AN APPROACH FOR SCHEMA EVOLUTION IN ODMG DATABASES

Cecilia Delgado, José Samos
Universidad de Granada

Manuel Torres
Universidad de Almería

Keywords: Schema Evolution, Object-Oriented Databases, External Schemas, Schema Changes.

Abstract: Schema evolution is the process of applying changes to a schema in a consistent way and propagating these changes to the instances while the database is in operation. However, when a database is shared by many users, updates to the database schema are always difficult. To overcome this problem, in this paper we propose a version mechanism for schema evolution in ODMG databases that preserves old schemas for continued support of existing programs running on the shared database when schema changes are produced. Our approach uses external schema definition techniques and is based on the fact that if a schema change is requested on an external schema, rather than modifying the schema, a new schema, which reflects the semantics of the schema change, is defined.

1 INTRODUCTION

Schema evolution is the process of applying changes to a schema in a consistent way and propagating these changes to the instances while the database is in operation.

However, when a database is shared by many users, updates to the database schema are difficult and often prohibited because there is a risk of making existing application programs inoperative when they run against the modified schema.

To overcome this problem, in this paper we propose a version mechanism for schema evolution in ODMG databases that preserves old schemas for continued support of existing programs running on the shared database. To achieve this, we use external schema definition techniques, particularly the methodology presented in Torres (2002). The basic principle of our approach is that, given a schema change request on an external schema, rather than modifying the schema, a new schema, which reflects the semantics of the schema change, is defined. The new schema is assigned to the user, while the old one is maintained for other application programs.

In our mechanism, unlike in other systems (Kim *et al.*, 1988; Monk *et al.*, 1993), the scope of a schema version (external schema) is not confined to the objects which have been created under this particular version, but rather any object in the database can be accessed and modified through any version, whenever the classes of the objects are included in the external schema associated with it. Thus, each object can be directly shared by different versions of the schema without having to version the object at the instance level. There are two reasons for carrying out schema changes in this way: (i) all objects are associated with a single conceptual schema; (ii) each version of the schema is implemented via an external schema defined on the conceptual schema.

In this work we propose an approach that is not limited to capacity-reducing or capacity-preserving schema changes and which only requires a conventional external schema definition mechanism. Furthermore, to facilitate reutilization and to offer a common definition framework for all schemas, we propose the use of the repository.

This paper is organized as follows: section 2 provides a summary of the ODMG object model, the external schema definition mechanism and other basic concepts; in section 3, a methodology for ODMG schema evolution is proposed; in section 4, a

Delgado C., Samos J. and Torres M. (2004).
AN APPROACH FOR SCHEMA EVOLUTION IN ODMG DATABASES.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 136-141
DOI: 10.5220/0002617901360141

mechanism to propagate schema changes is showed; in section 5, a brief review of the related research is given; in section 6, conclusions are exposed.

2 BACKGROUND

ODMG Object Model. In the ODMG object model, the basic modelling primitives are the *object* and the *literal*. The difference between the two concepts is based on the notion of identity. Objects have a unique identifier whereas literals do not have an identifier. Objects and literals can be categorized by their *types*. The types are classified as *object types*, if their instances are objects, and as *literal types* if their instances are literals. The state of an object is defined by the values it carries for a set of *properties*. These properties can be *attributes* or *relationships* with other objects. The behaviour of an object is defined by the set of operations executed on or by the object. One aspect to be taken into account in the definition of a type is its *external specification*. The specification of an object type (or simply *specification*) can be carried out in two ways: an *interface* is a specification that defines only the abstract behaviour of an object type; a *class* is a specification that defines the abstract behaviour and state of an object type. Objects cannot be created from an interface but must be created from a class.

In the ODMG object model, the inheritance is based on the subtype relationship, and there exist two kinds of inheritance: *behaviour inheritance* (ISA) and *state and behaviour inheritance* (EXTENDS). The ISA relationship is a multiple inheritance relationship between specifications of type that allows us to create more specialized types (subtypes) from existing types (supertypes). This relationship only defines inheritance between the behaviour of object types. The EXTENDS relationship is a simple inheritance relationship between two classes where the subordinate class inherits the state and the behaviour of the class that it is extending.

