An Algorithm to Convert Class Diagram to Object Oriented Database

Fathalla R. Mansouri, Abdulhakim M. Etlawrghi, Wael Othman Alnaas
The Higher Institute of Engineering Professions, Benghazi -Libya

ABSTRACT

The Unified Modeling Language (UML) has been adopted and accepted in both academic and applied applications as a most widely used language in modeling and software development systems. The UML are also utilized to form relational, Object Oriented Databases (OODB), and object relational databases. The relational databases are used to develop the software systems; therefore, it is important to derive the relational tables, relationships among them, and the methods that are undertaken to perform the designed models.

This paper presents an algorithm with the required conversion steps based on the standard relational database management systems (RDBMSs). The implemented steps are classified mainly in two groups; the first group contains the steps to prepare whereas the second defines the steps to extract the relational tables with their methods from the class diagram. The group of the preparing steps is used to prepare the (Class Diagram), and to confirm its relation types, degree, and Cardinality ratio, all of these phases are normally conducted following to the audition and revision steps. Subsequent to the mentioned procedure, the (Class Diagram) is then converted to a group of relational tables, which will be mapped as an Object Oriented Database, OOD. The design procedure has been demonstrated analytically using a (case or notation) methodology with further analysis and comment details.

Keywords: UML, OODB, RDBMS, OCL, ORD.

الملخص

تستخدم لغة النمذجة الموحدة التي تعرف اختصارا باللغة الانجليزية **(UML)** على نطاق واسع وقد لاقت قبولا في كل من الاستخدامات الأكاديمية والتطبيقية باعتبارها اللغة الأكثر استخدامًا في أنظمة النمذجة وتطوير البرمجيات، حيث يتم ادماج واستخدام **UML** في تكوين قواعد بيانات علائقية وقواعد بيانات كائنية المعروفة بالرمز **(OODB)** وكذلك قواعد بيانات علائقية للكائنات. ونظرا لاستخدام قواعد البيانات العلائقية في تطوير أنظمة البرمجيات؛ لذلك أصبح من الضرورة الالمام بفهم وتوضيح اشتقاق الجداول العلائقية والعلاقات بينها والطرق التي يتم اتباعها لبناء النماذج المصممة.

في هذا البحث تستعرض خوارزمية تتضمن خطوات التحويل اللازمة اعتمادا على أنظمة إدارة قواعد البيانات العلائقية القياسية والمعروفة بالرمز **(RDBMSs)**، كما صنفت هذه الخطوات بشكل أساسي في مجموعتين، تحتوي المجموعة الأولى منها على خطوات التحضير، بينما تحدد الثانية خطوات استخراج الجداول العلائقية والاساليب المعتمدة من مخطط الفصل المعروف اصطلاحا بـــ **(Class Diagram)**. تُستخدم مجموعة خطوات التحضير لإعداد (**مخطط الفصل**)، وللتأكيد على أنواع علاقته ودرجته ونسبة العلاقة الأساسية والتي تسمى (**Cardinality ratio**)، بعد ان يتم اتخاذ خطوات التدقيق والمراجعة. وفي سياق تطبيق الخوارزمية المقترحة، سيتم تحويل (**مخطط الفصل**، **Class Diagram**) إلى مجموعة من الجداول العلائقية التي سيتم تعيينها على أنها (**قاعدة بيانات كائنية التوجه**). هذا وقد اعتمد البحث إجراء التصميم بشكل منتظم بتبني منهجية الحالة أو الترميز او كما تعرف بالانجليزية **(case or notation)** مع اضفاء مزيد من التفاصيل والتحليلات.

