Int. J. Open Problems Compt. Math., Vol. 1, No. 1, June 2008

The Constraints of Object-Oriented Databases

Belal Zaqaibeh and Essam Al Daoud
Department of Computer Science, Zarqa Private University, Jordan
e-mail: belal@zpu.edu.jo

Abstract

Object-Oriented Databases (OODBs) have been designed to support large and complex programming projects. The data accuracy, consistency, and integrity in OODBs are extremely important for developers and users. Checking the integrity constraints in OODBs is a fundamental problem in database design. Existing OODB Management Systems (OODBMSs) lack to a capability of an ad-hoc declarative specification of enforcing and maintaining integrity constraints that are appeared among attributes in association, composition, and inheritance hierarchies' relationships. A critical problem in the existing OODBs is that they cannot support User-Defined Constraints (UDCs) that can be defined in classes with composition (logical or physical composition) and inherence (single or multiple inheritance) hierarchies. Integrity constraints in the current OODBMSs are maintained either by disallowing and rolling back transaction or modifying operations that may produce a violation.

1 Introduction

Integrity constraints refer to the expression of integrity validity and do not include the enforcement or the maintenance part. The term integrity covers consistency (data is well organized in accordance with the requirements of a data model) and validity (all invalid data is excluded from the database).

The proper handling of integrity constraints is essential to any data storage and management. Handling integrity constraints is an essential premise to managing semantically rich data [1], [2]. In Object-Oriented Databases (OODBs), checking the integrity constraints is a fundamental problem in the database design [1], [3], [4]. The automated verification of constraints and their enforcement provided by current OODB Management Systems (OODBMSs) is limited [1], [5] due to the user participation is required.

Maintaining constraints that are scattered in applications is called Application-Oriented Integrity Maintenance (AOIM) [5], [6]. Centralizing the management of integrity constraints by extending database systems to have a dedicated component for constraint enforcement is called Centralized Integrity Maintenance (CIM) [5], [6], [7].

OODBMSs do not have adequate support for certain types of constraints especially the ones defined in a class composition and inherence hierarchies [1], [6], [8], [9], [10], [11], [12]. The integrity constraints must be maintained in the backward direction along the class composition and inheritance hierarchies as well as in the forward direction. The Assertion Model of Integrity Constraints (AMIC) [4] keeps the derivation path along with the attributes' relationships that are derived from association, composition, and inheritance hierarchies. The AMIC techniques are designed to implement the needed functions that are collecting the attributes' relationships and checking the integrity constraints.

2 Backgrounds

The class composition hierarchy is represented by IS-PART-OF relationship, and class inheritance hierarchy is represented by IS-A relationship [13], [14], [15]. The Object-Oriented Data Model (OODM) can support three types of relationships between classes, which are:

- Composition hierarchy (logical or physical composition) is a relationship between two classes where the instances of one class are in someway attributes, methods, and constraints of the other.
- Inheritance hierarchy (single or multiple inheritance) is a relationship between superclasses and subclasses. A superclass may have any number of subclasses, which subclasses inherit attributes and methods of superclass. This means all global attributes, methods, and constraints in a superclass exist in subclasses. In addition, subclasses may have additional attributes, methods, and constraints.
- Class association is a relationship between classes that can be in the form of 1:1, 1:M, or M:N.

Composite objects are grouping of inter-related objects that can be viewed logically as a single object. Composite objects are typically used to model relationships that have the semantic meaning of IS-PART-OF (e.g., wheels are part of a car). The methods that are applied to the root object can be propagated to all objects within that group. Corresponding to the references along the composite relationship among objects, the class composition hierarchy arises from the aggregation relationship between a class and its attributes [13], [14], [15].

New data type can be defined from composition hierarchy, and it can be a mix of imperative types (number, char, date, etc.) and a collection of objects (set, bag, and sequence) [2], [14]. A set contains an unordered group of objects of the same type. Since no duplicates are allowed in sets, this reduces the number of constraints to be checked when an event occurs. A bag contains an unordered group of objects of the same type. Unlike a set, duplicates are allowed in a bag [2], [16]. A sequence contains an ordered group of objects of the same type where duplicates are allowed. This collection of objects enables us to do searching using the keys of the related information for any object in the database. Those keys are known as Object Identifiers (OID). OID is an internal OODB identifier, which might include the page number and the offset where the object will be stored [16], [17].

Constraints rules are important to manage the integrity constraints. The notion of object-oriented that constraints are used to define the connectivity among objects required for the valid expression of constraint and rule conditions. One of these rules is the Object Assertion Language for Integrity Constraints (OALIC) which can be used to create classes and collect attributes and their constraints that are derived from composition and inheritance hierarchies [18]. The OALIC is designed to support enforcing and maintaining integrity constraints for OODBs. A new technique called detection method is designed to check the object metadata to detect and catch the OODBs violation before it occurs.

