Journal of Theoretical and Applied Information Technology
30th September 2024. Vol.102. No. 18
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EXTRACTING A MULTIDIMENSIONAL CONCEPTUAL MODEL FROM A DATA LAKE USING AN MDA APPROACH

LAMYA OUKHOUYA[1] , ANASS EL HADDADI[2], BRAHIM ER-RAHA[1], ASMA SBAI[3]

[1]ESTIDMA team, National School of Applied Sciences, Agadir, Morocco
[2]SDIC team, National School of Applied Sciences, Al Hoceima, Morocco
[3]LBH Laboratory, Faculty of Medicine and Pharmacy, Marrakech, Morocco
E-mail: [1]l.oukhouya@uiz.ac.ma, [2]a.elhaddadi@uae.ac.ma, [1]b.raha@uiz.ac.ma , [3]asma.sbai@uca.ac.ma

ID 55342 Submission	Editorial Screening	Conditional Acceptance	Final Revision Acceptance
15-08-24	20-08-2024	11-09-2024	15-09-2024

ABSTRACT

Data warehouse design is based on a thorough analysis of an organization's operational data sources. These sources are then reorganized into conceptual models to enable multidimensional analyses. However, extracting a model with a unified multidimensional structure across all of these data sources presents some difficulties because it is necessary to have full documentation of data sources to perform this task. To overcome these issues, this article presents an approach to modernizing data warehouses using a data lake as a source of consolidating data from the organization's operational sources. Our approach begins by extracting the relational physical model from each data source, which is then integrated into the data lake using a domain ontology. This ontology helps detect duplicate elements in physical models and merge them into a unified relational model for the data lake. Finally, from this unified model, we extract the multidimensional conceptual model. This approach is automated by aligning with the model-driven architecture. We also validated our contribution with a prototype whose objective is to design a tool for the automatic extraction of the conceptual model from a data lake consolidating the data sources. Furthermore, a comparison between our prototype and a manual process carried out by a computer scientist revealed that our prototype simplifies the extraction task and saves significant time compared to the manual process.

Keywords: *D*ata Warehouse , *D*ata *L*ake , *M*ultidimensional *M*odel, *M*etadata , *M*DA *A*pproach.

1. INTRODUCTION

Facts and dimensions are multidimensional concepts that captivate decision-makers interests as they are linked to dynamically unfolding events within organizations [1]. Typically, these concepts are modeled as tables in operational data sources. Thus, one of the main steps in designing a data warehouse is to detect tables that model facts and dimensions. However, this task presented by extracting the multidimensional model from data sources may require a high level of expertise in application, and it is often tedious and time-consuming for designers[2].

These data sources are characterized by various types of structured as well as unstructured data whose physical grouping makes it possible to naturally create the so-called data lakes [3]. The latter refers to a massive aggregation or grouping of data preventing heterogeneous data sources from constituting the information system, thus providing leaders with a global view of the organization's data. These data can be organized according to a multidimensional data model to support certain types of decision-making processes. However, no design techniques exist to map a data lake to a conceptual schema to design a data warehouses.

Furthermore, the adoption of the data lakes with data warehouses is part of the modernization aspect of the latter especially when used as a source of consolidation for the data warehouse[4]. Indeed, data warehouses modernization involves reorganizing and strengthening the infrastructure to take advantage of the latest technological advances[5]. This includes adopting the most recent data storage, processing, and analysis solutions which are often based on distributed architectures, to improve scalability, agility, and overall system performance.

This being the case, our present paper seeks to address these problems by presenting an automatic process called ToExtractMD for extracting a multidimensional model from a data lake consolidating the operational data sources of the organization. This process presents the first part of a data warehousing project, while [6] presented the second part aimed at implementing the multidimensional model on NoSQL and relational systems.

The information system of the higher education sector in Morocco is used as a case study. Indubitably, our approach is stimulated by the perspective that the digitization of information systems in Moroccan universities can lead to a more sophisticated decision-making system, offering the opportunity to use in-depth analysis to guide the strategic choices made by leaders. These systems, such as e-learning management systems (LMS), MOOCs, Enterprise Resource Planning (ERP), and business applications all rely on relational databases, which disperse information, thus creating information silos. The extraction of value is therefore compromised, or even nonexistent in some cases. In addition, data silos lead to data duplication, reflecting the lack of a data integration approach in these institutions [7]. Thus, the use of data lakes has aroused our interest not only as it will allow us to solve the problems mentioned above, but also because of the capacity of these systems to store large quantities of data, sometimes reaching several terabytes, and for their fast data generation [8]. In addition, the implementation of analytical processes requires the creation of data warehouses, where the use of NoSQL systems becomes necessary due to the specificity of data from different systems [9].

The objective of this article is to design a modern decision support system capable of adopting the most sophisticated approaches in this field. Specifically, it aims to combine the concepts of data lake and data warehouse within the same decision-making architecture. In this article, we present the first phase, where the data lake is designed as a data consolidation zone for the data warehouse, integrating all data sources from a Moroccan institution or university. From this data lake, a multidimensional model will be extracted, which will be used in the process described in work [6] to design the data warehouse. This approach will provide decision-makers with a comprehensive view of university data, enabling the application of advanced analytical processes, while also modernizing traditional data warehouse design methods through the integration of technologies such as NoSQL systems, which offer the capability to manage large-scale data.

The remainder of this article is organized as follows. Section 2 presents a detailed description of the approach adopted. Section 3 describes a state of the art of work carried out in this area. Section 4 presents the ToExtratMD process. Section 5 validates our proposition through an experiment.Section 6 discussion and future work , and finally, section 6 includes the general conclusion.

2. STATE OF THE ART

The design of a data warehouse is first based on a first level which involves structuring of data from data sources in the form of a conceptual data model. In this sense, our problem concerns the extraction of the multidimensional structure from a lake consolidating and integrating data sources, and reorganizing it into a conceptual model. In the literature, many works propose processes for integrating data sources into the data lake, while others propose the implementation of a data lake on data warehouses.

This article [10] proposes an approach to aggregate and harmonize relational databases in a data lake. To do this, the authors propose a semantic layer on top of the data lake, composed of a global domain ontology and a set of correspondences between the entities of the ontology and the data lake. This semantic layer is designed using the Ontop tool, which allows relational databases to be exposed as an RDF graph for querying using SPARQL queries. These can also be translated into SQL queries once the matches have been established. Furthermore, the data lake is implemented in this approach by the Hadoop ecosystem, while the ingestion of the relational databases is carried out by Apache SQOOP. In the same vein, this work [11] presents an intelligent approach to solve the problem of integrating data from heterogeneous sources for their use in analytical processes. To this end, the authors proposed a hybrid architecture to drive and orchestrate data from a data lake to a data warehouse. This architecture is divided into four functional layers: Acquisition, Exploration, Semantics, and visualization. The acquisition layer is responsible for obtaining and importing data from heterogeneous sources with a view to storing it in the data lake in raw format. The second layer is exploration in which data patterns are discovered and grouped based on metadata, which are redirected to the next layer (semantic layer) for advanced processing by machine learning. The third layer is the semantic layer dedicated to preparing new datasets to create knowledge. At this layer, the data is cleaned, normalized and harmonized for schematic reading whether by a data warehouse or another database management system (relational or NoSQL). Finally, the visualization layer allows the integration of data sets into reports, algorithms and/or simulations to create different types of information, such as diagnostics, predictions and prescriptions. In another article [12], the same authors detail the operation of the semantic layer presented previously. Indeed, to integrate the data into the data lake, the authors used an ontology to resolve the interoperability of data sources. This approach takes massive data sets as input and produces an OWL ontology as output. More precisely, the method consists of wrapping each data source in a data lake, which will then be transformed into a local ontology. Afterwards, these local ontologies are combined into a global OWL ontology, allowing the visualization layer to meaningfully leverage information from multiple data sources to generate statistics and reports. In the same direction, in this article [13], the authors present an approach for designing a data warehouse from a massive data source. The proposed approach uses a data lake to integrate the different social media data sources with the aim of designing a NoSQL data warehouse from the data lake.

in this work [14],the authors have presented a data lake architecture for integrating various biomedical data sources in a single location. This architecture is based on HDFS storage and apache drill for real-time data analysis. The aim of this architecture is to use the data lake for storage and consolidation of the various data sources, and the data warehouse for reading and analysis. In the same context , this work [26] presents an approach for designing a medical data lake. Specifically, it describes a functional architecture of the data lake, consisting of multiple layers, each responsible for a specific task. The first layer, called ingestion, extracts data from various medical sources. The storage layer stores the ingested data in a centralized repository within the data lake. The transformation layer applies normalization and data cleansing processes. Finally, the interaction layer allows users to visualize the transformed data. Another work [27] proposes the design of a data lake, this time based on a technological architecture built around the Hadoop ecosystem. Storage is handled by HDFS files, while data and metadata management are ensured by Apache Hive. Data ingestion is carried out using Apache Flume and Sqoop. For data processing, Apache Spark was chosen, along with Apache Kafka for streaming processing. Finally, task scheduling is managed by Apache Oozie and Apache Airflow.

The table below summarizes all the work in relation to these characteristics:

- ✓ Multiple data sources (a);
- ✓ Integration of data into the data lake (b)
- ✓ Extraction of the conceptual model from data lake (c),
- ✓ Automation of the approach(d).

Table 1: Comparative analysis of source integration and conceptual model extraction processes in a data lake

Article/Characteristics	(a)	(b)	(c)	(d)
[10]	x	x		
[11]	x	x		
[12]	x	x		x
[13]	x	x		
[14]	x	x		x
[25]	x	x		x
[26]	x	x		x

From this in-depth analysis of this work, we note that the work [12] resolves the semantic conflicts of the sources to be integrated by constructing a global ontology. Likewise, the works [10] and [11] also used a global ontology obtained by matching the local ontologies of each data source to be integrated. Although ontologies are the most appropriate solution for integrating data into a data lake, using mappings between local ontologies to design a global ontology does not ensure maintainability and scalability because modifying, adding or removing local ontologies can easily affect other mappings to the global ontology. The same goes for the approach used in [12], where the authors used the automatic knowledge graph for the design of the overall ontology without introducing business experts as a manual validation process to resolve the ambiguity of the terms in their contexts. In our proposal, we designed a global domain ontology for the data lake by introducing business experts in the middle of processing to increase the reliability of our extracted knowledge model.

In the work [12-13], the authors proposed architectures for the data lake through which they can design a data warehouse. however, they did not mention the proposed approach to designing a data warehouse from the data lake. Our approach, on the other hand, integrates data sources into the data lake and also extracts the multidimensional structure from it.

Finally, for the automation of the approach, works [12,14,25-26] automatically integrate data sources into the data lake without addressing model or data warehouse design. Our approach is more comprehensive, automating all these steps while adhering to a model-driven approach. This includes the integration of data sources, the extraction of multidimensional model data, and the automated implementation of this model in both NoSQL and relational systems Presented as the second process in our project, named ToCreateDWH, and detailed in the work [6].

