Robust Schema Evolution Strategies

Anders Torvill Bjorvand
Department of Informatics, University of Oslo
P.O. Box 1080 Blindern
N-0360 Oslo, Norway
+ 47 69 88 35 88
torvill@trolldata.no

ABSTRACT

In this paper, we describe two new schema evolution strategies.

Keywords

OODB, Schema evolution, Robust schemas.

1. INTRODUCTION

A common approach in modern database design is to model several different and parallell understandings by parallell and similar replicated structures and by excessive use of versioning/inheritance. This will most often result in a conceptually messed up model produced by coincidental developments. Such a model can be overly complex and hence hard to change. Another way to approach this problem is through explicit multiperspective modelling using views. This is a very direct and elegant way of approaching the problem. Efficiency can also be quite good. The expressiveness will unfortunately suffer due to the static nature of this approach. Only declarative transformations can be encapsulated into this model.

In a scenario where we have a very high number of users and a very high number of different applications, both of these approaches will experience some additional difficulties. We choose to disregard performance and replication issues, and will concentrate on expressiveness and consistency.

2. STATE OF THE ART

The problems with the versioning approach [2] is mainly in the domain of model consistency. There is no easy way to determine if all the different versions are consistent with one another semantically. When the number of versions grow very large, this task gets to be incomprehensible. Another problem is also with range restrictions. If a class/entity has a range restriction and a derived schema needs to loosen up the range restriction, it can't do that. Not all things you inherit are of benefit to you - like in life, you need to break free.

The problems with the view approach [1] is mainly in the domain of expressiveness. With diverse user communities, entities/classes and attributes surely need to be added, and that can't be accomplished by views alone, and new, separate database entities must be added. Since we have no clear concept of version or configuration, the total database will soon become a big mess.

It is our position that the task at hand needs a new approach that grasps the best of these two worlds. We need the expressiveness of the schema versioning approach, but we need the closeness to the original/base model that the view approach offers. The versioning tree in a system with very large number of applications will get so large and with so many branches that consistency will be hard to show.

3. TWO NOVEL APPROACHES

We will investigate two different and novel approaches to this problem. The first one is a variation over the versioning approach where all schema versions are collections of classes/entities that are directly derived/inherited from a class in a main/root schema (only one level of inheritance - no "grandchildren"). We thereby flatten the hierarchy of the schema versioning approach. The number of derivations can be many, but they all share a single source directly. This makes it much easier to check for consistency. We call this approach the robust schema versioning approach.

We are also investigating another, totally different approach called the constructive schema approach. Data storage is basically about storing simple (non-compound) data types in a proper structure. Dealing with schema differences is making proper mapping between different structures put on sets of simple data types.

All of these simple data entities are called properties and they are described by a property_id, property_default_name, property_type (non-compound data type). All instances of these properties have the attributes property_instance_id (a sort of objectid) and property_value. These instances have a one-to-many relation to a class with the following attributes: property_instance_id, class_id, class_version_id, property_override_name. This design ensures that a property value can belong to both several versions of a class, but also to entirely different classes due to different modelling requirements. Let's say that one schema connects a telephone number to a person, but another schema connects several telephone numbers to a person. Both these schemas can trivially be constructed from this structure.

By giving identity to the basic properties, we can construct new schemas at run-time.

4. REFERENCES

[1] Bratsberg, Svein Erik (1992). *Unified Class Evolution by Object-Oriented Views*. Proceedings of the 11th International Conference on the Entity-Relationship Approach (ER), Karlsruhe, Germany. Lecture Notes in Computer Science 645, pp. 423-439. Springer Verlag.

[2] Lautemann, Sven-Eric (1996). *An Introduction to Schema Versioning in OODBMS*. Proceedings of the 7th International Conference on Database and Expert Systems Applications (DEXA), Zürich, Switzerland, September 1996. Workshop Proceedings. IEEE Computer Society Press. pp. 132-139.