object_Oriented Analysis and Design”. Edition 2, (2002).

[3] J.Cabot & E. Teniente . “Incremental Integrity Checking of UML/OCL Conceptual Schemas”. Systems and Software. 82 (9), 1459-1478,(2009).

[4] M.Wahler. “Using Patterns to Develop Consistent Design Constraints”. PhD Thesis,.ETH Zurich, Switzerland ,(2008).

[5] M. Kaufmann, P.M. Fischer, N. May & D. Kossmann. “Benchmarking Bitemporal Database Systems: Ready for the Future or Stuck in the Past?”. Proc EDBT (2014), pp.738-749,(2014).

[6] T. Richard Snodgrass & Ilsoo Ahn. “Temporal Databases”. IEEE Computer 19(9), pp. 35– 42,(1986)

[7] Alexander Menne. “The Potential of Temporal Databases for the Application in Data Analytics”. Netherlands, (2019).

[8] K. Kulkarni & J. Michels. “Temporal Features in SQL: 2011”. SIGMOD Records, Vol. 41, No. 3. (2012).

[9] K.Kulkarni & J.Michels. “Temporal Extensions in the SQL Standard”. In: Liu L., Özsu M.T. (eds) Encyclopedia of Database Systems. Springer, New York, NY.(2018).

[10] S. Radovanović, E. Milovanović & Nenad Aničić. “Performance Evaluation Of Temporal Features Defined”. In Oracle 12C Database. PROC. SYMORG, p 858—866, (2014).

[11] J.T. Wheeler . “Extracting a Relational Database Schema from a Document Database”. University of North Florida , (2017)

[12] J.W. Rahayu ,E. Chang, T.S. Dillon & D. Taniar. “Performance evaluation of the object-relational transformation methodology”. Data Knowledge Engineering 38(3):265–300, (2001).

[13] J.C Date. “An Introduction to Database System” .8th Edition, United States of America, Pearson Education, Inc. (2003).

[14] A.A. Maatuk . “Migrating Relational Databases into Object-Based and XML Databases”. Northambria University . Newcastle. (2009).

REFERENCES

[15] M. Wang. “Solving Relational Database Problems with ORDBMS in an Advanced Database Course”. Information Systems Education Journal (ISEDJ),(2011).

[16] W.H. Inmon. “Building the Data Warehouse”, 2nd edition, John Wiley & Sons, New York, (1996).

[17] A.A. Vaisman & E. Zimányi . “Temporal Datawarehousing”. In: Liu L., Özsu M.T. (eds) Encyclopedia of Database Systems. Springer, New York, NY, (2018).

[18] T. Zurek. “Optimisation of Partitioned Temporal Joins”. University of Edinburgh,(1997)

[19] RP. Padhy, MR. Patra & SC. Satapathy. “RDBMS to NoSQL: reviewing some next-generation non-relational database’s”. Int J Adv Eng Sci Technol;11(1):15–30.(2011).

[20] NQ .Mehmood & R. Culmone. “An ANT+ protocol based health care system”. 29th international conference on advanced information networking and applications workshops (WAINA). New York. p. 193–8, (2015).

[21] N. Leavitt. “ Will NoSQL Databases Live Up to Their Promise?". Computer, vol. 43, pp. 12-14, feb. (2010).

[22] M.Vadanyan . “Picking the Right NoSQL Database Tool” .(2015)

[23] R. Angles , C. Gutierrez. “Survey of graph database models”. ACM Computing Surveys (CSUR), 40(1):1–39, (2008).

[24] Koivunen, Marja-Riitta & E. Miller. “W3c semantic web activity”. Semantic Web Kick-Off in Finland .pp27-44.(2001).

[25]L.Quin.“Extensible Markup Language (XML)”.Word Wide Web https://www.w3.org/XML/ , (2002).

[26] C. M. Sperberg-McQueen & Henry S. Thompson . “XML Schema”. World Wide Web Consortium (W3C). http://www.w3.org/XML/Schema.(2000)

[27] T. M. Connolly & C. E .Begg. “Database systems: a practical approach to design, implementation, and management” . Pearson Education. (2005).