In the ODMG object model, a schema can be defined as a rooted directed acyclic graph, where the finite set of vertices correspond to the specifications of types (interfaces and classes) and the finite set of directed edges correspond to the inheritance relationships between specifications. The root of the graph is an interface called `Objects`.

A formal definition for the ODMG object model is proposed in Delgado *et al.* (2003).

External Schema Definition Mechanism. Current specifications of the ODMG 3.0 standard (Cattell, 2000) do not include explicit support to define external schemas. However, the possibility of defining external schemas would allow ODMG databases to be adapted to the ANSI/SPARC three-level architecture, which could provide them data independence and functions of integration.

The mechanism proposed in Torres (2002) integrates the derived specifications of type in the repository using the *derivation relationship*, but to avoid extending the object-oriented paradigm, the external schemas do not use this relationship. The definition of an external schema is a three-step process: first, *selection of specifications of type* that will form the external schema; second, *closure of the schema*; in this step it is checked that there are no external references to specifications of types not included in the schema; if there are, the referenced specifications are replaced by derived specifications that eliminate these references; third, *generation of the external schema*; in this step, existing inheritance relationships between specifications of type are obtained. The definition of external schemas is carried out in the repository.

Information Capacity. According to Miller *et al.* (1993), the information capacity of a schema is defined as the set of all valid instances of the schema. Intuitively, a schema has an information capacity that is larger than or equal to that of another if every instance of the first can be mapped onto an instance of the second without loss of information.

Schema Transformations. Schema changes can be implemented by schema transformations (Delgado *et al.,* 2003). A schema transformation is a total mapping between sets of schemas**.** A schema transformation is *information capacity augmenting* if it induces a total, injective mapping between the family of the instances of the original schema and the family of instances of the transformed schema. A schema transformation occurs when it induces a bijection between the family of the instances of the original schema and that of the transformed schema. A schema transformation is *information capacity reducing* if it induces a functional, surjective mapping between the family of the instances of the original schema and that of the transformed schema.

3 A METHODOLOGY FOR SCHEMA EVOLUTION

According to the ANSI/X3/SPARC (1986) schema definition framework, external schemas are derived from the database conceptual schema. Each external schema describes the part of the information of the conceptual schema that is appropriate to the group of users to whom it is addressed.

In the ODMG model, the information contained in a schema is represented by specifications of type. External schemas may include non-derived specifications and derived specifications, directly or indirectly defined on the basis of conceptual schema

specifications (Samos *et al.*, 1997); this means that the information contained in derived specifications is already represented in the conceptual schema.

Derived specifications are defined and included in the *repository*. Specifications from which a derived specification is directly defined are called *base specifications* and they can be derived or non-derived. Each derived specification is related to its base specifications by a *derivation relationship*.

Users sometimes request new information that cannot be derived from the database. When an external schema with non-derived information has to be defined, the conceptual schema must be modified to include this information, and thus the external schema can be derived from the conceptual schema. Adding new information might either require the definition of new non-derived specifications or the modification of previously existing specifications.

Depending on the kind of change, a specification can evolve in two ways: (1) if the change is information capacity-preserving or reducing, then the specification is transformed into a derived specification, independently of whether the specification, before change, was derived or non-derived; (2) if the change is information capacity augmenting, the specification is transformed into a partially derived specification described below.

However, when changes are applied on an external schema, this always evolves to a new external schema, independently of the kind of change. The main difference with respect to the evolution of specifications is that if the change applied on an external schema is information capacity augmenting, the generation of the new external schema cannot be carried out until the conceptual schema has been modified.

3.1 Partially Derived Specifications

Conceptually, a solution for supporting schema evolution by information capacity augmenting transformations consists of allowing derived specifications of type that may contain non-derived information, i.e., *partially derived specifications of type*. The non-derived information is added to the information obtained from the base specifications, so that the need to modify the definition of other specifications is avoided.

The base specifications and the partially derived specifications share the information that may be contained in both specifications. The additional information that cannot be contained in the base specifications will be contained exclusively in the non-derived part of partially derived specifications.

The capability to define partially derived specifications simplifies the execution of operations that represent information capacity augmenting changes in a specification of type.

3.2 Test Environment

In the design process of an external schema, it is possible that new non-derived information may be required, which can be subsequently rejected; this means that a conceptual schema has to be continually modified until a final version of the external schema is achieved.