As a concluding note, our work addresses the limitations of the studies presented in this section. Our approach integrates the data lake as a source of consolidation for the data warehouse, allowing the centralization and integration of heterogeneous data into the lake. We then applied a semantic layer on top of the data lake to eliminate semantic conflicts, resulting in a unified data lake model. Finally, from this unified model, we extracted a multidimensional model based on the concepts of facts and dimensions. It is also worth noting that our approach follows a model-driven methodology, enabling full automation of the entire process.

The next section details the ToExtract process, explaining our method for data consolidation in the data lake and the extraction of the multidimensional model.

3. OVERVIEW OF TOEXTRACTMD PROCESS

The work presented in this article aims to extract a multidimensional model from a data lake. To accomplish this, we defined the ToExtractMD process, which is responsible for analyzing the consolidated data sources in the data lake, revealing their structure to extract a multidimensional conceptual model. It applies a sequence of transformations from a set of relational databases representing the input to the process. First, it extracts the relational structure, characterized by tables, columns, primary keys, foreign keys, and records.

At the same time, it reorganizes this structure in the form of a relational physical model specific to each data source. Next, ToExtractMD establishes a semantic layer over the data lake, whose function is to create mappings between the different tables, thus leading to a unique relational model for the data lake. Finally, from the unified relational physical model, the process performs transformations to derive the multidimensional concepts and present them as a multidimensional conceptual model. Figure 1 illustrates our ToExtractMD process.

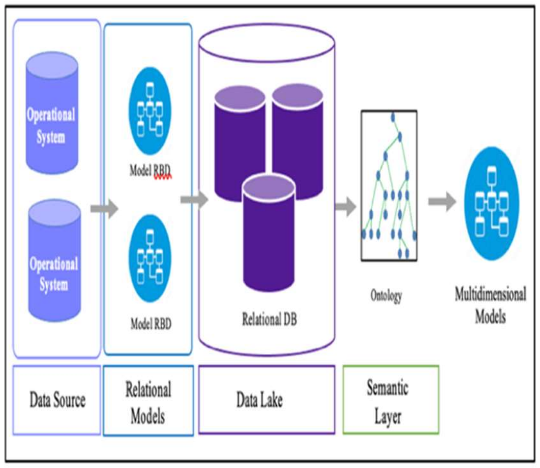


Figure 1: Our ToExtactMD Process

More concretely, our ToExtractMD process is composed of 3 steps, namely, DataSource2RelationalModel, Relational-ModelsMerge, and DataLake2-MultdimentionalModel, as schown in figure 2 .

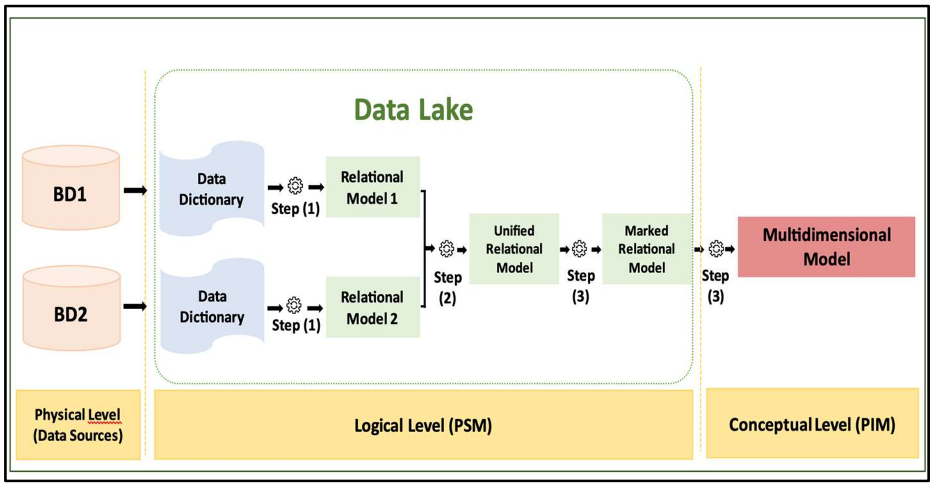


Figure 2 : Step of ToExtractMD Process

The first step DataSource2RelationalModel is devoted to the extraction of the relational model of each data source of the information system. To do this, we used a set of techniques to automate the process, namely:

- ✓ **The data dictionary.** The data dictionary is used as a first phase to extract the relational structure from each database base.
- ✓ **JAVA methods.** We chose the DatabaseMetadata API methods to access the data dictionary and manipulate the metadata stored there.
- ✓ **The relational model.** We consider the CWM relational metamodel to represent the relational structure at the logical level.

The second step in the ToExtractTM process is RelationalModelsMerge. Its role is to merge relational physical models into a unified model for the data lake. In this regard, we used a semantic web approach to align and harmonize data by applying semantic correspondences between concepts and relationships allowing for interoperability between the different relational models corresponding to the data sources. This approach includes :

- ✓ **A domain ontology.** We started from a domain ontology to model knowledge in the university setting. This ontology named OntoDL is designed to represent the knowledge and concepts of systems related to teaching management and students in academic institutions, in particular.
- ✓ **Semantic manipulation tools and interface.** We used the Apache Jena tool and the OWL API to manipulate the domain ontology and cross-reference it with the physical models designed in the previous step to obtain a unified model.

Finally, ToExtractMD process ends with the DataLake2MultdimentionalModel step, which extracts the multidimensional conceptual model from the unified relational physical model of the data lake.

ToExtractMD process is automated using a model-driven approach, which is a development standard through model manipulation, based on the separation of business and technical concerns and automation of transformations[15].

We opted for two levels to carry out our process, each of which is described by a model:

- ✓ **The PSM(Platform Specific Model) level.** It is presented by a model specific to the implementation technology [16]. In our approach, we use a physical relational model to describe the logical level which represents the model extracted from each data source as well as the unifying model of the data lake.
- ✓ **The PIM(Platform independent model) level.** It describes a business model specific to the application while ignoring the technical aspects[16]. The PIM model represents the multidimensional conceptual model extracted from the data lake.

The transition between these levels is carried out following M2M(model to model) [17], type transformations and we have chosen to use the general JAVA language to ensure these transformations. These cover the transition between the relational physical model PSM and the multidimensional conceptual model PIM. In addition, this language is also used to extract the relational structure and represent it according to the CWM (Common Warehouse Metamodel) metamodel . It should be noted that the nature of the ToExtractMD process is a reverse engineering process going from the physical level to the conceptual level.

In the upcoming section, we will detail the transition between the stages of the ToExtractMD process, namely, DataSource2RelationalModel, RelationalModelsMerge and DataLake2MultdimentionalModel through three points, the input, the output, and the applied transformation rules.

4. TOEXTRACMD PROCESS

The ToExtracMD process aims to extract the conceptual model from the relational data lake consolidating an organization's data sources. Figure 2 illustrates this process as already described in Section 3 through a reverse engineering process that executes a series of transformations, the input of which is a Physical Relational Model (PSM) and the output a multidimensional conceptual model (PIM). The data dictionary holds the role of analyzing the data source using the metadata to describe it according to a logical form (PSM model), whereas the semantic layer is presented by establishing a domain ontology that represents the embodiment of the data lake concept in our ToExtactTMD process. Each model in this approach conforms to its meta-model. However, we did not use languages dedicated to transformation between models such as QVT or ATL. This is explained by the fact that the calculated measures, access, and manipulation of metadata in the dictionary as well as the design of the ontology are not achievable with these languages. The process uses the JAVA language to access the metadata, determine the model, mark it with multidimensional elements, and deploy it as a multidimensional conceptual model. All models are represented in XMI(XML metadata interchange) format, an OMG standard for exchanging data models [18].

In the upcoming subsections, we will detail the ToExtractMD transformation process through the three major points that we have cited, namely, designing the relational model of each data source (DataSource2RelationalModel), merging the relational models (RelationalModelsMerge), and finally extracting the multidimensional model (DataLake2MultdimentionalModel).

4.1. First Step: DataSource2RelationalModel Transformation

In this section, we will present the DataSource2RelationalModeltransforma-tion executed in two tasks, namely, the extraction of the relational structure from the data source, and the reorganization of this into a physical relational model. We start by defining the elements required for this transformation, by defining the input and output, as well as the associated transformation rules for each of these tasks.

4.1.1 Input: Relational data source

The input to our process is a set of relational databases representing the data sources of the information system. However, to perform the DataSource2RelationalModel transformation, the input to this step must first be able to analyze these sources. This will be done based on the metadata stored in the data dictionary. Indeed, in the context of relational databases, information about data structure can be extracted using metadata which can be stored in data dictionaries. Therefore, the latter is used in our process to designate the structure of the data, as well as the calculation of measures attributing to the derivation of the multidimensional structure.

4.1.2 Output: PSM relational model

In our approach, the PSM model represents the logical level involving a platform-independent description. Consequently, the output of this transformation is a physical model described according to the relational approach; this is designed by reference to its metamodel, thus designating the physical structure of relational databases. For this reason, we opted for the CWM metamodel to represent the PSM relational model.

In principle, CWM provides a set of metamodels that are sufficiently comprehensive to be able to model an entire data warehouse, including data sources [19]. We have used a relational metamodel to represent all aspects of relational databases, a multidimensional metamodel to represent commonly used multidimensional data structures, and an XML metamodel to represent common metadata describing XML data resources. In our transformation, our PSM model is represented by the CWM relational metamodel describing the data source, mainly the data dictionary according to the following relational concepts: table, columns, foreign key, and primary key. Figure 3 illustrates part of the CWM relational metamodel.

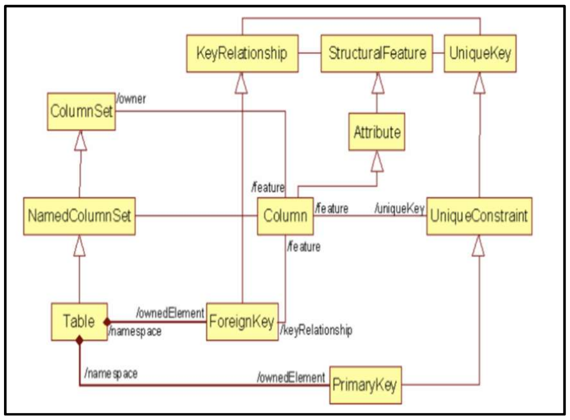


Figure 3: Part of the CWM relational metamodel[20].

4.1.3 DataSource2RelationalModel transformation rules

In this section, we present the rules for developing the PSM relational model from the data dictionary. We will begin our transformation process by extracting relational elements from the data dictionary, then this structure will be organized in our CWM relational model according to the XMI format. The passage between these elements is described in the JAVA language.

4.1.3.1 The extraction of relational elements

The process of identifying the relational structure is based on the DatabaseMetaData JAVA API, which consists of several classes and methods to facilitate metadata retrieval [21]. Based on these components, this API makes it possible to build more generic applications, capable of adapting to different databases by dynamically retrieving metadata information. Table 2 below includes all the DatabaseMetadata API methods used to extract the relational structure.

