ISSN No. 0976-5697

Volume 8, No. 5, May-June 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Semantic model for Complex object in Object oriented database

Sonal Kanungo
Smt.Z.S.Patel College of
Computer Application, Surat

Rustom. D Morena
Department of Computer Science
Veer Narmad South Gujarat University
Surat

Abstract: Object Oriented DataBase Management System is a collection of objects. The objects encapsulate the attributes and associated methods or member functions. One of the most important feature of Object-oriented databases have the capability to reference objects of complex structures, to make them perfect for complex data presentation. This capability is expected to build the semantic content of databases. They ought to support basic structural modeling and interrelationship normally. The support of complex objects forces a few necessities. The support of complex objects imposes several requirements.

Dynamic environment makeschanges in object and schema, that means change in classes, change in relationship between classes and even change in class lattice. This will affect not only stored objects as well as creation of new objects in database too.

Previous research presented models for the extended semantics of composite objects with some shortcomings. To eliminate shortcoming, we are presenting dynamic model for object oriented database applications to meet complex requirements.

Keywords:Relationship, Composite object, Association, Inheritance, Composition

1. INTRODUCTION

Object-oriented databases offer more adaptable presentation than traditional database systems. The object-oriented database will enable protection and security systems to be based on the notion of object. There is a natural correspondence between objects and real-world entities [1]. The data model incorporated into a database system characterizes a structure of concepts that can be utilized to express the real-world semantics of conventional (relational, network, hierarchical) data models, which has rigid framework therefore there will always be a semantic gap between an application/real-world and its database representation. Today, object-oriented databases based on the object-oriented data model are an attempt to limit this semantic gap [3]. The system designed using OODBMS are much nearer to the real world as the real-world system, as they are directly mapped into the system [7]. The evolution of objects must have done in such a way that transaction can reach out in time, provide the complex recovery and consistent concurrency control systems [13].

The evolutionary nature of object oriented database, theway of uses, expansion of domain and change in operations, ought to be supported robust security systems without block or shutdown system.These applications require support for the demonstrating and representation of complex objects and entities while conventional database and information management technologies are fundamentally record-based [2].

An object-oriented data model is a data model that enables any real-world entity to be demonstrated exactly as an object. A reference object is one whose presence relies on upon the presence of another object and is possessed by precisely one object. Previous studies lack the concepts such as composite objects and aggregate objects for defining and manipulating complex collections of related objects [3]. Therefore, new reliable model for composite object is needed for expressing the fullspectrum of the possible solutions of concurrencycontrol, recovery, which can also reduce the excess resource utilization, provide reduction in deadlocks and gives anefficient and concurrent output [14].

Our research focus on the semantics of composite objects and show their integration into the object-oriented data model.

2. RELATIONSHIPS BETWEEN CLASSES

A relationship is logical link between object of one class and object of another class. A relationship is “an abstraction stating that objects from certain classes are associated in some way; the association is given a name so that it can be manipulated. It is a natural concept used in ordinary discourse” [4].

Relationships should be dynamic that means it is possible to define new relationships or remove existing relationship between classes. Modification in relationship is also possible. Relationships can be bi-directional or unidirectional. Relationships can have a cardinality, either one-to-one, one-to-many, or many-to-many[7].

The object model specifically supports references. Object instances "reference" each other utilizing OID the identity of object. The relationships between classes helps us to know how objects can identify with each other [2].

The object-oriented data model in its conventional frame is adequate to speak to an accumulation of related objects. As we have seen, it captures the IS-A relationship between a class and its superclass, and it allows an object to reference different objects through its instance variables. IS-PART-OF relationship between an object and objects references, the idea of composite objects expressly captures this relationship [3].The semantics of the class relationships also useful to examined to discover their lock modes, granule sizes for characterizing concurrency control in Object orientated database system [12].

Object oriented database supports inheritance, association, composition and aggregation relationships.