Introduction

Although, UML **[1-2]** has been accepted in both academic and applied sites as a de facto visual modeling language, however, non-formal and inaccurate except through repeated audits,there is no method or algorithm to verify them. Additionally, relational database and program design are the most important parts of software production in software engineering in general. The most important advantages of the relational model

are simplicity, clarity, information security, design power and ease of understanding which is considered one of the most important software tools for a good design.An object relational database **(ORD)** is composed of both a relational database management system **(RDBMS)** and an object-oriented database management system **(OODBMS)**. **(ORD)** contains characteristics of both the **(RDBMS)** and **(OODBMS)** models. The **(ORD)** is used to store data as same as traditional database, and can be accessed and manipulated using queries written in a query language such as **SQL**. Therefore, the basic approach of **(ORD)** is based on a relational database.

The inaccuracy of program design often leads to problems and difficulties in program production, especially if it designed in traditional and unorganized ways. The best way to design programs that reduce the problems of the design process and programs buildingis to develop a building and design plan by experience and test it for accuracy, which makes it easier for software designer to build a system with the least amount of problems,we will use a set of steps in two stages.

Since The second stage is the actually conversion process that receives the data from the class diagram and transfer it into the relational table schema so the First stage is the preparation process which by following it we will prepare the **(Class Diagram)**, audit it, revise and confirmation of its relations types, their degree and Cardinality ratio. By following the conversion algorithm the **(Class Diagram)** will be converted to a group of relational tables which will present an **(Object Oriented Database)**.

Thistransfer of the methods for each class may seem impossible if we consider the relational-ship attributes and derived attributes from other classes in the class diagram. Therefore, we have to develop some preparatory steps before the implementation steps.

The purpose of this paper is to make an algorithm which uses steps to transform the class diagram in a **UML** model to extract object relational database with all their attributes, and methods.

The rest of the paper is organized as follows: Section 2 reviews the status of the technical approaches for extracting the relational tables. Section 3 illustrates the proposed approach for extracting relational tables from UML class graphs and Part 4 concludes the paper.

Related works

In related works, transformation of models has been taken into consideration in both industrial and scientific fields.Researchers have conducted many research papers on this subject.

In Most recent researchers published papers there using graphical models derived from class diagram then transform them to relational tables which from our point of view has some difficulties and complexity in one way or another, in addition: some used a method to convert the class diagram to Entity Relationship Diagram **(ERD)**.

We present a short description of the papers in which one of the model transformation tools, were used to transform **UML** model to relational database model: Michael Lawley, in **[4]**, Introduced DATC **[4]** model transformation tool and TefKat **[4]** language, the authors explained core concepts of transformation rules, relationships evaluation and models definition. They examined the mapping of an object-oriented diagram as source model, and a relational database as destination model. They also studied the way in which the employed language to support the mapping ability and the reuse of transformation characters which are required to have a successful **MDA [4]** as an approach to generate high scale and long-life systems.

Jorn Bettin **[3]** presented concepts for a real syntax in model to model transformations, which are useful in industrialization of commercial software. In his paper, Jorn Bettin studied the transformation of UML **[10]** model to **RDBMS** model as an example of model to model transformation in which model's elements are inscribed form independent PIM platform to special PSM platform. The addition of transformation characters using OCL **[3]** in destination model elements and a textual model language for the mapping of source model elements and their transformation to destination model elements, as well as a visual description of model to model transformation by showing meta-models from the start to the end and by symbolization.

Gergely Varro' et al**. [6-8]** introduced a new approach to implement a graph transformation engine based on RDBMSs **(Relational Data Base Management System)**. The nature of this approach is to generate a database for each rule and for using pattern adaptation via internal linkage operations. As a result, the authors of this paper obtained a powerful and fast transformation motor which is particularly suitable

for followings: 1) Model development tools with a systematic **RDBMS** storage, and 2) embedded of model transformations inside large distributed applications in which models have been emphasized in a relational database as a recurring basis. It is also essential to controlcompatible large models.

Vahid Rafe et al**. [12]** present a formal yet automatic approach to extract relational tables from class diagrams. To do so, we adopt graph transformation systems. They have designed some graph transformation rules to derive necessary tables from **UML** class diagrams.