Equally important, the rules used to extract the relational structure using the metadata of the data dictionary are represented through two major processes; the acquisition process and the analysis process.

Table 2: Methods Utilized from DatabaseMetadata API

Method	Meaning/Explanation
getMetaData ()	This method returns database objects from a connection.
getTables ()	Method that returns all tables contained in the database. This method returns a list which is then scanned to obtain information about each table.
getColumns ()	A method that retrieves the names of all the attributes of a given table defined by the parameter as well as their characteristics (e.g. name, type, etc.).
getPrimaryKeys ()	A method that is used to retrieve the primary key metadata of a specific table in a database. The ResultSet class is used by this method to return the primary key name, along with a set of related information, such as the sequence of primary key columns.
getImportedKeys ()	A method used to retrieve foreign key metadata from a specific table in a database. The ResultSet class is also used by this method to extract the names of foreign keys, parent tables, and child tables.

✓ **The Acquisition Process :**

The acquisition process is dedicated to the preparation of the essential elements to extract the relational structure by applying a set of treatments. This process presents an iterative and individual processing applied to each table in the database along with its rows to acquire the constituent elements of the database, namely the tables, primary keys, foreign key records, and columns. This is achieved by:

(1) Creating a collection called BD to store table names and their related information. We used the method : (Map<String, Table>BD = new LinkedHashMap<>()) to define this collection.
(2) Retrieving the table name and creating an instance of the corresponding "Table" object. We used "resultSet.getString ("TABLE_NAME")" for this task.
(3) Retrieving table columns using the “metaData.getColumns()” method. Then, for each column, corresponding "Column" objects are created.
(4) Retrieving the primary keys of the table using metaData.getPrimaryKeys(). Then, corresponding "PrimaryKey" objects are created.
(5) Retrieving foreign keys using the metaData.getImportedKeys() method. Then, corresponding "ForeignKey" objects are created.
(6) Adding the Table to the collection (MAP) BD, where the key represents the name of the table processed, and the value presents the table containing the different objects created.

✓ **The Analysis Process :**

In this process, and following the same approach used in the acquisition process, each table is processed individually by applying an iteration on the rows. The objective, at this stage, is to calculate a set of measures allowing for the development of the multidimensional PIM model. These measures and their definition are represented as follows:

(1) Retrieving the number of rows from the table. The SQL query “SELECT COUNT(*)” is executed to obtain this result. Thus, the result of this query is represented by the V1 measure used in the rules for developing the PIM model.
(2) Calculating the insertion rate in the table. To obtain this result, we will combine two SQL queries, the first of which RQ1 returns the number of lines newly inserted into the table “SELECT COUNT (*), MAX(Id)”, and the second RQ2 presents the number of lines of the table. The insertion rate is calculated by dividing RQ1 by RQ2. The result of this operation presents the V2 measure used in the development of the PIM model.
(3) Updating the relevant collection based on the foreign keys encountered. This collection is used to count the number of foreign keys pointing to each table, which will be represented by the V3 measure also used in the development of the PIM model.
(4) Identifying the numeric attributes of the table and returning them as a sum in the V4 measure which will be used in the development of the PIM model.

4.1.3.2 Representation of extracted elements according to a relational model

Once the elements of the relational database are identified and the measures are calculated, the next step is to represent these elements according to a relational PSM model. To do this, we chose the XMI format to represent the relational structure according to the CWM relational metamodel presented in the previous section. The choice of the XMI standard is justified by the fact that it provides a standard structure for representing the elements of a model, such as classes, associations, properties, and constraints. More formally, the representation of the relational elements acquired in the previous level in XMI format following the CWM metamodel is carried out in three actions :

(1) Action 1: Initialization of the constructor with the data structure (Map <String, Table> tables) designed at the previous level. This collection contains the names of the relational database tables, each associated with its elements.

(2) Action 2: Conversion to XMI file, involving the use of the JAXB library to convert each table to an XMI file. This action is defined by a method taking as input the path of the file where it will be saved.

(3) Action 3: Conversion to XMI character string for all tables to represent the content in the XMI file.

After extracting and representing our physical relational models, the next step will be to integrate and merge them into the data lake. This process will be thoroughly explained in the next section

4.2. Second Step: RelationalModelsMerge Transformation

In this section, we present the second RelationalModelsMerge transformation of the ToExtractMD process, based on merging relational models extracted from data sources and consolidated into the data lake. To do this, we will present the input, output, and associated transformation rules.

4.2.1 Input: Ontology and relational models

The input of this operation is the set of relational models generated in the DataSource2RelationalModel step. These models are characterized by a relational structure consisting of tables, columns, foreign keys, and primary keys, according to the CWM relational metamodel presented in Section 4.1.2 (Figure 3).

Additionally, given the objective of this step RelationalModelsMerge and to set up a semantic layer to solve the semantic problems in the database consolidated in the data lake, a domain ontology is also used as input in combination with relational models. The domain ontology we used is specific to the higher education sector. More specifically, it is dedicated to course and student management within academic institutions. Ontology is designed according to the "METHONTOLOGY" approach [22], and implemented manually.

METHONTOLOGY is an ontology development method that allows to represent a domain in a formal and structured way. This method is designed to guide designers through the stages of the development process, from the specification of requirements to the creation of the ontological structure and documentation [22]. We designed our OntoDL ontology following 4 major phases of this approach, namely, Preparation, Specification, Conceptualization, Validation, and Implementation. After executing the first three phases of this approach, we obtained the result presented in table 3 describing the concepts, properties, and relationships of our OntoDL ontology.

Table 3 : concepts, properties and relationships for the OntoDL ontology

Concept	Properties	Relationships
Student	Student number, ID, Family name, First name, Email, Date of birth, Baccalaureate : stream,grade obtained or mention and year, ,address.	A student takes one or many exams
		A student has registered
		A student is supervised by one or more professors
		A student is enrolled in one or more modules/subjects
		A student takes one or more courses
		A student is enrolled in a discipline
Professor	Number, ID, Family name, First name, Email, Status, phone number, birth date.	A professor teaches one or many subjects
		A professor supervises one or many students
		A professor designs one or many exams
		A teacher is assigned a specialty or discipline
Subject	Course number, name, hours, description	A course is taught by one or many professors

		A course belongs to one discipline or degree
		A course is taken by one or more students
Module	Module number, name, hours, description	A module consists of one or more subjects/courses
		A module is assigned to a course
Evaluation	Exam number , Name , Grade	Une évaluation est rédigé par un enseignant
		An exam is related to a grade
		An exam is taken by one student
		An exam is related to one course
Registratio n	Number, date of registration, semester	A registration is made by a student
		Registration is valid for one academic year
Course	Course number, course name, Description, level , Hours, Room , learning objectives	A course is attended by one or more students
		A course is taught by one or more instructors
		A course is evaluated by one or more assessments
Results	Results Code, grades, average	A result is calculated from an exam
		A result is obtained by a student
Discipline	Discipline Code, Name	A discipline is linked to a degree
		A discipline includes modules
Degree	Degree Code, Name	A degree has a particular specialty or discipline
Academic year	Year Code, year, Month, Day	One academic year includes one registration

For the last implementation phase, so that our ontology is concretely manipulated with the formal language OWL, we chose the protected tool for the implementation. Figure 4 shows the OntoDL ontology on the protégé tool.

4.2.2 Output: A Relational Unified Model for the Data Lake

The output of this transformation corresponds to a unified relational model that presents the notion of «data lake». Its structure therefore follows a relational organization, which is in agreement with the CWM metamodel . This is because this model is created by merging the physical relational models generated by executing the DataSource2RelationalModel transformation.

4.2.3 Merging rules

In this section, we present the merging rules adopted to unify relational models of data sources into a unified consolidated model in the data lake. Specifically, this transformation aims to use the ontology we designed OntoDL to detect equivalent elements in PSM models, eliminating semantic conflicts and reducing data redundancy, and end with the unified relational model for the data lake in XMI file format.

Thus, our merging approach is composed of 4 major steps, namely: Loading, Reasoning, comparison and deployment.

- ✓ **Data Loading Phase:** From each data source, represented by its physical relational model, we started extracting the tables and columns, then stored them in separate lists, listDB1 and listDB2. Then, OntoDL is loaded from the OWL file using the OWLAPI library. This allows it to be stored in memory according to a data structure, making it easier to access and manipulate concepts, classes, properties and relationships efficiently.
- ✓ **Equivalent Classes Identification Phase (Reasoning):** In this step, we will identify the similar or equivalent classes in the first PSM relational model and ontology. More concretely, using the OWL API reasoner, we will browse the tables extracted for the first list (corresponds to the first relational model = listDB1) loaded in the previous step. Thus, for each table, we will perform reasoning on the ontology to identify equivalent classes in the ontology that match the semantics of the table. Then, equivalent classes are obtained using the OWLAPI reasoner using the subclass, superclass, and equivalent class search operations. Finally, the names of the equivalent classes are retrieved from their full URI and stored in a "HashMap" type variable.

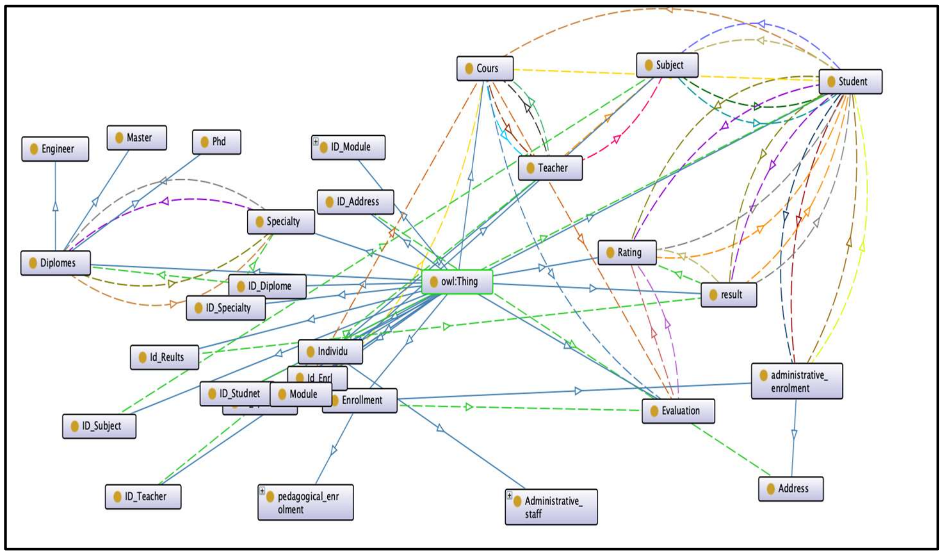


Figure 4 : OntoDL designed with protégé tool

- ✓ **Comparison and Storage of Correspondence Phase:** Once the names of the tables are retrieved and stored, we compare them with the tables extracted from the second list (corresponds to the second relational model = listDB2) loaded from the second relational model in the first step. Using the names of the equivalent classes detected and stored previously, we compare each table in listBD1 with the tables in listDB2. Once a similarity is detected, a match between the two models is identified. Thus, this match is recorded in a "HashMap" type structure under the name "HashMapTAB", where the keys are the table names of the listDB1 and the values are sets of matching table names from the DB2 list. Finally, for each table stored in "HashMapTAB", we proceed to detect identical columns, following the same principle used for detecting similar tables.
- ✓ **Unified XMI Model Deployment:** After all matches were collected in HashMapTAB, we organize them according to the XMI formalism. To do this, we started to define the structure of the XMI file ,and since working with relational models, the structure will also follow a relational organization, that is, a relational model according to the CWM metamodel used in the first step of the ExtrctTo process. Then, using the elements of the «HashMapTAB», we will follow the correspondences that we have defined, which
- ✓ implies the creation of the elements of the XMI models, described by the tables in the first place, followed by their column. Finally, we added non-equivalent tables in the XMI model. These tables are detected by a comparison of each ListeDB1 and ListeDB2 list with the structure defined in the XMI model to verify whether there is a match or not. If a match is detected, it means it is equivalent to an XMI model table. Otherwise, the table is declared not equivalent. In other words, the non-equivalent tables are those that have no match in the «HashMapTAB» structure. The columns of these tables are identified and mapped in the XMI model according to its attributes.