In order to avoid the continous modification of the conceptual schema, it may be very useful to have a *test environment* in which *provisional external schemas* can be defined. These schemas can include non-derived information without having to modify the conceptual schema.

The transformation of a provisional external schema with non-derived information into an external schema is a two-step process:

1. The conceptual schema is modified in order to include the non-derived information.
2. The external schema is derived from the new conceptual schema; thus the non-derived information contained in the provisional external schema becomes derived information.

The existing external schemas are not affected by the modification of the conceptual schema or, what amounts to the same thing, taking into account the concept of data independence established in the ANSI/X3/SPARC report (1986), conceptually, a change in a schema must not affect other schemas of the database. However, in some cases, the *logic ligature* must be re-established, i.e., it is necessary to repeat the transformation between the external schema and the new conceptual schema.

For example, if an external schema includes a non-derived specification and, as a result of a change in another external schema (e.g., a new property is added to the specification) the conceptual schema is modified, the specification may become derived. In this case, the external schema is not modified, but the ligature between the external schema and the new conceptual schema has to be established; the establishment of this ligature is an updating of the information stored in the repository.

Moreover, if a specification of the conceptual schema, which is the base specification of a derived specification or is included in an external schema, has to be modified, then a new specification is defined and the rest of existing specifications are not affected by this modification. The only thing that can change is its definition, i.e., it may become a derived specification, so that if the specification is no longer included in the conceptual schema it will still remain in the repository.

4 A MECHANISM FOR PROPAGATING CHANGES

When a schema change is requested, it is necessary to determine the impact of such a change on the existing application programs. If the schema change affects existing programs, we have two choices: (i) we can rewrite all affected programs to work with the new schema, or (ii) we can reject the update of the schema. The former would be extremely laborious, while the latter option is not suitable, since it may result in a system not being able to incorporate new information needs.

The mechanism that we propose overcomes these problems. After constructing an initial conceptual schema, each developer defines his own external schema on the shared database and if they subsequently consider a change of schema to be needed, they specify it on their own schema. The database system computes a new external schema, which reflects the semantics of the change; it then replaces the old schema with the new one. Thus, rather than directly modifying the old schema, the schema change is simulated by generating a new external schema. This process is transparent to users, i.e., the simulation of the change is indistinguishable from direct schema modification, since the user perceives a real schema change.

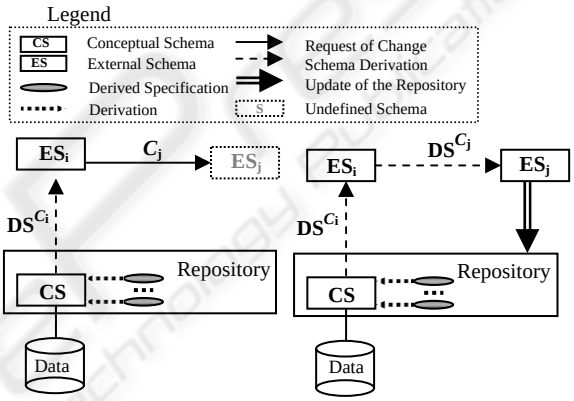
Although the schema change is simulated, sometimes the conceptual schema must be restructured in order to generate an external schema with a different semantics. In this case, the database must be reorganized at the instance level, especially when the change is information capacity-augmenting. When a change to an external schema is required, for example, the addition of a new attribute, the conceptual schema has to be augmented with new stored data. However, when a deletion is requested, this change results in a removal from the external schema (and its associated data), but there is no deletion from the conceptual schema nor any deletion of values from the database.

4.1 Information Capacity-Reducing or Preserving Schema Changes

The essential principle of our approach is that information capacity-reducing or preserving schema changes are achieved by constructing an external schema that reflects the semantics of the change (see Figure 1 for an illustration of the mechanism).

For an information capacity-reducing or preserving schema change C_j, the mechanism computes the change by creating a new external schema ES_j that reflects the intention of the operation. The generation of an external schema from the conceptual schema or from another external schema is called a *schema derivation operation* (DS). Then, if the change C_j is specified on an external schema ES_i (Figure 1.a), the new schema ES_j is obtained as $ES_j = DS^{Cj}(ES_i)$ (Figure 1.b).

