ABSTRACT

The object oriented and relational technologies are originated in distinct paradigms. The object-oriented technology is universally accepted approach for the development of business applications since it offers more appropriate methods and facilities for modelling of the real world objects. Meanwhile, relational database is still used to preserve the persistence of enterprise data. Connecting these two paradigms is not a straightforward task and necessitates a systematic mapping from one to another. Mapping from object model to relational database demands special attention because it would yield an object-relational impedance mismatch. One of the essential concepts in object technology is the idea of inheritance, and unfortunately relational database does not support this notion. This paper seeks to address various approaches in mapping from inheritance to tables and demonstrates how object-relational tool could be used to help software developers.

Keywords
Object Relational Mapping; Relational Databases; Object-oriented; Data Model; Software Development.

1 INTRODUCTION

Most of the software developers and industrial groups giving enormous effort trying to place together both object oriented technology and relational model in order to accomplish the current requirements of software development process [1 & 2]. Object oriented technology is considered as the mainstream approach in application development whereas the relational database is still perceived as the distinguished approach to store the enterprise data. Object Relational Mapping (ORM) is acknowledged as one of the programming techniques that attempt to map an object class into relational environment. Most of the fundamental ideas of object technology such as Aggregation, Inheritance, Polymorphism and Association are not supported in the relational database (4 & 12). Therefore in order to transform these concepts into the relational database, we need special algorithms and rules. Each of these concepts may be mapped with different methods [1]. This paper is going to discuss and elaborate one of the most common mappings algorithms: the mapping of inheritance structure to relational database. This paper will review various approaches and rules to map from inheritance to relational database. Based on the rules, we will develop a simple tool to
show how the mapping from inheritance to entity could be carried out.

2 RELATED WORKS

2.1 Object Oriented

Object oriented technology is a set of software design and development principles constructed on the concept of objects. Object oriented started with the development of the first object oriented programming language (OOP) of Simula in 1967 and then followed by Smalltalk in 1970 [4 & 8]. In the early 1990’s, object oriented technology has been adopted as one of the most renowned information system development methodologies. Object oriented technology suggested a different analysis and design concept: object-driven, instead of functional-driven [8 & 12]. There were many methods and techniques offered by various researchers to describe object oriented analysis and design such as Booch, Coad, Moses, OMT, OOSE, SOMA. Even though they varied mostly in terms of processes and life-cycle model, many object-oriented methodologies used the same diagram; and as a result, of their collaborative effort to unite the visual modelling language used in object-oriented methodology, the UML emerged. In 1995, Graddy Booch, Ivar Jacobson and James Rumbaugh worked together to form UML (Unified Modelling Language) as the object oriented standard for software analysis and design [8]. UML's main tools are the use cases (to describe users and functionalities of the system) and class diagrams (to portray the object model of a system).

2.2 Relational Database

Relational database concept was established in 1970 by Edgar F. Codd and grounded on the set theory to arrange data in terms of rows and columns [11]. Relational Database is considered as matured earlier than the object-oriented technology became the de facto standard for software development [4 & 12]. Most of business entities still rely on relational database management systems (RDBMS) such as Oracle, SQL Server, DB2, Sybase, Informix and etc. to organize, store and manage their enterprise data.

Relational database is based on entities and relationship among them. It is a collection of data items organized as a set of formally described tables from which data can be accessed easily. Relational database is considered as the most robust and reliable database management concepts because it enforce data normalization, where redundancy could be eliminated to ensure that there is only one source for each data element in the system, and therefore increase integrity through relationships [10].

2.5 Object to Relational Mapping

The notions of object paradigms are more concern to software engineering principles such as coupling, cohesion, inheritance and encapsulation. On the other hand, mathematical principles such as referential integrity and normalization are more bound to relational paradigm [3]. The impedance mismatch happens when these both paradigms employed in different perspectives and trying to access and manipulate data. Until now, various attempts have been made in a way to eradicate the impedance mismatch. Besides, modern developers have met several issues when it comes to mapping object to relational database. However, we will discuss only two existing issues because they are closely related to our objectives: mapping attribute and mapping inheritance structure [1, 3, 6, and 7].
2.6 Mapping Attributes To Columns

In relational database, class attributes are mapped to zero or more columns. In other words, an attribute will be mapped to zero or more columns depending on its persistent. Furthermore more attributes could be mapped to a single column in a table.

2.7 Mapping Inheritance

When mapping objects to tables, a number of pattern languages are designed based on the problem structure for each three problems: mapping aggregation, inheritance and associations. This paper will focus only on the inheritance. As we mentioned earlier, inheritance is not supported by the relational database and special algorithms and rules are needed to transform them into tables. Different strategies and approaches have been suggested in the previous researches for mapping inheritance structure of classes to the relational tables. Basically there are three fundamental approaches [1, 4, and 5]:

1) Vertical mapping: Map each class of the inheritance (abstract or concrete) to a separate table;
2) Horizontal mapping: Map each concrete class into a separate table; and
3) Filtered mapping: Map the whole inheritance structure into a single table.

In order to clarify the above mapping strategies, we will use the following figure as our reference’s model.

Solution 1: Vertical mapping: Map each class of the inheritance (abstract or concrete) to a separate table.

This approach is to map each class (abstract or concrete) to a separate table. In Figure 1, we have three different classes, Staff (superclass), Manager (subclass) and Sales (subclass). Each of these classes will be mapped into three different tables as shown in the following relations. The underlined attributes; primary key of superclass is inserted into each its subclasses (shows in relation form):

1. Staff (staff_ID, name, position, salary, DOB)
2. Manager (staff_ID, bonus, StartDate)
3. Sales (staff_ID, area, allowance)

Solution 2: Horizontal mapping: Map each concrete class into a separate table.