After presenting the process used to design the unified relational model for the data lake, in the next section, we will define the DataLake2MultdimentionalModel transformation to extract the PIM multidimensional conceptual model of the data lake.

4.3. Third Step: DataLake2-MultdimentionalModel transformation

In this section, we introduce DataLake2MultdimentionalModel, the third and last transformation of the ExtractMD process. The multidimensional conceptual model will be derived from the data lake. Thus, the input to this stage is the unifying PSM relational model, while the output is presented by the PIM multidimensional conceptual model. In what follows, we will describe these two models in detail, including the transformation rules to move from one model to the other.

4.3.1 Output: The multidimensional conceptual model

The output of the DataLake2MultdimentionalModel transformation is a conceptual model of a data warehouse. This conceptual level is typically used to define a technology-independent description of the multidimensional data warehouse. Although there is no standardized model for designing a data warehouse [23] constellation models are commonly recognized to represent multidimensional databases [24]. In this sense, we will use a constellation model at the PIM conceptual level, from which the data is structured according to a set of facts corresponding to the subjects of analysis, and a set of hierarchical dimensions constituting the axes of analysis. In what follows, we will formalize the concepts in our conceptual model. Figure 5 presents the PIM conceptual metamodel used.

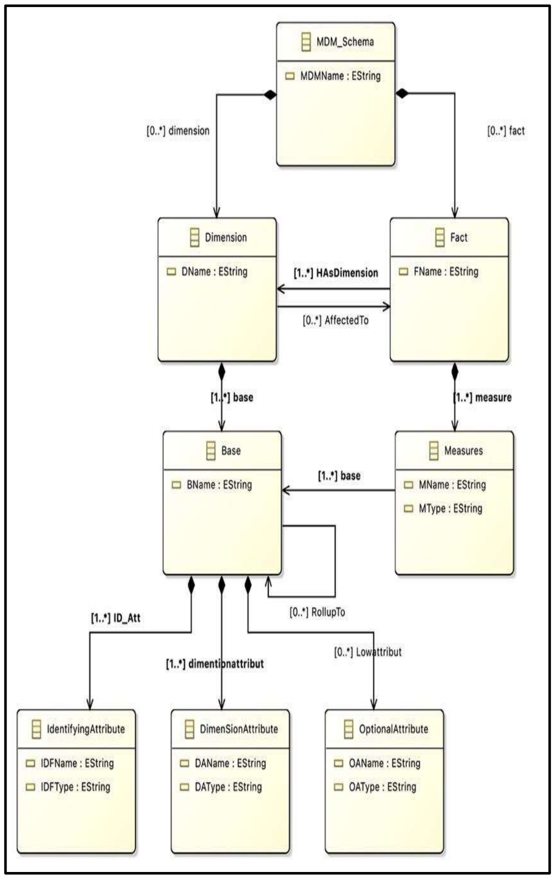


Figure 5: PIM Conceptual Metamodel

4.3.1.1 DataLake2MultdimentionalModel transformation rules

After obtaining the relational model of the data sources and the set of corresponding measures, a conceptual representation of the Multidimensional Model must be derived from the PIM model. This derivation process consists of two operations: marking the PSM model and deriving the multidimensional conceptual model from the marked PSM.

✓ **Physical model marked with multidimensional elements:**

Model marking is a technique used to reduce the complexity of transformations between models. In this step, our process marks each element of the physical model previously obtained by the appropriate concept of the multidimensional model, in accordance with the conceptual metamodel PIM described in section 4.3.1 (Figure 5). The marking is carried out by adding a prefix to the names of the elements of the PIM model.

Table 4 describes the correspondence between the marks and elements of the PSM relational model. This operation is ensured automatically by our process based on the measurements obtained in the process of extracting relational elements presented in section 4.1.3.1, as well as complementary rules ensuring the detection of multidimensional elements, namely, identification of facts, identification of dimensions, identification of measures, detection of hierarchies (represented by bases containing dimension attributes).

Table 4 : The marks used in the Relational model PSM

Prefix	Relational Elements	Multidimensional element
Fact_	Table	Fact
Dim_	Table	Dimension
Base_	Table	Base
Measure_	Column	Measure
ID_	PrimaryKey	IdentifyingAttributes
DA_	Column	DimensionAttributes

Identification of facts. A fact is a table that represents the logical relationship between several business concepts, including their numerical measures intended to support the decision-making process. A fact and its measures can be detected from a data source according to the variables calculated in the previous section. Thus, a table is marked as done in the physical relational model if and only if all of the following conditions are satisfied :

- A fact table is the largest table among all the tables in the data source. This condition is verified with the variable (measures) V1.
- A fact table is the most updated table among the others because it reflects the dynamics of the commercial process (or business process). This condition is checked with the variable V2.
- A fact table is the table with the largest number of foreign keys. This condition is checked with the variable V3.
- A fact table is the table with the largest number of numeric columns. This condition is checked with the variable V4.

To reinforce the aforementioned conditions, we proposed other complementary rules to verify the detected facts. They are described as follows :

- A fact table cannot have empty values.
- Numeric columns in a fact table must be foreign keys representing the primary keys of other tables. This condition can reinforce the result of the V3 variable because in certain cases, we can have a table satisfying the 3 conditions, the result of the last of which does not cover any business aspect relating to the decision-making process.

Identification of measures. A fact table is identified by two types of columns, columns that correspond to foreign keys or columns that indicate measures. Measures are analysis values applied by the decision process. A column is marked as a measure if it is numeric and not a foreign key.

Identification of dimensions. The dimension represents the axis of analysis of the facts. It thus determines the relevant measures for the decision-making process. A table detected and marked as dimension, if its primary key is a foreign key in a table marked as done.

Identification of bases. Any tables that have not been marked as facts or dimensions are marked as bases.

Identification of dimension attributes. The columns of each table marked as a Dimension or as a Base can be considered either dimension identifiers or dimension attributes. A column is marked as IdentifyingAttributes if it is a primary key. Otherwise, it is considered a DimensionAttributes. It is important to note that the OptionalAttributes attribute is manually set at the end of the transformation, so the user has the option to modify any of the DimensionAttributes with this attribute ; this is because there is no way to know if a column can be considered, at transformation time, as a dimension attribute or just an additional attribute for a given column.

✓ **Deployment of the PIM Conceptual Model:**

Once the PSM physical model marked with the multidimensional elements is obtained, we can continue to execute our process to achieve the multidimensional conceptual structure.

- **Acquisition of facts and dimensions.** From the marked PSM relational model, we can easily deduce our elements, with each table marked as 'Fact_' translated to 'Fact', as well as for dimensions that are inferred from tables marked with the 'Dim_' prefix. By applying the same principle, we can obtain both the measures and the attributes of the dimensions.
- **Acquisition of hierarchies.** A hierarchy organizes attributes based on the granularity it presents. In our model, the BASE provides the finest levels of dimension granularity. However, a dimension is attached directly to a single hierarchy, which involves creating a BASE bearing the name of the dimension while remaining attached to it when translated to the PIM. On the other hand, from an identified dimension, we can face situations where hierarchies are translated according to the type of aggregation. This is detected using the foreign key existing in the tables marked as BASE, which is translated in the PIM model by the "rollup" relationship connecting the two BASE .

5. DESCRIPTION OF THE DS2MD PROTOTYPE : EXTRACTION OF A MULTIDIMENSIONAL MODEL FROM A DATA LAKE CONSOLIDATING DATA SOURCES

This section aims to present a prototype that we designed to test our contributions in the field of data warehousing. This prototype called DS2MD offers all the mechanisms to consolidate data sources in a data lake and to extract a multidimensional conceptual model from it.

In what follows we will present the architecture of our prototype in section 5.1, followed by the implementation of ToExtractMD processes in section 5.2, then, an experiment to test the reliability of our proposal is shown in section 5.3.

5.1 The architecture of our prototype

The main objective of our system is to extract the multidimensional conceptual model from a relational data lake consolidating the data sources constituting an organization's information system. This system brings together two main components : the user interface, and the ToExtractMD module. The tool allows the user to extract the physical relational model from each operational database of an information system, then integrate them into the data lake in order to have a unified model for the latter. Finally, the system transforms this model into a multidimensional model. Our tool offers each transformation of model outputs presented with the XMI format. Figure 6 illustrates the architecture of our system. The implementation of the ToextractMD module will be detailed in the following sections.

5.2 The implementation of ToExtractMD Module

The purpose of this module is to extract the multidimensional model from a set of relational databases centralized in a data lake. The input is a relational database set, while the output is a multidimensional conceptual model. In addition, ToExtractMD includes a data integration phase in the data lake, allowing the merging of databases ingested into the data lake. Figure 7 shows the ToExtractTM module with three consecutive sub-modules.

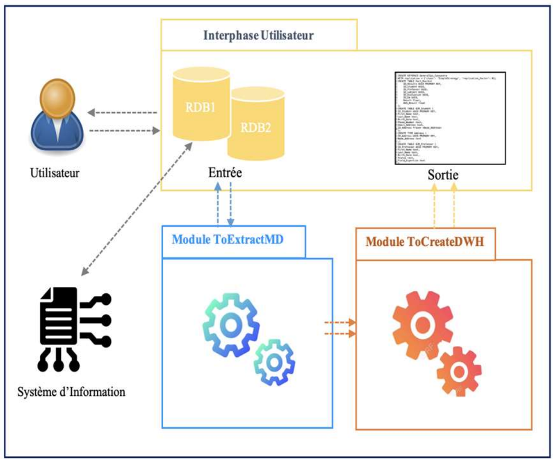


Figure 6: Architecture of DS2ML Prototype.

