Composition in Object-Relational Database

David Taniar
Monash University, Australia

Eric Pardede
La Trobe University, Australia

J. Wenny Rahayu
La Trobe University, Australia

INTRODUCTION

Object-Relational Database (ORDB) is increasingly popular as the database storage. Its popularity is based on its ability to capture the object-oriented modeling semantic and the maturity of relational implementation.

Many works have proposed the design method for ORDB. The design includes different data structures and relationships. One type of relationship is *composition*. It is not the same as aggregation. While aggregation is identified as a relationship in which a composite object ("whole") consists of other component objects ("parts") (Rumbaugh et al., 1991), composition has two additional constraints mentioned below.

- *Non-shareable*. This is the case when one class can only be the part of one and only one other class. Treating this type as aggregation enables other classes to own the "part" class and thus, violate the conceptual semantic.
- *Existence-dependent*. This is the case when one class can only exist with the existence of another class. Treating this as aggregation will enable a "part" class to remain in existence even though the "whole" class has been removed.

These reasons have motivated us to differentiate between the composition and the aggregation. This work will focus to preserve the composition hierarchy.

In ORDB, composition can be implemented as the attribute of row type. Row type is the constructed data type that contains a sequence of attribute names and their data types (Fortier, 1999; Melton, 2002). Row type attribute will be fully dependent and exclusive to the object that owns it.

This article aims to propose models for preserving a composition in ORDB, and in particular, we introduce the use of row type. We also propose the main queries required for the composition hierarchy.

BACKGROUND

In this section, we show a brief overview on SQL row type. We also show the related work on composition relationship mapping in object-relational database.

Row Type: An Overview

Before we start using row type in ORDB, we will briefly discuss this data type. Row type is one data type available in the Structured Query Language (SQL). This language was introduced in 1970 and has emerged as the standard language for Relational Database (RDB) (Melton, Simon, & Gray, 2001). It is used for database definition and manipulation.

Along with the establishment of RDB and the emergence of ORDB, SQL has undergone many changes, one of which is the additional data types. It is required to accommodate complex data structures. One of the data type is Row Type.

Fortier (1999) and Melton (2002) define row type as constructed type that contains a sequence of attribute names and their data types. This type is actually not a new data type in the database system. It has been used even since the legacy data model era (CODASYL Database Task Group, 1971).

After the emergence of relational model, there is also a data model that is aimed at capturing the nested structure such as row type in relations. The model is called Nested Relational Model (Jaeschke & Schek, 1982; Roth & Korth, 1987). Nevertheless, traditional relational model still dominates the database community. Even until recently there is no commercial DBMS which has chosen to implement the Nested Relational Model even in its original form (Elmasri & Navathe, 2002).

Not until the release of SQL 1999, relational model recognizes row type as one data type that can enrich its data structure (Fortier, 1999). In SQL4, it is even possible to have varying levels of row type (Melton, 2002). It has become a powerful means to capture real world problems that can rarely be represented by a simple flat table. General syntax for the SQL4 row type is shown as follows:

SQL4 Row Type

CREATE TABLE <table schema>
($attr_1$ data type
CONSTRAINT $attr_1$ **PRIMARY KEY**, ...,
$attr_i$ **ROW** ($attr_{i1}$ data type,...,
$attr_{ij}$ data type));

Existing Mapping Methods for Composition in ORDB

Mapping method can be defined as a formal process of transforming a schema level (such as conceptual schema) to another schema level (such as logical schema) in a database system. A mapping method works effectively if the schema result is complied with the requests. If the result schema does not preserve all semantics of the requests, the mapping method is not effective.

Some works on mapping methods have tried to preserve composition hierarchy in ORDB. However, no work utilizes the row data types that were introduced by SQL 1999 (Fortier, 1999) and is enriched in SQL4 (Melton, 2002). Very often the "part-of" relationship is either flattened or split into an entirely separate table (see Figure 1).

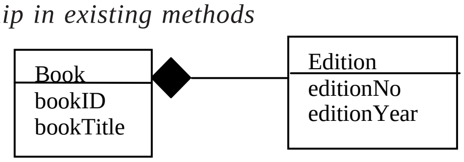
The most common practice of implementing composition relationship is by separating the "part" component in another table with composite PKs. This practice only uses object-oriented paradigm for conceptual modeling. The implementation is purely relational (Ambler, 1997; Hsieh & Chang, 1993).

Marcos, Vela, Cavero and Caceres (2001) propose a design of aggregation and composition in ORDB. In the logical and the implementation level, they use the nested table for the "part" component. Two problems arise. Firstly, nested table is a vendor-specific feature. It is not recognized in SQL standard. Secondly, the implementation of nested table cannot represent the composition type because actually the "part" component is stored in a separate table.

Rahayu and Taniar (2002) use two different ways of preserving this type of relationship. It uses index clustering for relationships that requires ordering semantics, and nested tables for the one which does not. However, this work can only be applicable for regular aggregation relationship.

We find that the existing works either have not preserved the composition semantic or have preserved it as

Figure 1. Composition relationship in existing methods



Composition flattened into separate tables

Book	
PK	
book_id	**book_title**
EN1	Fundamentals of Database System

Book-Edition		
PK, FK	PK	
book_id	**ed-no**	**ed-year**
EN1	1	1989
EN1	2	1997
EN1	3	2000

Composition split in nested table

Book		
book_id	**book_title**	**edition**
EN1	Fundamental of Database System	♦

edition
•
•
•

ed-no	ed-year
1	1989
2	1997
3	2000

Composition flattened using cluster

Book	
Cluster	
book_id	**book_title**
EN1	Fundamentals of Database System

Edition		
Cluster		
book_id	**ed-no**	**ed-year**
EN1	1	1989
EN1	2	1997
EN1	3	2000