1) Association

Association is a "has-a" sort relationship. Association establish the relationship between two classes utilizing through their objects. Relationship in Association can be one to one, one to many, many to one and many to many [7]. For example, assume we have two classes then these two classes are said to be "has-a" relationship if both of these elements share each other's object for some work and at the same time. They can exist without each other's reliance or both have their own particular life time. To quality as an association, and object and another object must have the accompanying relationship [15].

2) Aggregation

Like composition, an aggregation is as yet a part-whole relationship, where the parts are contained inside the entire, and it is a unidirectional relationship [2].

In any case, not at all like a creation, parts can have a place with more than one object at any given moment, and the entire object is not in charge of the presence and lifespan of the parts. At the point when an aggregation is created, the aggregation is not in charge of creating the parts. At the point when an aggregation is demolished, the aggregation is not in charge of destroying the parts. We can say that aggregation models "has-a" relationships. To quality as an aggregation, an entire object and its parts must have the accompanying relationship [7].

3) Composition

Composition is a "part-of" relationship. Basically, composition means utilization of instance variables that are references to different objects.This relationship enables classes to be generic for their protection mechanisms, which are specified when instances are created. Protection is based on object ownership: every object has exactly one fixed owner [11].

In composition relationship both elements are related of each other for example "motor is part of car", "heart is part of body"[15]. Give us a chance to take an example of car and motor. Motor is a part of each car and both are subject to each other [16]. To qualify as a composition, an object and a part should have the accompanying relationship.

4) Inheritance

Inheritance is where the one class inherits the attributes and methodsfrom another class i.e. from parent class. The advantage of inheritance is that the child relationship doesn't have to redeclare and redefine all the entities which it inherits from the parent relationship. It is thusly a way to reusability [9].

Inheritance where base relationship has generic code which is shared by relationships in an inheritance hierarchy. The inheritance can be classified as exclusive inheritance or shared inheritance. The inheritance can be single inheritance, multilevel inheritance, multiple inheritance allows selective inheritance of a parent class to at least one child classes. In any case, in hierarchical inheritance, several sub classes are acquired from the same parent class or the parent is shared by many siblings.

Inheritance is "IS-A" type of relationship. Inheritance is a parent-child relationship where we create a new class by using existing class code. Examples of inheritance can be that "A is type of B". For example, is "Apple is a fruit", "Ferrari is a car" [2].

3. COMPOSITE OBJECT

Two types of class hierarchies may be created. One is the IS-A hierarchy where a class has subclasses associated with it. The second-class hierarchy is the IS-PART-OF hierarchy. Here an object of a class is considered to be the aggregation/composition [8].

An object has a number of attributes; the value of an attribute is itself can be anobject [1].A composite object has a solitary root object, and the root references different children objects, each through an example variable. Every child object can thus reference its own particular children objects, again through occurrence factors. A parent object might only claim children objects, and all things considered the presence of children objects is predicated on the presence of their parent. Children objects of an object are along these lines dependent/independent Objects. The object contains the references to both dependent objects and independent objects [10].

The nesting of objects in an object-oriented data model is another intense idea. One essential organization which should be superimposed on the nested object is the IS-PART-OF relationship, that is, the thought that an object is a part of another object. An arrangement of component objects which shape a single logical entity has been called a composite object or a complex object [6].

A Vehicle instance then is an object which contains a Body object and a Drivetrain object, where a Body object has a set of Door objects, and a Drivetrain object comprises of an Engine and a Transmission, and a Door has a Position. [3].

We characterize a composite object as an object with a hierarchy of selective segment objects, and allude to the hierarchy of classes to which the objects have a place as a composite object hierarchy [3].

If component object is only part of one composite object, in which an object cannot be part of more than one object that means they are independent and relate for some particular time is also possible. Be that as it may, it doesn't capture the IS-PART-OF relationship between objects; one object just references, in any case, does not possess, different objects [5].