The first submodule is DataSource2RelationalModel presenting the input to the ToExtractMD module. Its role consists of extracting the physical relational model PSM from the relational database. The transformation executed by this submodule presented in section 4.1 is described in the JAVA language where Figure 8 presents an extract of this code.

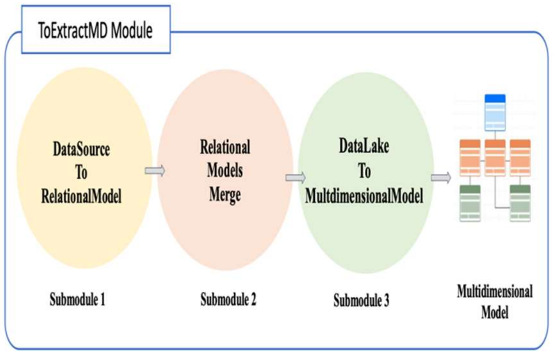


Figure 7 : ToExtractMD module

The second submodule is RelationalModelsMerge, which allows for merging the relational models obtained previously into a unified relational model for the data lake. Figure 9 presents an extract of the transformations presented in the section 4.2 is described in the JAVA language.

The last submodule is DataLake2MultdimentionalModel. It represents the output of the ToExtractMD module, which relies on extracting the multidimensional model from the unified relational model. Thus, the figure 10 presents the JAVA code describing the transformations executed by this submodule presented in subsection 4.3. All of these transformations allow us to describe the models under the XMI format.

```
………………………
int k = 0;//incrementation
Map<String,Table> tables = new LinkedHashMap<>();
Connection con = DriverManager.getConnection(url,login,password);
DatabaseMetaData metaData = con.getMetaData();
Map<String,Integer> inDegrees = new HashMap<>();
ResultSet resultSet = metaData.getTables(database, null, null, new
String[]{"TABLE"});
while(resultSet.next()) {
String tableName = resultSet.getString("TABLE_NAME");
Table table = new Table(tableName);
Statement st =
con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSe
t.CONCUR_UPDATABLE);
ResultSet rs = st.executeQuery("SELECT COUNT(*) FROM " +
database + "."+tableName);
  rs.next();
  int rowCount = rs.getInt(1);
  table.setLineCount(rowCount);
  ResultSet columns = metaData.getColumns(database, null,
tableName,null);
  while(columns.next()){
  String columnName = columns.getString("COLUMN_NAME");
  String columnType = columns.getString("TYPE_NAME");
  Column column = new Column(columnName,columnType);
  table.addColumn(column);}
  ResultSet primaryKeys = metaData.getPrimaryKeys(database, null,
tableName);
  while(primaryKeys.next()){
…………………………………………………
```

Figure 8 : Part of Java script used to extract relational Model PSM from the data dictionary

```
import org.apache.jena.ontology.OntModel;
import org.apache.jena.query.*;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.util.FileManager;
import javax.xml.xpath.*;
…………………………………
   public static NodeList fetch(String file,String xpathCustom) throws
Exception { XPath xpath = XPathFactory.newInstance().newXPath();
      InputSource xml = new
InputSource("src/main/resources/"+file+".xmi");
      String result = (String) xpath.evaluate(xpathCustom, xml,
XPathConstants.STRING);
      XPathExpression expr = xpath.compile(xpathCustom);
      return (NodeList)expr.evaluate(xml, XPathConstants.NODESET);}
private static void fnc2() throws Exception {
      NodeList tablesDb1=fetch("bd_estudiantine","//tables");
      NodeList tablesDb2=fetch("sysensaf","//tables");
      OWLOntologyManager manager =
OWLManager.createOWLOntologyManager();
      OWLOntology ontology =
manager.loadOntologyFromOntologyDocument(new
File("src/main/resources/Student_Activities_1.owl"));
      HashMap<String, HashSet<String>> tabcleHash=new HashMap<>();
      for (int i = 0; i < tablesDb1.getLength(); i++) { String node1 =
tablesDb1.item(i).getAttributes().getNamedItem("TName").getTextContent
();
         node1 = node1.substring(0, 1).toUpperCase() + node1.substring(1);
         tabcleHash.put(node1,new HashSet<>()); IRI iri =
IRI.create("http://www.semanticweb.org/lamyaoukhouya/ontologies/2023/
6/Learning#"+node1);
OWLClass style = manager.getOWLDataFactory() .getOWLClass(iri);
OWLReasonerFactory reasonerFactory = new
StructuralReasonerFactory(); OWLReasoner reasoner =
reasonerFactory.createReasoner(ontology;
NodeSet<OWLClass> subClasses = reasoner. getSubClasses(style, true);
NodeSet<OWLClass> superClasses = reasoner.getSuperClasses(style,
true);Node<OWLClass> equivClasses =
reasoner.getEquivalentClasses(style);
Set<OWLClass> classes = subClasses.getFlattened();
classes.addAll(superClasses.getFlattened());
classes.addAll(equivClasses.getEntities());
Set<String>names=classes.stream().map(Object::toString).collect(Collecto
rs.toSet());
names=names.stream().map(w->w.substring(w.indexOf("#") + 1,
w.length() -1)).collect(Collectors.toSet());
         System.out.println("-------- "+node1+" --------");
         System.out.print("OWL equivalents :  "); names.forEach(w-
>System.out.print(w +" "));
         System.out.println();
         System.out.println(names);
         System.out.print("Compare to :  ");
         for (int j = 0; j < tablesDb2.getLength(); j++) { String node2 =
tablesDb2.item(j).getAttributes().getNamedItem("TName").getTextContent
();
<http://www.semanticweb.org/lamyaoukhouya/ontologies/2023/6/Learning
#%s> . " +,"?propertyrdfs:range}
   private static void fnc3(HashMap<String,HashSet<String>> tabSet){
……………………
```

Figure 9 : Part of the Java script to build the unified relational model PSM for the data lake

```
import java.io.StringWriter;
import java.sql.SQLException;
import java.util.*;
public class ModelMultidimensionalMarque
{ModelRelationnelMarque model;
private List<Dimension> dimensionList;
private List<Fact> factList;
publicModelMultidimensionalMarque(ModelRelationnelMarque
model) { this.model = model;
dimensionList = new ArrayList<>();
factList = new ArrayList<>();
int i = 0;
for (Table fac : model.getFactTables()) {Fact fact = new Fact();
fact.setName(fac.getName().split("FACT_")[1]);fact.setIndice(i);
for (Column column : fac.getColumns()){
if (column.getName().startsWith("MEASURE_")){
Measures measures = new Measures();
measures.setName(column.getName().split("MEASURE_")[1]);
measures.setType(column.getType());
```

```
fact.getMeasuresList().add(measures);}}
factList.add(fact);
i++;} i = 0;
for (Table dim : model.getDimensionTables()){
Dimension dimension = new Dimension();
dimension.setName(dim.getName().split("DIM_")[1]);
dimension.setIndice(i); Base basedim = new Base();
basedim.setIndice(0);
basedim.setName(dim.getName().split("DIM_")[1]);
for (Column column : dim.getColumns()){
Dimentionattribut dimentionattribut = new Dimentionattribut
(column.getName(), column.getType());
basedim.getDimentionattributs().add(dimentionattribut);}
for (PrimaryKey primaryKey : dim.getPrimaryKeys()){
ID_att id_att = new ID_att(primaryKey.getName(),
primaryKey.getType());
basedim.getId_atts().add(id_att);}
dimension.getBases().add(basedim);
this.getBases_1(dimension, dim, 1, 0);
this.getBases_2(dimension, dim);
fact.setHasDemension(fact.getHasDemension() + "//
for (Base base : dimension.getBases()){
measures.setBase(measures.getBase() + "//@dimension." +
dimension.getIndice() + "/@base." + base.getIndice());}}}}}}
k++;}
dimension.setAffectedTo(affected.toString());
dimensionList.add(dimension);
i++;}
…..
```

Figure 10 : part of JAVA script for marking and transforming a relational model PSM into a multidimensional conceptual model PIM

5.3 Validation and experimentation of the ToExtractMD module:

The objective of this section is to show how to use the DSToMD system to automate the extraction of the multidimensional model from a data lake consolidating the operational sources of an organization. More precisely, this section presents an experiment on the transformations carried out by the ToExtractMD process through a case study of two computer applications forming part of the information system of one of the institutions of Ibn Zohr University. We will begin with the presentation of these two applications, followed by the validation of our proposals by comparing the result of our automatic prototype with a manual process carried out by an experienced computer scientist. This validation will be carried out on the execution time to recover the PIM conceptual model, as well as for the authenticity of the two results of the automatic and manual process compared to the models obtained at the level of each transformation of the ToExtractMD process.

5.3.1 Presentation of Ibn Zohr University Information System Applications:

- ✓ **Application 1 : ERP student affairs management:**

The first application is an ERP dedicated to the organization and management of students and courses. This ERP is centralized at the information system of Ibn Zohr University, managing through a single database, all students from 22 institutions, which implies a volume of data reaching several terabytes. Furthermore, the major objective of this application is to offer institutions complete management of student registrations and records, from registration to the getting of their diplomas.

- ✓ **Application 2 : Learning management system LMS**

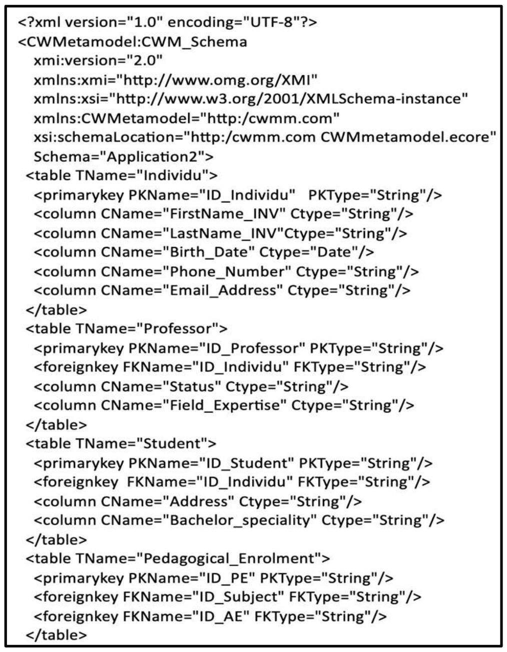
Concerning the second application of the information system, we chose a LMS platform to facilitate interaction with students. This tool also uses a single relational database for the centralization of data, mainly a relational database per institution, from which we chose only one to carry out this experiment. The features characterizing this LMS include managing student registrations and admission, curriculum management, and evaluation management.

5.3.2 Automatic Process

The two applications we have just presented have different objectives. The first one manages the curriculum of the student from his first registration in the university until the delivery of his diploma, while the second creates a centralized space for interaction between teachers, students and administration, and manages the academic monitoring of the student.

