Issues in the Design and Implementation of Views in Object-Oriented Databases

V. Venugopal
Department of Computer and Information Science,
The Ohio State University,
2036 Neil Avenue Mall,
Columbus, Ohio 43210

Views have traditionally been a mechanism to coordinate access to shared data in a database. View definition schemes in relational databases have aimed at consistency with the conceptual model, and the use of a limited view specification language that allows efficient and unambiguous translation of view updates. These goals are equally applicable to object-oriented databases. Consequently, it makes sense to model views in object-oriented databases as objects themselves.

OODBs differ from relational databases in that they have notions of classes and instances, and allow nesting of both classes and instances. Thus the underlying theory of the object-oriented data model is graph-based, in contrast to the relational theory which is set-based. Object-oriented views can be thought of as graph transformations applied to class or instance graphs in the underlying database. Depending on the nature of the underlying objects, views can be class-lattice views or complex object views. The exact set of graph transformations that can be applied to class or instance graphs need to be limited to allow views to be virtualizable for querying purposes. A language for specifying views as graph transformations is proposed in [2] and is based on the basic idea of disciplined graph rearrangement proposed in [1].

A study of views also provides an opportunity to study the use of object-oriented techniques in the view management process, to study the suitability of forms of reuse and polymorphism other than inheritance such as delegation, and to contrast work on the topic in software development environments and databases. In addition the varying nature and granularity of objects in the graph, raises the question as to whether the same graph transformations should apply to lightweight as well as heavyweight object graphs. The other question pertains to whether read-only views should be allowed to be unrestrained graph rearrangements, or whether they should have greater power than rearrangements to updatable views, while being restricted in some ways. These questions need to be studied with the domain in mind, and a study of the application of views to program slicing and dataflow analysis may hold some answers to these questions for CASE environments.

References