Ramez Elmasri, in **[11]** present used a method to convert the class diagram to an entity relationship diagram **(ERD)**, then convert to relational tables.

Our proposed approach

Through our studies and experiments we have become certain that all models and schemes of any modeling language are not the final solution or the steps that follow to analyze, design and build programs, we always believe that some modifications may be added, or even some excesses when using models, perhaps at the stage Depending on the nature of the program being built, some models are insufficient, unimportant, or even unstable.

In general, we look forward to find a set of steps that will move us from the analysis phase to the design phase, which, as mentioned earlier, begins with preparatory steps followed by conversion steps.

- **Preparatory steps:**
  - Data elements must be classified in their respective class.
  - The methods of each set of data elements must be defined and classified into a particular class.
  - Confirmation of the existence of a primary keys data elements or a distinct data elements in each class.
  - We consider classes at this stage as basic classes and then any class extracted after finding and studying the relationships is called a dependent, partial or derivative class as a result of a relationship between two or more classes.
  - The inheritance relationships must be observed and extracted, which is one of the most important organizational relationships.

- Ensure the existence and extraction of aggregation and composition relationships.
- Emphasis on the degree of relationship and cardinality ratio between two or more classes.
- In the association relationship must be emphasized to extract a new class representing the relationship between the basic classes, especially if the cardinality ratio of the relationship between the basic classes is (*) from both parties.
- Methods must be reviewed and updated for each class in the class diagram.

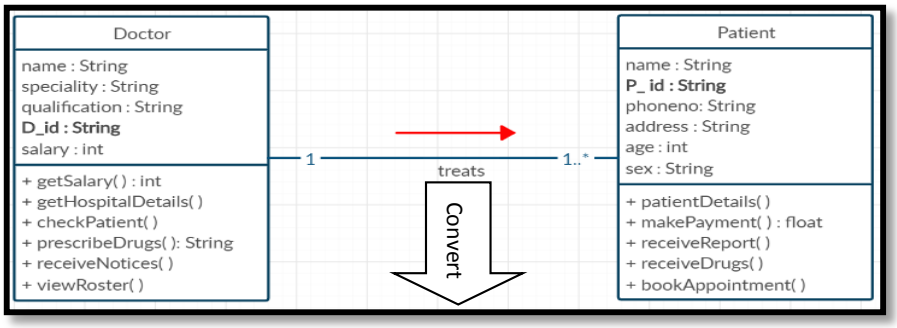
In the previous steps, the class diagram layout is constructed completely and accurately. It is a complete diagram that simulates the following stages such as the design process, and the software development process, and may help in the process of designing of the program interfaces. It cannot be overridden if we want to design and build a high quality software, because what matters is to build a design and develop a software.

- **Conversion steps:**

The following conversion steps are a logical design process for the data. These are the steps that we will list in a simple and clear way to design the object oriented relational tables those forming the object oriented database.

- Each basic class is an object oriented relational table that contains data elements and the operations performed on the data elements.
- If there is anassociation relationship between two classes and the cardinality ratio of the relationship (1) from one party and (*) from another party, the primary key must be included in the class from side (1) to the class from the side (*) after being converted to object oriented relational tables, and reset the primary keys to prevent duplication.

- As shown in Figure (1)