A second approach is to transform each concrete class to a separate table. All attributes of the superclass will be inserted along with each attributes of the subclasses.
into a table. Based on the figure 1, the result of this approach is in the following relations:

1. Manager (staff ID, name, position, salary, DOB, bonus, StartDate)
2. Sales (staff ID, name, position, salary, DOB, area, allowance)

**Solution 3:** Filtered mapping: Map the whole inheritance structure into a single table.

A third approach is also known as the simplest approach, which is mapping the whole inheritance hierarchy to a single table, the result of this mapping is in the following relation:

Staff (staff ID, name, position, salary, DOB, bonus, StartDate, area, allowance, skill, overtimeRate, is_manager, is_sales)

### 2.8 Adding a New Class into Inheritance Structure

Supposed that wanted to add a new subclass; called Technician with its specialized attributes of skill and overtime rate as in figure 2. Based on the mapping strategies discussed in the previous section, the new subclass will create the following solutions.

**Filtered mapping** approach will produce the following relationship.

Staff (staff ID, name, position, salary, DOB, bonus, StartDate, area, allowance, skill, overtimeRate, is_manager, is_sales, is_technician)

**Horizontal mapping** will add a new relationship called Technician.

1. Manager (staff ID, name, position, salary, DOB, bonus, StartDate)
2. Sales (staff ID, name, position, salary, DOB, area, allowance)
3. Technician (staff ID, name, position, salary, DOB, skill, overtimeRate)

**Vertical mapping** approach will add a new relationship called Technician, but only includes a minimum attributes.

1. Staff (staff ID, name, position, salary, DOB)
2. Manager (staff ID, bonus, StartDate)
3. Sales (staff_ID, area, allowance)
4. Technician (Staff_ID, skill, overtimeRate)

3. TOOL AND RESULTS

As we noted earlier, representation of database relationships are different in both paradigms. Relational model maintains the relationships between two relations by using the foreign keys; which normally the primary keys from other relations. Whereas the object oriented model uses object identifier (OID) as the main reference for the relationships. The purpose of this ORM tool is to fill the gap between object oriented technology and relational database paradigms.

Based on the algorithms and rules discussed in the previous section, we have developed an automatic ORM tool to help system developers to transform their inheritance structures into the relational database. The system developers do not have to concern about the underlying rules and algorithms in the transformation process. This following section describes how our ORM tool maps the inheritance structures into tables.

3.1 Steps to generate tables form inheritance structures.

The system will automatically create a database named ORM to store all classes, tables and attributes.

![Figure 3: Define superclass](image)

**FINDING SUPER CLASS**

![Figure 4: Define Attributes (Variables) of Superclass](image)

3.1.1 Define superclass and its attributes

All subclasses will inherit attributes from superclass. First, the system will prompt to user to enter superclass name followed by the number of its attributes as shown in figure 3. After that, user has to define the attribute (variable name), data type and the length of the attribute as shown in the following figure.
3.1.2 Define Subclasses and their attributes

The next step is to define the subclasses and their corresponding attributes as shown in figure 5 and 6.

The most important step is to transform the given inheritance structure into relational database tables. At this point, user will have three choices; to create a single table (Filtered Mapping) or; to create one class for each table (Vertical Mapping) or one class per concrete table (Horizontal Mapping).

3.1.4 The result for each mapping strategies.

The following diagrams shown how tables will be generated based on the user preferences.

Filtered Mapping is considered as the simplest strategy and required less maintenance and mostly suitable for a small database system. Any attributes of a new subclass will automatically insert into the same table. There is no join tables needed, and this approach yields the maximum performance for polymorphic queries. However, there are some drawbacks in this approach, which are [1, 2, 3 and 6]:

a. Waste a lot of space with the NULL values;
b. Need to add type identification for each subclasses;
c. Heavy traffic for the table since all subclasses are crowded in the same table;
d. Coupling within the class hierarchy is increased because all classes are directly coupled to the same table;
e. Any modifications in one class can affect the table which can then affect the other classes in the hierarchy;
f. Table can propagate quickly for huge hierarchies.

Figure 8: Vertical Mapping (One class one table)

**Vertical Mapping** is considered as the most efficient and acceptable approach by most of the software developer since it will maintain the data normalization. Transforming any other subclass will not affect the existing tables. New subclass will automatically create another table. This strategy however produces some disadvantages [1, 2, 3 and 6]:

a. Space consumption is realistic because each class involves one table. Nonetheless it might produce too many tables for a large database system;

b. Read, write and update data using this technique will take longer because we need to access and join multiple tables;
c. Not suitable to perform ad-hoc reporting.

Figure 9: Horizontal Mapping (Concrete table)

Figure 9 depicts the **Horizontal Mapping** where only concrete classes are transforming into tables and attributes of superclass will be added into each subclass tables. The space consumption is at the best because there no redundant attributes. Furthermore, this approach produce a good performance in terms of accessing a single object’s data, however yields few drawbacks such as [1, and 3]:

a. Altering a class will affect not only that particular table but it also will change the table of any of its subclasses.
b. This method will not support multiple roles and hard to conserve data integrity.
4 CONCLUSION AND FUTURE RESEARCH

Because of the divergence in the paradigms and representation of data in object technology and relational database, the system developers have consumed a lot of their treasured development time on maintaining consistency between the two approaches. On the other hand, ORM is formed to resolve the anomalies of data representation involving these two different paradigms. In this paper, we have discussed one of the essential concepts in object oriented technology, which is the inheritance; and have revealed how this inheritance structure could be represented into relational database. Then, based on the algorithms and rules discussed, we have established a simple tool to assist system developers to convert their inheritance structure into relational tables. This project has given us some insight to further our research into other essential concepts of object-oriented technology such as aggregation, composition and association.

5 REFERENCES