[28] Fallside, D. C. & Walmsley, P. “ XML schema part 0 : primer 2eme Edition”. W3C recommendation, 16, (2004).

[29] CE. Atay. “A Comparison of Attribute and Tuple Time Stamped Bitemporal Relational Data Models”. Proc of the Int Conf on Applied Computer Science. pp: 479--489, (2010).

[30] D. Petković . “Performance Issues Concerning Storage of Time-Variant Data”. Egyptian Computer Science Journal. Vol.38 . (2014).

[31] V.T.N Chau & Chittayasothorn . "A Temporal Object Relational SQL Language with Attribute Timestamping in a Temporal Transparency Environment". Data & Knowledge Engineering , Elsevier, vol. (67), p. 331--361, (2008).

[32] SY. Noh, S.K. Gadia & H. J Jang. "Comparisons of three data storage models in parametric temporal databases". Journal of central south university, Springer, vol .20, p 1919—1927. (2013).

[33] ISO/IEC 9075-2:2011. " Information technology - Database languages - SQL - Part 2: Foundation (SQL/Foundation)", (2011).

[34] S.Vesić, S. B.Nenad Aničić. "Use of the Temporal Concepts in Transaction Database", PROC. SYMORG, p 850--857, (2014).

[35] S. Radovanović, E. Milovanović & N.Aničić. "Performance Evaluation Of Temporal Features Defined In Oracle 12C Database". PROC. SYMORG, p 858--866,(2014).

[36] M. F. Golobisky & A. Vecchietti. "Mapping UML class diagrams into object-relational schemas". InProceedings of the Argentine Symposium on Software Engineering (ASSE 2005), 34 JAIIO, 65-79 pp, (2005).

[37] M. Wang ."Using UML For Object-Relational Database Systems Development: A framework" . Issues in Information Systems, VOL 9, No. 2 .(2008).

[38] Wai Yin Mok & D. P. Paper. "On transformations from UML models to object-relational databases". Proceedings of the 34th Annual Hawaii International Conference on System Sciences, Maui, HI, USA, 2001, pp. 10 pp.-, doi: 10.1109/HICSS.2001.926341.

[39]W.Y. Mok. "Designing nesting structures of user-defined types in object-relational databases". Information and Software Technology, Volume 49, Issues 9–10, Pages 1017-1029,(2007).

[40] J. Cabot , R. Clarisó & D. Riera. "Verification of UML/OCL Class Diagrams using Constraint Programming". Proceedings of the 2008 IEEE International Conference on Software Testing Verification and Validation Workshop, p.73-80, April 09-11, (2008).

[41] A. Queralt, A. Artale, D.Calvanese & E. Teniente. "OCL-Lite: Finite reasoning on UML/OCL conceptualschemas".Data Knowl.Eng, 73,1 -22,(2012).

[42] M.Gogolla & F.Hilken. "Model Validation and Verification Options in a Contemporary UML and OCL Analysis Tool". In: Oberweis, A., Reussner, R. (eds.) Proc. Modellierung (MODELLIERUNG' 2016). pp. 203–218. GI, LNI 254 ,(2016).

[43] L.Maciaszek & W. Kin-shin. "UML Dialect for Designing Object-Relational Databases".In challenge of information Technology Management in the 21 century, Information Ressources managementAssociation International Conference,pp 473-447 , (2000) .

[44] M. F. Golobisky & A. Vecchietti. “Mapping UML class diagrams into object-relational schemas”. In Proceedings of the Argentine Symposium on Software Engineering (ASSE 2005), 34 JAIIO, 65-79 pp, (2005).

[45] E. Marcos , B. Vela & J. M. Cavero. “A methodological approach for object-relational database design using UML” . Software and System Modeling, 2(1):59–75, (2003).