The object-oriented data display, in its conventional frame, is adequate to speak to a gathering of related objects. A composite object hierarchy captures the IS-PART-OF relationship between a parent class and its segment classes [6].

The model strengths a top-down production of a composite object; that is, before a component object might be made, its parent object should as of now exist. This keeps a bottom-up production of objects by amassing effectively existing objects. The model requires that the existence of a component object relies upon the existence of the parent object; that is, if an object doesnot exist, all its component objects are likewise deleted. Since it liberates the applications from searching and erase all nested components of a deleted object [3].

4. LITERATURE SURVEY

1. **Won Kim, Jay Banerjee, Hong-Tai Chou,Jorge F. Garza, Darrell Woelk: "Composite Object Support in an Object-Oriented Database System**"

This paper presented the research into composite objects, have been implemented in ORION. they described the basic semantics of composite objects under an object-oriented data model, two major extensions to the semantics of composite objects which have turned out to be necessary subsequently of our support of dynamic schema evolution and versioning of objects. Also, presented systems we use for taking advantage of the semantics of composite objects in enhancing the performance of a database system, by using the composite object as a unit of clustering on
disk, and as a unit of concurrency control in retrieving from the database.

2. **Won Kim, Elisa Bertino, Jorge F. Garza: "Composite Objects Revised"**

This paper presented another model of composite objects by neatly isolating out various different semantics which the model of composite objects developed overloaded on the reference between a couple of new model distinguishes four sorts of composite reference, that is, a reference on which the IS-PART-OF relationship between a couple of objects is superimposed. This include independent exclusive, dependent exclusive, dependent shared, and dependent shared composite references.

3. **Xiaoyan Lu, J. WennyRahayu, David Taniar: "ODMG Extension of Composite Objects in OODBMS: A Proposal"**

This paper proposes an augmentation of ODMG (Object Data Administration Group) standard for the Object-Oriented Database Administration Systems (OODBMS). The augmentation concentrates on composite objects, which gives another worldview, and furthermore enhances customary OODBMS to address the issues emerging fromthe aggregation hierarchy. As of now in ODMG, the semantic of the aggregation relationship is investigated at the displaying stage what's more, is portrayed in normal dialect.

4. **Alisdair Wren: "Relationships for object-oriented programming languages"**

In this thesis, demonstrates how relationships are all around represented in models of object-oriented systems, and therefore in programmer intuition, however not in object-oriented languages themselves.

5. SIMULATION

We had created number of user defined dynamic classes and objects. Domain is made up of classes and relationship is established between classes.Different types of relationships are presented, they can be dependent or independent. Classes can have independent relationship like associations. In associations, all classes and their objects are independent, object and classes can create or destroy independently. 'Uses' relationship is established with associative classes. Objects can have reference of each other. Reference can be unidirectional and bidirectional. Cardinality between reference objects can be one-one, one-many, many-many, many-one.

Inheritance is created between classes; the use of inheritance is reusability. IS-A relationship is created between classes. Classes can be super classes and sub classes. A superclass can be derived into more than one subclasses. A subclass can be inherited from more than one super classes. Superclass can be inherited into more than one levels of sub classes.

Composition is having 'ISA-Part of' dependent type of relationship between classes. First owner class will create then other dependent classes can be created. Objects of owner class is containing objects of dependent classes. No object can create or destroyed without each other.

We can also create nested relationship, that means relationship can be further established between existing relationships. Any kind of domain with n number of relationships can be created. The below example banking system is created.

Classes their relationships and cardinality is established according to it n number of objects and relationship between objects are established. Our system can also change and add relationship to already existing relationships. This is a Dynamic system, classes and their relationships can be change any time.