A constraint Rule Language (ARL) that is designed to support a database environment that provides general means for specifying active database rules with an object-oriented view of data [16]. The language was developed as an extension of Assertion Language for Integrity Constraint Expression (ALICE) [16].

The ARL rules have a syntax that corresponds to the syntax of Event Condition Action (ECA) rules in active database systems. A unique rule name is required for each rule. Rules can also belong to specific rule sets so that different users can have different sets of rules, and different sets of rules can be used in different situations, thereby reducing the scope of rule searches [16].

Terminology Query Language (TQL) [19] is a simple query language interface to serve implementations of concept-oriented terminologies. TQL*, is a fragment of the language TQL++ for conceptual modeling of OODB applications [20]. TQL* addresses the satisfiability of a specific class of object-oriented constraints, including recursive types, bags, sets, cardinality constraints, and a restricted form of path constraints. The data model of Formica work is based on TQL*, that it is aimed at modeling the structural aspects and the integrity constraints of an OODB application. A constraint with two fields (operands) cannot be presented in TQL* [20].

3 The Importance of the OODBs

Advanced database applications like Computer-Aided Design (CAD), Computer-Aided Manufacturing (CAM), Computer-Aided Software Engineering (CASE), Office Information System (OIS), Multimedia System, Geographic Information Systems (GIS), and Interactive and Dynamic Web Sites are focusing heavily on the use of OODBs technology as the process integration framework. This is due to the OODBMSs that can manage complex and highly interrelated information. The Relational Databases (RDBs) have showed some shortcomings in managing this type of information.

The data accuracy, consistency, and integrity in OODBs are extremely important for developers and users. Checking the integrity constraints in OODBs is a fundamental problem in database design [9], [10], [20]. It is not an easy task to detect violations and check integrity constraints in the OODBs. This is due to the fact that detecting all constraints that appear as a result of composition and inheritance of hierarchies requires an efficient and smart model, which is something that has not been tackled in current OODBMSs [9], [20].

Existing OODBMSs lack the capability of an ad-hoc declarative specification of maintaining the integrity constraints [5], [10], [12]. This is because the automated verification of constraints and their enforcement that is provided by current OODBMSs is limited. A critical problem in the existing OODBs is that they cannot support User-Defined Constraints (UDCs) that are defined in the class composition or inherence hierarchies. The existence of facilities to handle semantic integrity constraints is an essential premise for managing semantic data. Since integrity constraints are scattered in each application, it is difficult to maintain them consistently [5], [20].

Integrity constraints in current OODBMSs are maintained by rolling back a certain transaction, disallowing a transaction, or modifying operations that may produce an inconsistent state for the OODBs [5], [21]. Consequently, the need for an integrated OODB increased and the emphasis on process integration has become a driving force for the adoption of the OODBs. Critical problems in the current OODBs are that:

- Cannot support UDCs that are defined in a class composition (logical or physical composition) hierarchy.
- Cannot adequately (it supports but not efficient enough) support UDCs that are defined in an inherence (single or multiple inheritance) hierarchy.
    - Cannot support binary operands (attributes or aggregate functions) for the constraints expression.

  - Cannot detect the constraint violation among inter-object and intra-object constraints in forward and backward directions.

- Cannot detect the violation and derivation path for the composed and inherited constraints.
- Cannot enforce inconsistent and duplicate UDCs.
- Cannot enforce and maintain adequately the redundant (subset constraint) UDCs, which it can enforce a subset constraint on a particular attribute but cannot enforce two separated constraints on a particular attribute.
- Lack to an efficient constraint optimization method to reduce coupling.
- Cannot analyze, enforce, and maintain UDCs automatically without user participation.

4 Open Problems

OODBs lack the capability for an ad-hoc declarative specification of maintaining integrity constraints. Supporting integrity constraints in OODBMS requires a high integration of the constraints with the rich concepts available. A needed work can be done in the following directions:

- Enforcing and maintaining integrity constraints during the run-time when copying object (instance of a superclass) to another object (instance of a subclass) and vise versa. For such problem two concepts must be taken in account which are: Downcasting (it moves down a hierarchy, which is converting a superclass to a subclass) and Slicing (it is the process of converting an instance of a subclass into an instance of its superclass).
- Multiple inheritance occurs when a class inherits attributes from more then one superclass. If the same attribute name of more than one propagated attributes exist in the superclasses the declaration of the pure virtual functions or virtual classes is needed.
- The integrity model is built on the object-oriented data model. The object-oriented data model can be combined with the relational model to form the so-called object-relational model, which provides an SQL-like interface but organizes the data in object-oriented structure.
- Finally, In the integrity model, there are several commonly seen constraint domains. As the applications of OODBs to different areas, more constraint domains may be developed such as image, video, and audio.