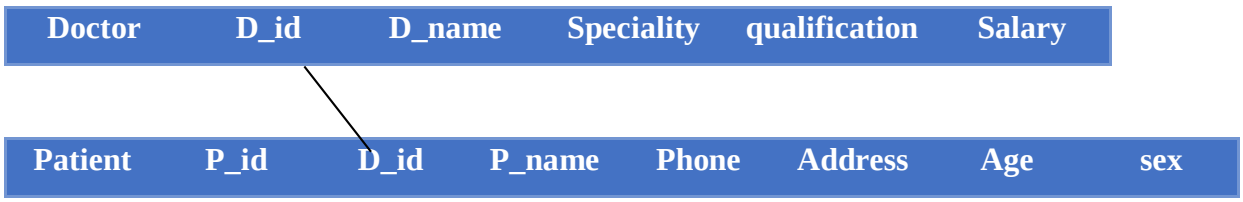
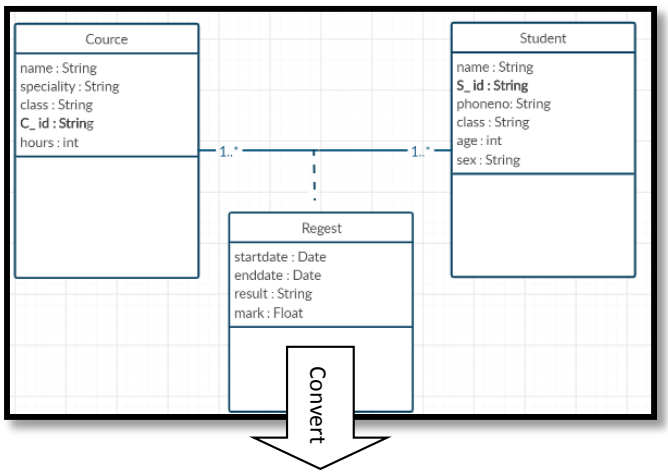


Figure (1). Converter the cardinality ratio of the relationship 1 to*.

- The new classes, which is derived from a relationships between two or more basic classes with the cardinality ratio **(*)** from both sides, is transformed into an object oriented relational table, It will contains the primary keys of the basic classes, and reset the primary keys to prevent duplication. As shown in Figure **(2)**.



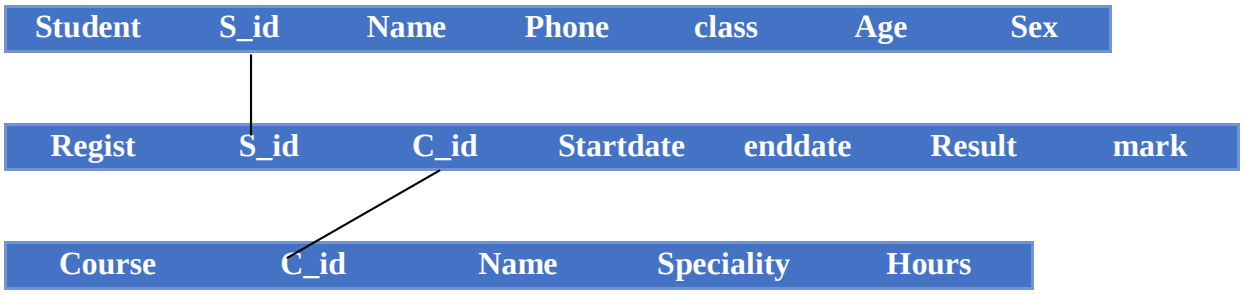


Figure (2). Converter the cardinality ratio of the relationship * to *.

- In the relationships of aggregation, composition and inheritance, the primary key of the primary class is included in the derivative or partial class may be called sub class, and reset the primary keys to prevent duplication. As shown in Figure (3).