6. FIGURE AND DIAGRAM

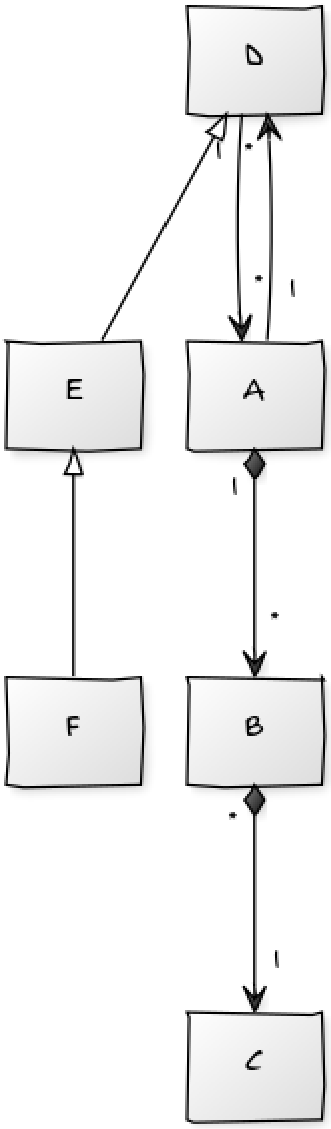

Figure 6.1 Class Relationship

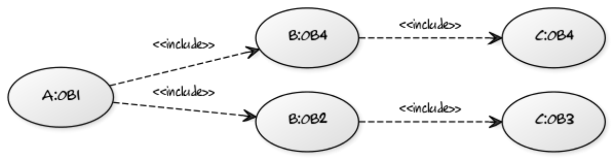

Figure 6.2 Object relationship Composition

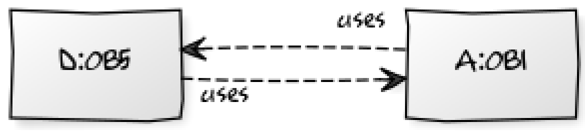

Figure 6.3 Object relationship Association

Figure 6.1 represents relationship between classes. These classes are having relationships like Association, Composition and Inheritance. Figure 6.2 Composite object shows outer object includes objects of another class.
Figure 6.3 Binary Association between objects of two classes.

7. SCHEMA EVALUATION

Previous methods only describe complex object of only one type of relationship. Our model represents complex model from relationship of relationship. That means object can create not only from single class, it be can create from multiple relationships.
We can say that an object A has a reference to (or references) another object B, if B contains the object identifier (OID) of A.
We will recognize two sorts of reference from one object to another: Associative and Composite. A composite reference is a feeble reference expanded with the IS-PART-OF relationship; a composite reference from B to an implies that A will be a part of B.
We additionally refine the semantics of a composite reference, on the premise of whether the existence of an object relies on upon the existence of its parent object; that is, a composite reference might be dependent or independent. A dependent composite reference from B to an implies that the existence of A relies on upon the existence of B; while an independent composite reference does not convey this extra semantics. The cancellation of an object will trigger recursive cancellation of all objects referenced by the object through dependent composite references.
Associative objects are independent, they are not dependent on each other not with objects of its associative class. They have independent existence so they can survive without each other. Objects can create and destroy independently.
Our system is establishing relationships. when we want to access these related classes, or objects this system give faster results, as we are saving relationships in form of objects, which are persist and easy to access.
Schema can be change during Adding a new instance variable to a class, drop an existing instance variable from a class, Change the Default value of an instance variable, adding or dropping of class.

8. DISCUSSION

Complex objects are used in applications like computer-aided design, computer-aided software engineering, multimedia and image databases, and document/hypertext database.
The object, which encapsulates both state and behavior, is a more normal and realistic portrayal of real-world objects. OODBMS is more qualified to taking care of complex, interrelated data than a RDBMS implies that an OODBMS can beat a RDBMS relying upon the complexity of the data being dealt with. The object-oriented data model permits the 'real world' to be modeled all the more nearly.
An object can store all relationships it has with different objects, including many-to-many relationships, and objects can be framed into complex objects that the conventional data models can't adapt to effortlessly. A complex object can be controlled all in all, yet parts of it (related objects) too, Addition and deletion operations may have a related object as an operand. It might likewise happen that a related object is moved inside a complex object.Composite objects are easy to access with OID;therefore, they can give better concurrency control and recovery also.