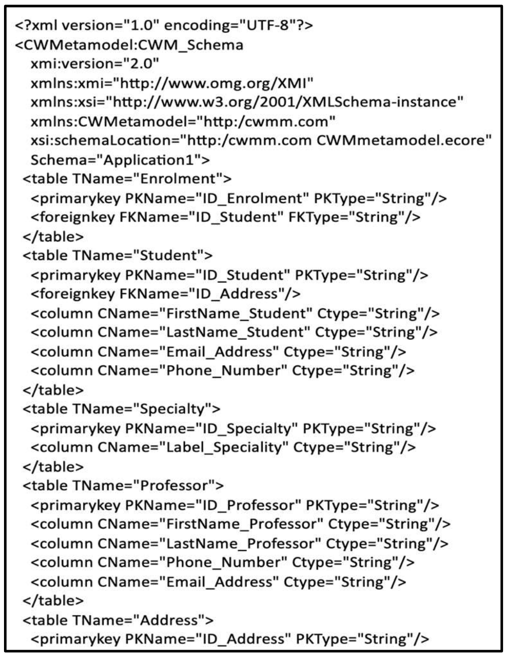
Despite this difference in purpose between the two applications, we can see that the features are similar on the one hand, and complementary on the other. Therefore, we used our tool on both applications to validate our approach.We started by entering the URIs of the two relational databases. Thus, by clicking on the «DS2RDM» button, we were able to extract the relational physical structure for each database presented in XMI format. This result describes the first step DataSource2RelationalToExtract module, which provides an output physical PSM model for each data source in XMI format. This follows the CWM relational metamodel also presented in the same chapter. Figure 11((A) and 11(B)) shows extracts for the two PSM physical models of the two XMI format databases generated by our system.

After that, we executed the second RelationalModelsMerge step of the same module, by clicking on the "RDM2Merge" button, which merges the two PSMs and creates a unified relational model for the data lake.This operation displays all new merged tables, as well as the remaining tables in both databases. At this stage, the user intervenes to name the new tables created and to choose the tables intended to constitute the unified model. Once this model was designed, by clicking on the "DL2MD" button presenting the last step of this module, we proceeded to extract the multidimensional conceptual PIM model from the unified PSM relational model representing the data lake.



```
<?xml version="1.0" encoding="UTF-8"?>
<CWMetamodel:CWM_Schema
  xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:CWMetamodel="http:/cwmm.com"
  xsi:schemaLocation="http:/cwmm.com CWMmetamodel.ecore"
  Schema="Application2">
 <table TName="Individu">
  <primarykey PKName="ID_Individu"   PKType="String"/>
  <column CName="FirstName_INV" Ctype="String"/>
  <column CName="LastName_INV"Ctype="String"/>
  <column CName="Birth_Date" Ctype="Date"/>
  <column CName="Phone_Number" Ctype="String"/>
  <column CName="Email_Address" Ctype="String"/>
 </table>
 <table TName="Professor">
  <primarykey PKName="ID_Professor" PKType="String"/>
  <foreignkey FKName="ID_Individu" FKType="String"/>
  <column CName="Status" Ctype="String"/>
  <column CName="Field_Expertise" Ctype="String"/>
 </table>
 <table TName="Student">
  <primarykey PKName="ID_Student" PKType="String"/>
  <foreignkey  FKName="ID_Individu" FKType="String"/>
  <column CName="Address" Ctype="String"/>
  <column CName="Bachelor_speciality" Ctype="String"/>
 </table>
 <table TName="Pedagogical_Enrolment">
  <primarykey PKName="ID_PE" PKType="String"/>
  <foreignkey FKName="ID_Subject" FKType="String"/>
  <foreignkey FKName="ID_AE" FKType="String"/>
 </table>
```

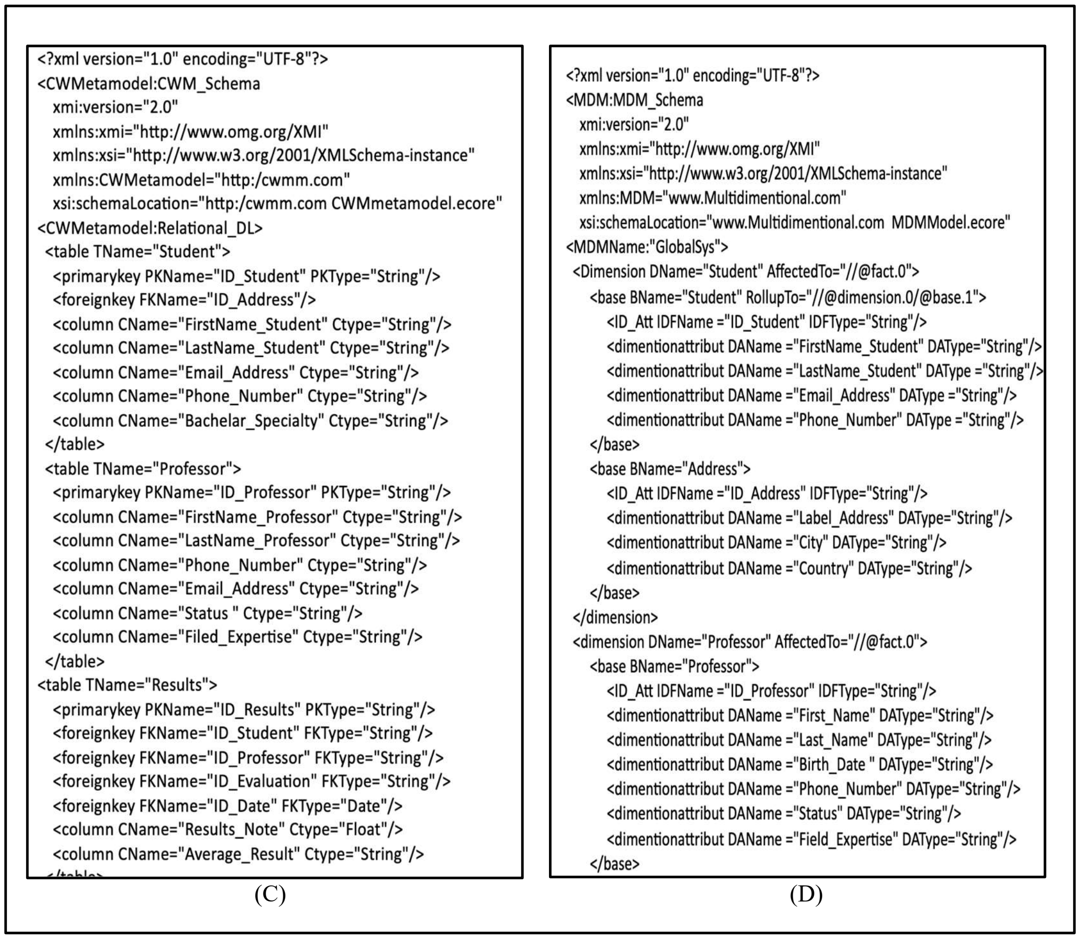
(A)



```
<?xml version="1.0" encoding="UTF-8"?>
<CWMetamodel:CWM_Schema
  xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:CWMetamodel="http:/cwmm.com"
  xsi:schemaLocation="http:/cwmm.com CWMmetamodel.ecore"
  Schema="Application1">
 <table TName="Enrolment">
  <primarykey PKName="ID_Enrolment" PKType="String"/>
  <foreignkey FKName="ID_Student" FKType="String"/>
 </table>
 <table TName="Student">
  <primarykey PKName="ID_Student" PKType="String"/>
  <foreignkey FKName="ID_Address"/>
  <column CName="FirstName_Student" Ctype="String"/>
  <column CName="LastName_Student" Ctype="String"/>
  <column CName="Email_Address" Ctype="String"/>
  <column CName="Phone_Number" Ctype="String"/>
 </table>
 <table TName="Specialty">
  <primarykey PKName="ID_Specialty" PKType="String"/>
  <column CName="Label_Speciality" Ctype="String"/>
 </table>
 <table TName="Professor">
  <primarykey PKName="ID_Professor" PKType="String"/>
  <column CName="FirstName_Professor" Ctype="String"/>
  <column CName="LastName_Professor" Ctype="String"/>
  <column CName="Phone_Number" Ctype="String"/>
  <column CName="Email_Address" Ctype="String"/>
 </table>
 <table TName="Address">
  <primarykey PKName="ID_Address" PKType="String"/>
```

(B)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



```
<?xml version="1.0" encoding="UTF-8"?>
<CWMetamodel:CWM_Schema
  xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:CWMetamodel="http:/cwmm.com"
  xsi:schemaLocation="http:/cwmm.com CWMmetamodel.ecore"
<CWMetamodel:Relational_DL>
 <table TName="Student">
  <primarykey PKName="ID_Student" PKType="String"/>
  <foreignkey FKName="ID_Address"/>
  <column CName="FirstName_Student" Ctype="String"/>
  <column CName="LastName_Student" Ctype="String"/>
  <column CName="Email_Address" Ctype="String"/>
  <column CName="Phone_Number" Ctype="String"/>
  <column CName="Bachelar_Specialty" Ctype="String"/>
 </table>
 <table TName="Professor">
  <primarykey PKName="ID_Professor" PKType="String"/>
  <column CName="FirstName_Professor" Ctype="String"/>
  <column CName="LastName_Professor" Ctype="String"/>
  <column CName="Phone_Number" Ctype="String"/>
  <column CName="Email_Address" Ctype="String"/>
  <column CName="Status " Ctype="String"/>
  <column CName="Filed_Expertise" Ctype="String"/>
 </table>
<table TName="Results">
  <primarykey PKName="ID_Results" PKType="String"/>
  <foreignkey FKName="ID_Student" FKType="String"/>
  <foreignkey FKName="ID_Professor" FKType="String"/>
  <foreignkey FKName="ID_Evaluation" FKType="String"/>
  <foreignkey FKName="ID_Date" FKType="Date"/>
  <column CName="Results_Note" Ctype="Float"/>
  <column CName="Average_Result" Ctype="String"/>
```

(C)

```
<?xml version="1.0" encoding="UTF-8"?>
<MDM:MDM_Schema
  xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:MDM="www.Multidimentional.com"
  xsi:schemaLocation="www.Multidimentional.com MDMModel.ecore"
<MDMName:"GlobalSys">
 <Dimension DName="Student" AffectedTo="//@fact.0">
   <base BName="Student" RollupTo="//@dimension.0/@base.1">
     <ID_Att IDFName ="ID_Student" IDFType="String"/>
     <dimentionattribut DAName ="FirstName_Student" DAType="String"/>
     <dimentionattribut DAName ="LastName_Student" DAType ="String"/>
     <dimentionattribut DAName ="Email_Address" DAType ="String"/>
     <dimentionattribut DAName ="Phone_Number" DAType ="String"/>
   </base>
   <base BName="Address">
     <ID_Att IDFName ="ID_Address" IDFType="String"/>
     <dimentionattribut DAName ="Label_Address" DAType="String"/>
     <dimentionattribut DAName ="City" DAType="String"/>
     <dimentionattribut DAName ="Country" DAType="String"/>
   </base>
 </dimension>
 <dimension DName="Professor" AffectedTo="//@fact.0">
   <base BName="Professor">
     <ID_Att IDFName ="ID_Professor" IDFType="String"/>
     <dimentionattribut DAName ="First_Name" DAType="String"/>
     <dimentionattribut DAName ="Last_Name" DAType="String"/>
     <dimentionattribut DAName ="Birth_Date " DAType="String"/>
     <dimentionattribut DAName ="Phone_Number" DAType="String"/>
     <dimentionattribut DAName ="Status" DAType="String"/>
     <dimentionattribut DAName ="Field_Expertise" DAType="String"/>
   </base>
```

