Declarative and Procedural Object-Oriented Views

Ralph Busse, Peter Fankhauser
GMD-IPSI, Integrated Publication and Information Systems Institute
Dolivostrasse 15, 64293 Darmstadt, Germany
{busse,fankhaus}@darmstadt.gmd.de

Abstract

One major approach to realise database integration is to adapt and merge the database schemas by defining views. When integrating object-oriented databases, the views need to adequately support object identity and methods. View objects need to be identified on the basis of the objects they have been derived from. Methods must be callable from the query processor without impeding query optimisation. Our view system for ODMG-93 supports both declarative and procedural integration of object-oriented databases. It provides flexible integration semantics without sacrificing the optimisation potential.

1. Introduction

Views are a powerful means to integrate heterogeneous databases. They serve to identify overlapping or related portions from several databases, and to overcome modelling differences in naming, scaling, structure, and semantics between the identified portions. Thereby applications can access heterogeneous databases like a single, integrated one.

When integrating object-oriented databases, views need to adequately support the two main concepts of object-oriented data models: Object-identity and methods. Object-identity gets a particular flavour for database integration. Unlike in integrated databases, where every object is uniquely distinguished by its identifier, database integration leads to objects that are derived from several objects. The identity of these derived objects should thus be determined from the identity of their constituent objects.

Methods allow to overcome arbitrarily complex modelling differences, and thereby surpass the integration capabilities of declarative query languages used for relational views. But the execution of such methods requires an efficient dynamic or static binding of their implementation to derived objects, and according method dispatching mechanisms. Furthermore, the implementation of methods can not be used by a query optimiser. Therefore, methods need to be complemented and substituted by declarative query mappings where possible.

2. Approach

In the IRO-DB project [2][3], we have developed an object-oriented view concept on top of the ODMG-93 standard. A view is declaratively specified with ODMG's object definition and object query languages. A C++ language binding is automatically generated from the specification and can be further enhanced with C++ methods. An object manager implements the necessary functionality for consistent identification of view objects, object activation and deactivation, and dynamic object accesses.

Each view consists of a set of classes, which are populated with newly generated proxy objects. Each proxy object is a placeholder for an original object providing it with a new interface. The view schema is closed, i.e. all relationships connect only view classes. Back references keep the connection between a view class and its original classes. This object-generating approach facilitates object polymorphism even when the data model prohibits multiple interfaces for an object. Furthermore, it allows to merge multiple database objects into a single view object, and to import objects over system boundaries.

Further details can be found in [1].

References

[1] R. Busse and P. Fankhauser. Declarative and Procedural Object-Oriented Views. GMD Report 38, GMD – Forschungszentrum Informationstechnik GmbH, St. Augustin, Germany, Oct. 1998.

[2] R. Busse, P. Fankhauser, and E. J. Neuhold. Federated Schemata in ODMG. In *2nd Intl. East/West Database Workshop 1994*, London, 1995. Springer.

[3] G. Gardarin, B. Finance, and P. Fankhauser. Federating Object-Oriented and Relational Databases: The IRO-DB Experience. In *CoopIS'97*, Los Alamitos, 1997. IEEE Computer Society Press.