Conclusion

In this paper, first we displayed another model of composite objects by neatly isolating out various distinctive semantics which the model of composite objects developed of composite reference, that is, a reference on which the IS-PART-OF relationship between a couple of objects is superimposed.
Next, we investigated the results of the new model of composite Objects on the semantics of schema evolution, authorization, on composite objects. In this paper, we encourage the utility of composite objects by demonstrating their utilization as a unit of authorization. We can further use these objects for concurrency controland recoverypurposes.

REFERENCE

[1] JAY BANERJEE, HONG-TAI CHOU, JORGE F. GARZA, WON KIM, DARRELL WOELK, and NAT BALLOU: "Data Model Issues for Object-Oriented Applications", ACM Transactions on Office Information Systems, Vol. 5, No. 1, January 1987, Pages 3-26.
[2] BANERJEE. J., W. KIM, H.J. KIM, AND H.F. KORTH: "Semantics and Implementation of Schema Evolution in Object-Oriented Databases," in Proc.ACM SIGMOD Conference, 1987.
[3] WON KIM, JAY BANERJEE, HONG-TAI CHOU, JORGE F. GARZA, DARRELL WOELK:"Composite Object Support in an Object-Oriented Database System", OOPSIA, 87 Proceedings, October 4-8, 1987, page 118-125.
[4] RUMBAUGH, J.: "Relations as Semantic Constructs in an Object-Oriented Language", SIGPLAN Notices, Vol.22, No.12, 1987, pp.466-481
[5] GARZA, J. E, AND W. KIM: "Transaction Management in an Object-Oriented Database System," In Proc. ACM-SIGMOD Intl. Conf. on Management of Data, Chicago, May 1988.
[6] WON KIM, ELISA BERTINO, JORGE F. GARZA: "Composite Objects Revisited", 1989 ACM, Page no 337-347.
[7] JAMES RUMBAUGH MICHAEL BLAHA WILLIAM PREMERLANI FREDERICK EDDY WILLIAM LORENSEN: "Object-Oriented Modeling and Design", Prentice Hall, 1991.

[8] M.B. THURAISINGHAM: “Mandatory Security In Object-Oriented Database Systems”, ACM OOPSLA ‘89 Proceedings, October 1-6 1989, page 203-210.

[9] JUHN YOUNG LEE AND SANG H. SON,MYUNG-JOON LEE: “Issues in Developing Object-Oriented Database Systems for Real-Time Applications”, 1994 IEEE.

[10] XIAOYAN LU, J. WENNY RAHAYU, DAVID TANIAR: “ODMG Extension of Composite Objects in OODBMS: A Proposal”,40th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS Pacific 2002), Sydney, Australia.

[11] ABDELSALAM SHANNEB JOHN POTTER: “Flexible Exclusion Control for Composite Objects”,the Australasian Computer ScienceConference(ACSC 2005), The University of Newcastle, Australia,2005

[12] VENKATASUBRAMANIAN GEETHA AND NILADURI SREENATH: “Semantic Multi-Granular Lock Model in Object Oriented Database Systems**”,** International Journal of Database Theory and Application Vol. 6, No. 1, February, 2013.

[13] SONAL KANUNGO, R.D. MORENA: “Analysis and Comparison of Concurrency Control Techniques”, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 3, March 2015.

[14] SONAL KANUNGO, R.D. MORENA: “Comparison of Concurrency Control and Deadlock Handing in Different OODBMS”, International Journal of Engineering Research & Technology, Vol. 5 Issue 05, May-2016.

[15] A UNIFORM APPROACH AWAIS RASHID, PETER SAWYER:” Dynamic Relationships in Object Oriented Databases”

[16] http://www.infoworld.com