The old schema ES_i remains intact for continued support of existing applications, while the new schema ES_i provides a more appropriate interface for new application programs. Moreover, ES_j can be viewed as if it was directly derived from the conceptual schema (CS), i.e., $ES_j = DS^{Cj} \circ DS^{Ci}(CS)$ is the sequence of schema derivations that obtains ES_j from CS, where DS^{Ci} is the schema derivation that generates ES_i from CS. This ensures that all external schemas ES_i and ES_j operate on the same conceptual schema and data.



a. Before schema change b. Application of the change

Figure 1: Mechanism for Capacity-Reducing/Preserving Schema Changes

As a result of an information capacity-reducing or preserving schema change, some new derived specifications can be defined. Therefore, in addition to including the definition of the new external schema in the repository, it will have to include the definition of the new derived specifications (Figure 1.b) since, as mentioned previously and unlike other systems (Ra *et al*., 1997; Rundensteiner *et al*., 1998), the derived specifications are not included in the conceptual schema.

4.2 Information Capacity-Augmenting Schema Changes

Information capacity-augmenting schema changes cannot be carried out in the same way as that of information capacity-reducing or preserving schema changes due to the inherent limitation of the external schema definition mechanism not being able to augment the information capacity of the database. We therefore propose a solution based on a two-step

process that uses the same external schema definition mechanism presented in section 2.

First, the *conceptual schema augmentation* step translates the schema change request C_j specified on the external schema ES_i (Figure 2.a) into an operation C_j' to be executed on the conceptual schema CS in order to restructure CS, denoted by <1> in Figure 2.b. After this step is executed, the conceptual schema CS no longer exists in its initial form, but has been modified by augmentation with information that is possibly irrelevant to many applications. For this reason, the schema derivations specified upon it may have become undefined (DS^{Ci} in Figure 2.b), although the external schema ES_i still exists. Furthermore, the data is associated with the new schema CS' instead of with the former CS.

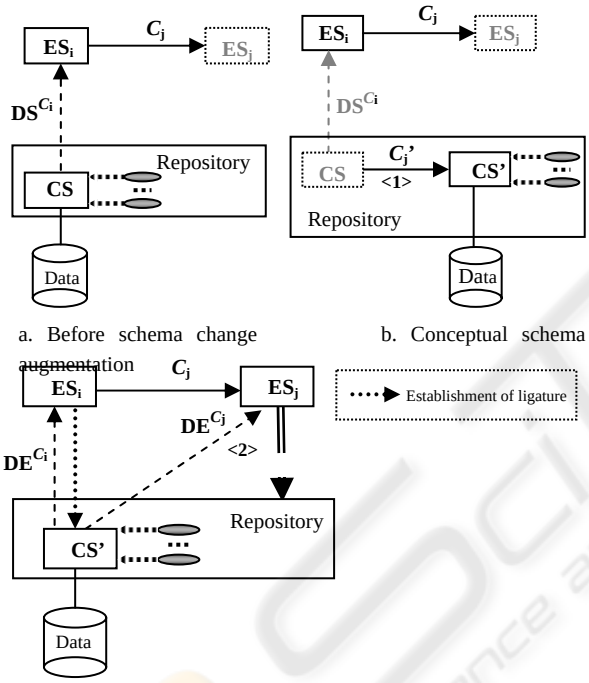


Figure 2: Mechanism for Capacity-Augmenting Schema Changes

Second, the *new external schema generation* step, denoted by <2> in Figure 2.c, generates the new schema ES_j as if the change C_j had been performed on the external schema ES_j directly. Given that the information capacity of the conceptual schema CS has been appropriately augmented by the first step, the schema derivation can always be carried out.

Since the original conceptual schema CS no longer exists, for continuing support of all the existing external schemas (and thus all applications) defined on CS, in this second step, all schema derivations that had become undefined in the previous step are restored on CS' and new logic ligatures are established. As a result of the change, new derived specifications can be defined or non-derived specifications may become derived; therefore, in addition to including the definition of the new external schema in the repository, it will have to be updated to add the definition of the new derived specifications or to modify existing definitions (Figure 2.c).

4.3 Characteristics of the Mechanism

The fundamental characteristics of our mechanism are the following:

1. The schema change applied to an external schema *does not affect to any of the existing external schemas*. This ensures that earlier applications can still work properly. Besides, this feature allows users to restructure their external schema independently from other users working on the same conceptual schema.
2. Both old and new versions of the schema are able to *share the same* (persistent) *data*, independently of the version with which the data was originally created.
3. Schema changes are *transparent* to the external schema users, since the fact that the schema change is simulated is not apparent to them.