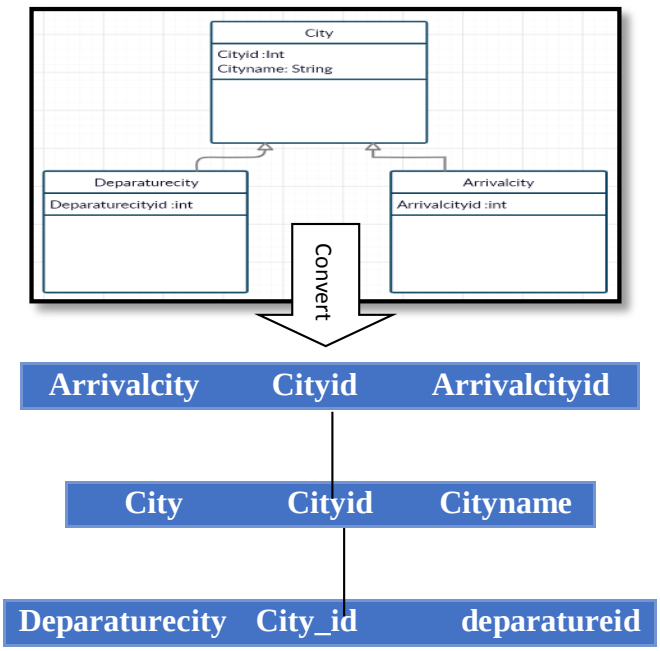


Figure (3). Converter the cardinality ratio of the relationship inheritance.

- After each of the previous steps the operations or methods are updated on the data.
- All object oriented relational tables are converted to object oriented database at the stage of physical data design.

Conclusion:

Through the previous process we tried to draw the reader's attention to the importance of classifying data elements and processes and find relationships between them, while building the class diagram, and then convert this diagram to an object oriented relational tables and finally will be converted to an Object oriented data base.

The question remains, do we need an object oriented data base? Is it the best way to design and build programs?

The answer to the previous question can only be answered by the following question, what program is required? What is the environment in which the program will work in?

* * * * * * * * * * * * * * * *

References

1. **Y. Ou., (1998)**.On Mapping between UML and Entity-Relationship Model,***intrnatinal workshop***,*uml'98' mulhouse*, 45-57.
2. **B. Appukuttan, T. Clark, S. Reddy, and L. Tratt and R. Venkatesh.,(2003)**. A model driven approach to building implementable model transformations, **Workshop in Software Model Engineering San Francisco, USA**.
3. **J. Bettin., (2003)**.Ideas for a Concrete Visual Syntax for Model-to-Model Transformations, **OOPSLA Workshop on Generative Techniques in the Context of Model-Driven Architecture**.
4. **M. Lawley, K. Duddy, A. Gerber and K. Raymond., (2004)**. Language Features for Re-Use and Maintainability of MDA Transformations, **OOPSLA workshop on Best Practices for Model-Driven Software Development, Vancouver, Canada.**
5. **G. Varro' and D. Varro', (2004)**. Graph Transformation with Incremental Updates, **Electronic Notes in Theoretical Computer Science, 109: 71-83**.
6. **G. Varro', K. Friedl and D. Varro', (2005)**. Graph Transformation in Relational Databases, 127: 167-180.
7. **G. Taentzer, K. Ehrig, E. Guerra, J.d. Lara, L. Lengyel, T.Levendovszky, U. Prange, D. Varro' and S.Varro Gyapay., (2005**). Model Transformation by Graph Transformation: **A Comparative Study, Satellite Event of MoDELS, Montego Bay 2005, Jamaica.**
8. **G. Varro', K. Friedl and D. Varro'. (2006)**. Implementing a GraphTransformation Engine in Relational Databases, **Software and Systems Modeling, 5: 313-341.**
9. **J. M. Vara, B. Vela, J. M.C. Barca, and E. Marcos., (2007)**. ModelTransformation for Object-Relational Database Development,**ACM, 1012-1019**.

10. **C. Anderton, L. Barroca, D. Bowers and M. Newton., (2009**). Converting Class Diagrams to Object-Relational Schema: **Using the MDA to exploit the expressive power of the full SQL, 1744-1986 30th**.
11. **R. Elmasri and S. B. Navathe., (2010**). Fundamentals of Database Systems, 13: 245-284 6th.
12. V**. Rafe, S. Jamali, M. Rahmani and F.Mahdian., (2011)**. From Class Diagrams to Relational Tables: A Graph Transformation-based Approach, **PRZEGLĄD ELEKTROTECHNICZNY (Electrical Review)**, 163-165.

* * * * * * * * * * * * * * * *