[46] E.Pardede, J. W. Rahayu & D. Taniar. “Mapping methods and query for aggregation and association in object-relational database using collection”. In ITCC (1), volume 1, pages 539–, Las Vegas, Nevada, USA. IEEE Computer Society. (2004).

[47] CASTELLANOS, Malű et SALTOR, Felix. “Semantic enrichment of database schemes: an object oriented approach”. In : Interoperability in Multidatabase Systems,. IMS'91. Proceedings., First International Workshop on. IEEE, 1991. p. 71-78,(1991).

[48] Castellanos & Mal. “Semantic Enrichment of interoperable databases”. In : Research Issues in Data Engineering, 1993: Interoperability in Multidatabase Systems, Proceedings RIDE-IMS'93., Third International Workshop on. IEEE, 1993. p. 126-129, (1993).

[49] A.A.Maatuk , A.Akhtar & Nick. ROSSITER. “Semantic enrichent: The first phase of relational database migration”. In : Innovations and Advances in Computer Sciences and Engineering. Springer Netherlands,p. 373-378 , (2010).

[50] A. El Alami & M. Bahaj. “Migration of the Relational Data Base (RDB) to the Object Relational Data Base (ORDB)”. World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol. 8, no 1, p. 242-248, (2014).

[51] J. Tekli. “An Overview on XML Semantic Disambiguation from Unstructured Text to Semi-Structured Data: Background, Applications, and Ongoing Challenges”. IEEE Transactions on Knowledge and Data, Vol: 28, Issue: 6 , pp: 1383 - 1407, (2016).

[52] F. Rizzolo & A. Vaisman. “Temporal XML: Modeling, indexing, and query processing”. The VLDB Journal , 17(5):1179–1212, (2008).

[53] F. Wang, X. Zhou & C. Zaniolo. “Using XML to Build Efficient Transaction-Time Temporal Database Systems on Relational Databases”. 22nd International Conference on Data Engineering (ICDE'06), Atlanta, GA, USA, pp. 131-131,(2006).

[54]G. Z. Qadah. “Indexing techniques for processing generalized XML documents”. Computer Standards and Interfaces, vol 49:34–43, (2017).

[55] SY. Noh, Shashi, K. Gadia & Ma.Shihe. “An XML-based methodology for parametric temporal database model implementation”. Journal of Systems and Software 81(6): 929-948,(2008).

[56] L. Ying, M. Jun & S. Yuyin. “Applying Dewey Encoding to Construct XML Index for Path and Keyword Query”. 2009 First International Workshop on Database Technology and Applications, Wuhan, Hubei, pp. 553-556, 2006

[57] A. Qtaish & K. Ahmad. “XAncestor: An efficient mapping approach for storing and querying XML documents in relational database using path-based technique”. Knowledge-Based Systems, 114(October):167–192, (2016).

[58] H. Hamrouni, F.Grandi & Z. Brahmia. “Deferred repair of inconsistencies resulting from retroactive updates of temporal XML currency data”. IJWIS 13(4): 485-519 ,(2017).

[59] R. Lawrence. “Integration and Virtualization of Relational SQL and NoSQL Systems Including MySQL and MongoDB”, in 2014 The International Conference on Computational Science and Computational Intelligence , Las Vegas, NV, USA, (2014).

[60] A. Singh. “Data Migration from Relational Database to MongoDB”.Global Journal of Computer Science and Technology:C Software & Data Engineering ,Vol 19,Issue 2 ,(2019).

[61] T. Jia, X. Zhao, Z. Wang, D. Gong &G. Ding . “Model Transformation and Data Migration from Relational Database to MongoDB”, in 2016 IEEE International Congress on Big Data (BigData Congress), San Francisco, CA, (2016).

[62] L. Stanescu, M. Brezovan, D.D. Burdescu . “An Algorithm for Mapping the Relational Databases To MongoDB--A Case Study”, International Journal of Computer Science & Applications, 14(1), (2017).

[63] L.Stanescu, M.Brezovan , CS. Spahui & DD. Burdescu . “A framework for mapping the mysql Databases to Mongodb– Algorithm, Implementation and experiments. International Journal of Computer Science and Applications, Vol.15, No. 1, pp. 65 – 82, (2018).