References

[1] Formica A. 2002. “Finite Satisfiability of Integrity Constraints in Object-Oriented Database Schemas”. *The IEEE Transactions on Knowledge and Data Engineering*, Vol. 14, No. 1, pp. 123-139.

[2] Rao B. R. 1994. “Object-Oriented Database Technology, Applications, and Products”. McGraw-Hill.

[3] Belal Zaqaibeh, Hamidah Ibrahim, Ali Mamat and Md. Nasir Sulaiman, "Enforcing User-Defined Constraints during the Run-Time", *The International Arab Journal of Information Technology (IJAIT)*, 2008.

[4] Belal Zaqaibeh, Hamidah Ibrahim, Ali Mamat and Md. Nasir Sulaiman, “An Assertion Model for Controlling Integrity Constraints in an Object-Oriented Database”, *Proceedings of the Joint International Conference on Informatics and RWICT*, 2004, Vol. 1 No 1, pp: 413-421.

[5] Eick C. F., and Werstein P. 1993. “Rule-Based Consistency Enforcement for Knowledge-Based Systems”. *The IEEE Transactions of Knowledge and Data Engineering*, Vol. 5, No. 1, pp. 52-64.

[6] Do N. C., Bae S., and Choi I. J. 1997. “Constraint Maintenance in Engineering Design System: An Active Object-Oriented Approach”. *Computers and Industrial Engineering*, Vol. 33, No. 4, pp. 643-647.

[7] Urban S. D., and Desiderio, 1992. “CONTEXT: A Constraint Explanation Tool”. *Data and Knowledge Engineering*, pp. 153-183.

[8] Hamidah Ibrahim, Belal Zaqaibeh, Ali Mamat, and Md. Nasir Sulaiman, “Enforcing and Maintaining Constraints Base during the Compile-Time”, *The World Scientific and Engineering Academy and Society (WSEAS) Transactions on Computer Research,* February 2007, No. 2, Vol. 6, pp: 373-379.

[9] Bagui S. 2003. “Achievements and Weaknesses of Object-Oriented Databases”. *Journal of Object Technology*, Vol. 2, No. 4, pp. 29-41.

[10] Do N. C., Choi I. J., and Jang M. 2002. “A Structure-Oriented Data Representation of Engineering Changes for Supporting Integrity Constraints”. *The International Journal of Advanced Manufacturing Technology*, Vol. 20, No. 8, pp. 564-570.

[11] Choi I., Bae S., Do N., and Yun M. 1997. “Backward Propagation of Engineering Constraints in Active Object-Oriented Databases”. *Proceedings of the 22nd International Conference on Computers and Industrial Engineering*, Cairo, pp. 20-23.

[12] Junmang R. 1997. “Formal Specification of Integrity in Object-Oriented Database Systems”. PhD Thesis, Kansas State University.

[13] Graham I. 2001. “Object-Oriented Methods Principles and Practice”. Addison Wesley.

[14] Brown P. 2001. “Object-Relational Database Development”. Addison-Wesley.

[15] David W. and Embley, 1998. “Object Database Development Concepts and Principles”. Addison-Wesley.

[16] Urban S. D., and Wang A. M. 1995. “The Design of a Constraint/Rule Language for an Object-Oriented Data Model”. *Elsevier Science System Software*, Vol. 28, pp. 203-224.

[17] Kifer M., Bernstein A., and Lewis P. M. 2005. “Database Systems an Application-Oriented Approach”. Addison Wesley.

[18] Belal Zaqaibeh, Hamidah Ibrahim, "An Object Assertion Language for Integrity Constraints in OODBs", *Proceeding of the 2nd International Conference on Informatics*, Kuala Lumpur, Malaysia, 27-28 November 2007, Vol. 2, pp: 29-36.

[19] Hogarth M. A., Gertz M., and Gorin F.A. 2000. “Terminology Query Language: A Server Interface for Concept-Oriented Terminology Systems”. *AMIA Symposium*, pp. 349-353.

[20] Formica A. 2003. “Satisfiability of Object-Oriented Database Constraints with Set and Bag Attributes”. *Elsevier Science*, Vol. 28, No. 3, pp. 213-224.

[21] Do N. C., and Choi I. J. 1994. “Backward Propagation of User-Defined Integrity Constraints on Active Object-Oriented Database”. *The Korea Database Journal*, Vol. 1, No. 1, pp. 63-81.