(D)

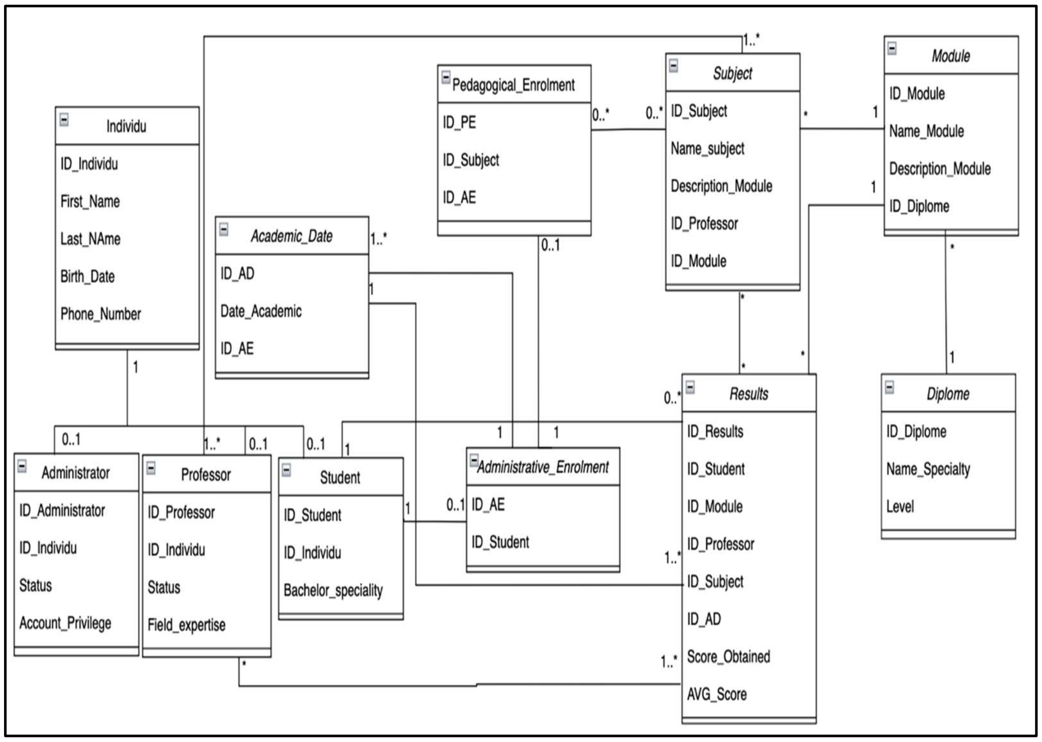
Figure 11 : (A)PSM physical models DB1 ; (B) PSM physical model DB2 ; (C) PSM model unifier ; (D) PIM multidimensional Conceptual Model

5.3.3 Manual Process

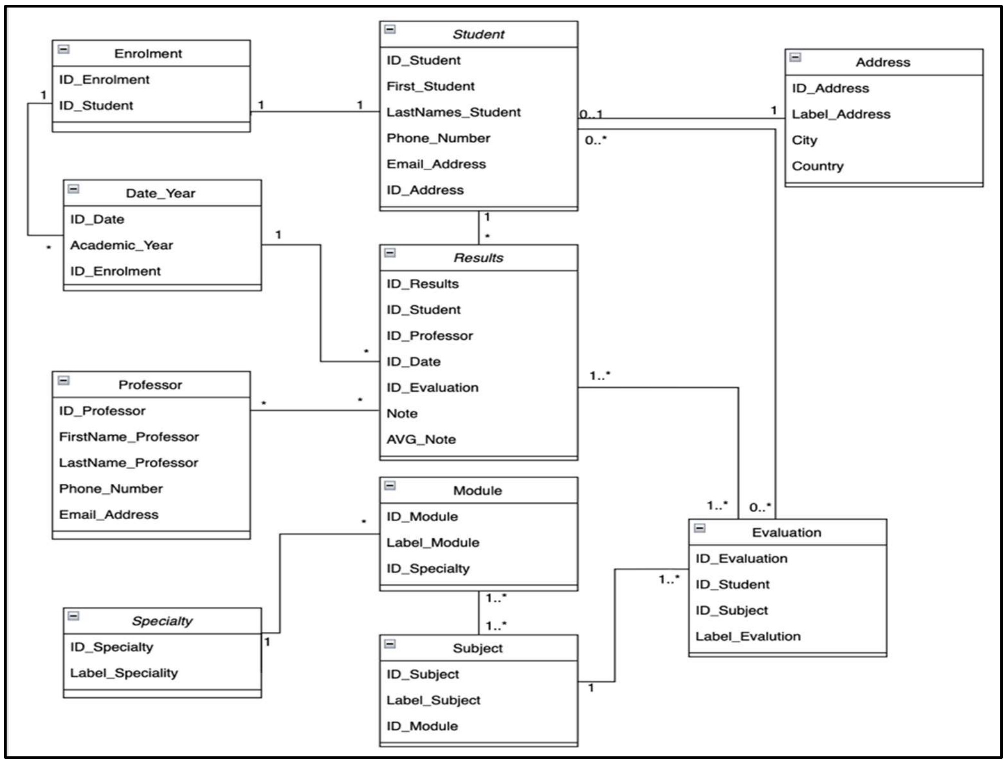
The manual process experience is designed by an IT professional who clearly knows how to implement and manipulate databases, as well as data warehouses. However, he has no idea about the structure of these two databases. This user profile was not chosen at random; we want to be credible and reliable when comparing the execution time of the two processes, knowing that our system has the ability to extract the relational structure from any relational data source. First, the computer scientist applied his process of extracting the relational model from each database. He began by analyzing the data, establishing the structure of the data, identifying entities and relationships between them, and determining the attributes associated with each entity. He then merged these models by comparing the table names to detect similarities.

Then, for each of the similar tables, he examined the columns to identify which were similar and which were not. On this basis, he decided which tables and columns to merge. Then, using the previous concepts, he designed a relational model for each database, identifying the main tables, determining their primary and foreign keys and their attributes.
Next, we assigned the same rules we used to extract the multidimensional elements to the computer scientist. This way, the user's business needs do not interfere with the design, which is based on the analysis of the data source.
Figure 12(A) and 12 (B) successively illustrates the two models designed by the computer scientist, as well as the final result presented by the multidimensional conceptual model.

ISSN: **1992-8645** www.jatit.org E-ISSN: **1817-3195**

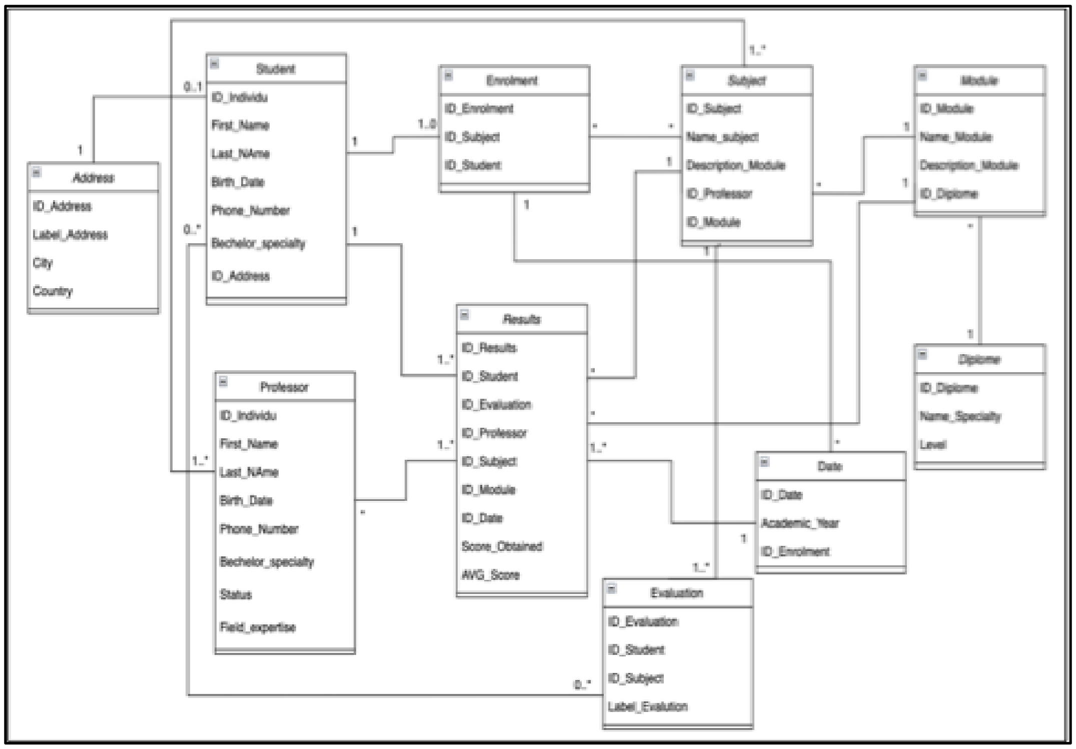


(A)

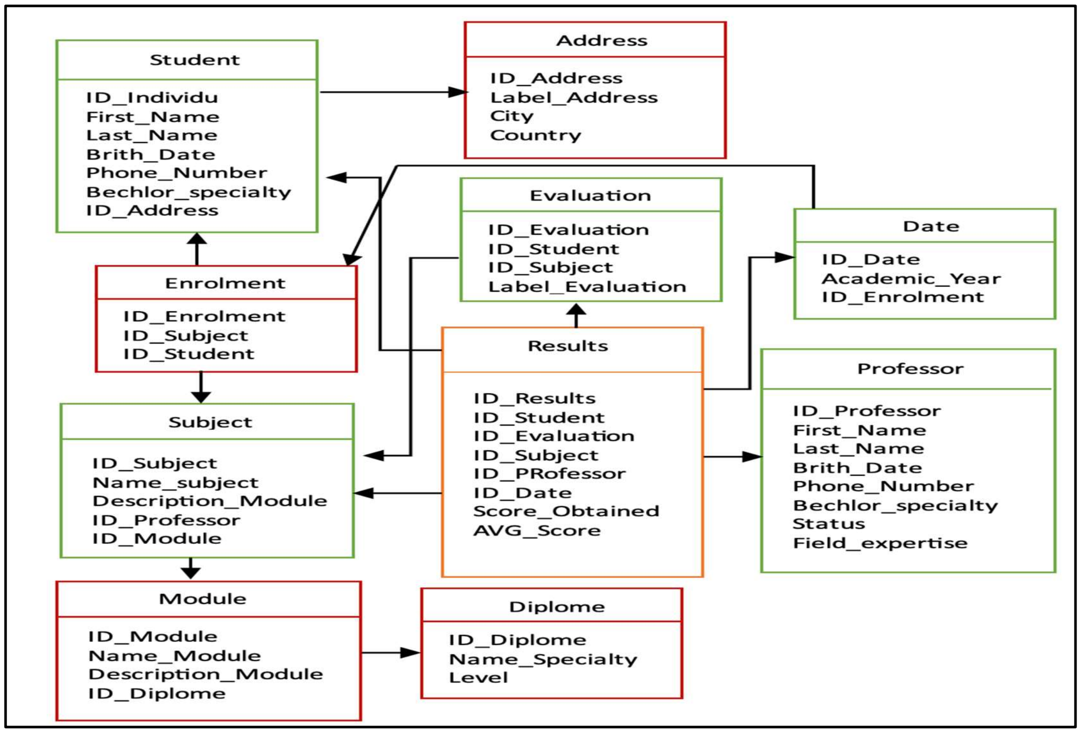


(B)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



(C)



(D)

Figure 12: (A) Relational Model for Application 1 ;(B) Relational Model for Application 2 ; (C) Relational Merge Model ; (D) Multidimensional model

5.3.4 Evaluation of the results

To compare the results obtained, we used the criteria relating to the quality of the result and the transformation execution time between the manual process and our automated process.