[64] T. Fouad & M. Bahaj. “Model Transformation From Object Relational Database to NoSQL Document database”. In 2019 International Conference on Networking, Information Systems & Security, Rabat, Morocco , (2019) .

[65] D. Chauhan & K.L. Bansal. "Using the Advantages of NoSQL: A case study on MongoDB", International Journal on Recent and Innovation Trends in Computing and Communication, vol 5(2), ISSN 232/-8169, (2017).

[66] S. Brahmia, Z. Brahmia, F. Grandi, R. Bouaziz, "A Disciplined Approach to Temporal Evolution and Versioning Support in JSON Data Stores" . In Emerging Technologies and Applications in Data Processing and Management", IGI Global, 114-133, 2019. DOI: 10.4018/978-1 -5225- 8446-9.ch006

[67] Y. Widyani, H. Laksmiwati & E. D. Bangun, "Mapping spatio-temporal disaster data into MongoDB",In 2016 International Conference on Data and Software Engineering (ICoDSE), Denpasa, Indonesia, (2016).

[68]A. Boicea, F. Radulescu & L. I. Agapin, "MongoDB vs Oracle -- Database Comparison", In 2012 Third International Conference on Emerging Intelligent Data and Web Technologies, Bucharest, (2012).

[69] Chau, V.T.N. and S. Chittayasothorn . "A Temporal Object Relational SQL Language with Attribute Timestamping in a Temporal Transparency Environment" .Data & Knowledge Engineering., vol 67, p. 331-361, (2008).

[70] T.fouad & M.BAHAJ. "Extracting UML Models And OCL Integrity Constraint From Object Relational Database", Journal of Theoretical and Applied Information Technology. Vol.96. No 4, February (2018).

[71] Ainars Auzins and Janis Eiduks and Alina Vasilevska and Reinis Dzenis .ObjectRelational Database Structure Model and Structure Optimisation",Applied Computer Systems, pp 28-36,vol 23,2018

[72] J. Cai, V.Eske, Xu. Wang, Semantic Web & Ontologies. http://www.mpi-inf.mpg.de/departments/d5/teaching/ss03/xml-seminar/talks/CaiEskeWang.pdf

[73] S. Balamurugan & Ayyasamy . " Performance Evaluation of Native XML database and XML Enabled database". IJARCSSE Journal, vol 7, (2017).

[74] M. Wang ,M. Xiao ,S. Peng : "A hybrid Index for Temporal Big" Data.FGCS Journal, vol 72 , , pp 264-272 , (2017).

[75] Z. Brahmia, F. Grandi & R. Bouaziz : "Changes To XML Namespaces in XML Schema and Their Effects on Associated Xml Documents Under Schema Versioning". In: Proceeding of the 11 Th International Conference on Digital Information Managment (ICDIM), (2016).

[76] G.GARANI & CE.ATAY. "Comparison of different temporal data warehouses approaches".The online Journal of science and technology . Vol7(2),(2017).

[77] E.Malinowski & E. Zimanyi. "Logical representation of a conceptual Model for spatial data warehouses".GeoInformatica.vol 11(4)pp431 -457,(2007).

[78] Z.Gansen, L.Qiaoying, L.Libo & L.Zijing. “Schema Conversion Model of SQL Database to NoSQL”. In 2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. Guangdong, China. DOI: 10.1109/3PGCIC.2014.137, (2014).

[79] Byrne, B, Nelson, David & Jayakumar. “Big Data Technology - Can We Abandon the Teaching of Normalisation?”, in 2017 I9th annual International Conference on Education and New Learning Technologies, Barcelona, Spain.(2017).

[80] A. Boicea, F. Radulescu & L. I. Agapin. “MongoDB vs Oracle -- nDatabase Comparison”, In 2012 Third International Conference on Emerging Intelligent Data and Web Technologies, Bucharest, (2012).