- ✓ **Model Quality:**

The goal is to determine the quality of physical data structures, both in the generated source code as well as the resulting guidelines. In other words, we aim to evaluate the adequacy of all transformations involved in the process of extracting relational models, their merge into unified models, and the transformation of these into multidimensional models for implementation on one of the physical models for one of the relational or NoSQL families. Regarding the process carried out by the computer scientist, the result he obtained manually was compared to all the phases of our automated process, namely, the relational model extracted from the data source, the unified model, the multidimensional model designed and the physical model obtained with its translation into CQL code. Both results are identical. We should also mention that the merged model is not identical to our model, as each of us assigned different names to the merger tables during the merge process.

- ✓ **The transformation time:**

Likewise, we compared the time required for our automated process with the time taken by the computer scientist to obtain an equivalent result manually.

For an almost identical result (relational model unified model + multidimensional model physical model with code), the time to complete the process was significantly reduced. In other words, model transformations performed with our process save significant time compared to the manual process, as shown in table 5.

Table 5 : Execution times of different phases for manual and automatic process

Process	The Phases of Transformation			
	(1)	(2)	(3)	Total Duration
Manual	8h	1h	30min	9h30min
Automatic	3min	1min	5min	9min

(1) Extraction of the relational model
(2) Model merge
(3) Extraction of the multidimensional model

We can note that the time saved between the manual and automated processes increases with the complexity of the relational database that constitutes the entry point of both processes.

6. Discussion and Future Work

This work falls within the framework of designing decision support systems by utilizing innovative concepts to modernize traditional approaches to data warehouse design and manage massive data. We have adopted the concept of a data lake for consolidating data from heterogeneous sources. To optimize this approach, we integrated a semantic layer on top of the data lake, which allows for the resolution of semantic conflicts and the development of a unified model. This unified model is then used to extract the conceptual model of the data warehouse.

This work represents a fundamental first step in the development of lakehouse architectures, which combine the concepts of data lakes and data warehouses. This combination addresses the limitations of each individual approach. As data management needs evolve, lakehouse architectures have emerged to leverage the benefits of data lakes while providing analytical and structured management capabilities similar to those of data warehouses. This fusion optimizes analytical performance while retaining the flexibility of the data lake.

For future work, we plan to migrate to a lakehouse architecture while integrating the concept of data mesh. This migration aims to combine the advantages of both approaches to create a more robust and adaptable solution. The data mesh introduces decentralized data management, where each team treats data as a product and takes responsibility for its quality and governance. By integrating data mesh with our lakehouse architecture, we will enhance the flexibility and scalability of our system, enabling a more distributed and collaborative management of data.

7. CONCLUSION :

In this paper, we presented an automated approach for extracting a multidimensional model from a data lake consolidating relational data sources. To do this, we proposed a process called ToExtractMD, consisting of three steps : DataSource2RelationalModel, RelationalModels-Merge, and DataLake2MultidimensionalModel.

The first step is to extract the PSM physical model from each database using the metadata stored in the data dictionary. The second step merges the previously extracted PSM models into a single physical model through the use of an ontology specific to the field of higher education, resulting in the PSM model of the data lake. Finally, the third

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

step allows us to extract the conceptual PIM model from the unified PSM model.

In our future endeavors, we plan to migrate to a lakehouse architecture while expanding the variety of data integrated into the data lake by incorporating heterogeneous data sources from the university information system. Additionally, we aim to integrate the data mesh concept to enable decentralized data management, treating data as a product with clear ownership and governance. To extract the conceptual model from this diverse data lake, we will establish a metadata management system to facilitate the detection of multidimensional structures. This approach will combine the strengths of the data lake, lakehouse, and data mesh to enhance our data management and analysis capabilities.

REFERENCES:

[1] INMON, William H. *Building the data warehouse*. John wiley & sons, 2005.

[2] ROMERO, Oscar et ABELLÓ, Alberto. A survey of multidimensional modeling methodologies. International Journal of Data Warehousing and Mining (IJDWM), 2009, vol. 5, no 2, p. 1-23

[3] OUKHOUYA, Lamya,EL HADADDI, Anass, ER-RAHA, Brahim, et al. A generic metadata management model for heterogeneous sources in a data warehouse. In : E3S Web of Conferences. EDP Sciences, 2021. p. 01069.

[4] NAMBIAR, Athira et MUNDRA, Divyansh. An overview of data warehouse and data lake in modern enterprise data management. Big data and cognitive computing, 2022, vol. 6, no 4, p. 132.

[5] OUKHOUYA, Lamya, EL HADDADI, Anass, ER-RAHA, Brahim, et al. A Proposed Big Data Architecture Using Data Lakes for Education Systems. In : International Conference on Networking, Intelligent Systems and Security. Cham : Springer International Publishing, 2022. p. 53-62

[6] OUKHOUYA, LAMYA, EL HADDADI, ANASS, ER-RAHA, BRAHIM, et al. Automating Data Warehouse Design With MDA Approach Using NoSQL and Relational Systems. Journal of Theoretical and Applied Information Technology, 2023, vol. 101, no 23

[7] SHAHROKNI, Ali et SÖDERBERG, Jan. Beyond Information Silos Challenges in Integrating Industrial Model-based Data. In : BigMDE@ STAF. 2015. p. 63-72.

[8] WIBOWO, Merlinda, SULAIMAN, Sarina, et SHAMSUDDIN, Siti Mariyam. Machine learning in data lake for combining data silos. In : Data Mining and Big Data: Second International Conference, DMBD 2017, Fukuoka, Japan, July 27–August 1, 2017, Proceedings 2. Springer International Publishing, 2017. p. 294-306.

[9] OUKHOUYA, Lamya, HADDADI, Anass El, ER-RAHA, Brahim, et al. Designing Hybrid Storage Architectures with RDBMS and NoSQL Systems: A Survey. In : International Conference on Advanced Intelligent Systems for Sustainable Development. Cham : Springer Nature Switzerland, 2022. p. 332-343.

[10] SCHWADE, Florian et SCHUBERT, Petra. A semantic data lake for harmonizing data from cross-platform digital workspaces using ontology-based data access. 2020.

[11] KACHAOUI, Jabrane et BELANGOUR, Abdessamad. From single architectural design to a reference conceptual meta-model: an intelligent data lake for new data insights. International Journal, 2020, vol. 8, no 4, p. 1460-1465.

[12] KACHAOUI, Jabrane, LARIOUI, Jihane, et BELANGOUR, Abdessamad. Towards an ontology proposal model in data lake for real-time COVID-19 cases prevention. 2020.

[13] DABBÈCHI, Hichem, HADDAR, Nahla Zaaboub, ELGHAZEL, Haytham, et al. Social media data integration: From data lake to nosql data warehouse. In : International Conference on Intelligent Systems Design and Applications. Cham : Springer International Publishing, 2020. p. 701-710.

[14] KATHIRAVELU, Pradeeban et SHARMA, Ashish. A dynamic data warehousing platform for creating and accessing biomedical data lakes. In : Second International Workshop, DMAH 2016, Held at VLDB 2016, New Delhi, India, September 9, 2016, Revised Selected Papers 2. Springer International Publishing, 2017. p. 101-120.

[15] BLANC, Xavier et SALVATORI, Olivier. MDA en action: Ingénierie logicielle guidée par les modèles. Editions Eyrolles, 2011.

[16] MAZÓN, Jose-Norberto et TRUJILLO, Juan. An MDA approach for the development of data warehouses. Decision Support Systems, 2008, vol. 45, no 1, p. 41-58.

[17] MIAO, Gao, HONGXING, Liu, SONGYU, Xie, et al. Research on user interface transformation method based on MDA. In

: 2017 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES). IEEE, 2017. p. 150-153.

[18] JECKLE, Mario. OMG's XML Metadata Interchange Format XMI. XML4BPM 2004, 2004, p. 25.

[19] MEDINA, Enrique et TRUJILLO, Juan. Metamodel (CWM). In : Advances in Databases and Information Systems: 6th East European Conference, ADBIS 2002, Bratislava, Slovakia, September 8-11, 2002, Proceedings. Springer, 2003. p. 232

[20] MAZON, Jose-Norberto, TRUJILLO, Juan, SERRANO, Manuel, et al. Applying MDA to the development of data warehouses. In : Proceedings of the 8th ACM international workshop on Data warehousing and OLAP. 2005. p. 57-66.

[21] ROCHA, Leonardo, VALE, Fernando, CIRILO, Elder, et al. A framework for migrating relational datasets to NoSQL. Procedia Computer Science, 2015, vol. 51, p. 2593-2602.

[22] JONES, Dean, BENCH-CAPON, Trevor, et VISSER, Pepijn. Methodologies for ontology development. 1998

[23] SEN, Arun et SINHA, Atish P. A comparison of data warehousing methodologies. Communications of the ACM, 2005, vol. 48, no 3, p. 79-84.

[24] GOLFARELLI, Matteo, RIZZI, Stefano, et VRDOLJAK, Boris. Data warehouse design from XML sources. In : Proceedings of the 4th ACM International Workshop on Data Warehousing and OLAP. 2001. p. 40-47.

[25] MANCO, Carlo, DOLCI, Tommaso, AZZALINI, Fabio, et al.HEALER: A Data Lake Architecture for Healthcare. In : EDBT/ICDT Workshops. 2023.

[26] BENJELLOUN, Sarah, EL AISSI, Mohamed El Mehdi, LAKHRISSI, Younes, et al. Data lake architecture for smart fish farming data-driven strategy. Applied System Innovation, 2023, vol. 6, no 1, p. 8.