Taking into account these characteristics, the mechanism presents the following advantages:

1. *It allows all instances to be directly shared by all schemas*. Maintaining separate copies of the instances makes it difficult and expensive to propagate the update of an instance of one schema version to all copies of the instance under other schema versions. In our mechanism, because all instances are associated with a single conceptual schema, instances are directly shared by all versions of the schema, i.e., there exists only one copy of each instance.
2. *It integrates the restructuring power of external schemas and schema evolution*. This provides many advantages, because external schemas enable programs to use data representations adapted to their needs and the evolution of the schema makes it possible to incorporate new data requested by applications.
3. *Version merging*. The version merging process is very complicated because all instance versions with the same object identity must be merged into a single instance, and schema versions must be combined into one consistent schema. This process can be easily solved in our mechanism, because instances of different external schemas are never duplicated, and all the specifications derived are defined in the repository.

5 RELATED WORK

In Tresch *et al.* (1993) it is stated that schema changes can be simulated using external schemas (views) if they are not information capacity augmenting. In this work, some examples are presented and it is mentioned that for information capacity-augmenting schema changes the meta-database must be dynamically accessible at run time.

Bertino (1992) presents a view mechanism that can be used to simulate schema evolution. The mechanism proposed is information capacity-augmenting in the sense that new stored attributes are added to views. However, the paper focuses on the evolution of individual classes rather than on schemas.

In Ra *et al*. (1997), an extension of conventional view definition mechanisms to simulate information capacity-augmenting schema changes is proposed. This mechanism, unlike that of Bertino, is directed toward schema evolution. However, commercial object-oriented database systems do not support such information capacity-augmenting views.

In Rundensteiner *et al*. (1998), a methodology to achieve transparent schema changes and that does not require information capacity-augmenting views is presented. In this methodology, when it is necessary to modify the base schema because an information capacity-augmenting schema change is required, the former base schema is reconstructed as a view schema from the augmented base schema in order to continue supporting all the existing view schemas. However, we believe this reconstruction to be unnecessary, since earlier view schemas can be derived from the new base schema.

Some version systems (Kim *et al*., 1988; Monk *et al*., 1993; Lautemann, 1997) build new versions of schemas and instances, so that every instance of the old schema is copied and converted into an instance of the new schema version. The main differences between these systems and our proposal are: (i) duplicates of the objects are created, which produces serious limitations when modifications are propagated; (ii) updates via newer schema versions are often not propagated to older versions; (iii) data inconsistencies between different versions of an object may arise over time. For these reasons, true interoperability between old and new applications is not achieved.

6 CONCLUSIONS

In this paper, we present a solution for the problem of schema evolution in ODMG databases based on external schema definition techniques. The basic principle of our solution is that when a change to an external schema is requested, instead of modifying the schema, a new one is defined, reflecting the semantics of the schema change. The new schema is assigned to the user, while the old one is maintained for other application programs.

Moreover, in order to avoid the continuous modification of the conceptual schema, when information capacity-augmenting schema changes are requested, we propose the use of a test environment in which provisional external schemas can be defined. These schemas can include non-derived information without having to modify the conceptual schema. If necessary, the conceptual schema will be modified when the provisional external schema is accepted by the users.

ACKNOWLEDGMENTS

This work has been supported by the CICYT under project TIC2000-1723-C02-02.

REFERENCES

ANSI/X3/SPARC, 1986. Database System Study Group, "Reference Model for DBMS Standardisation". *SIGMOD RECORD, 5*, pp.19-58.

Bertino E., 1992. A View Mechanism for Object-Oriented Databases. In *EDBT'92, LNCS 580*, pp136-151.

Cattell R.G.G, 2000. *The Object Data Standard: ODMG 3.0*. Morgan Kaufmann.

Delgado C. *et al.*, 2003. Primitive Operations for Schema Evolution in ODMG Database. *OOIS'03, LNCS 2817*, pp. 326-337.

Kim W. and Chou H., 1988. Versions of Schema for OODBs. In *VLDB'88*, pp. 148-159.

Lautemann S.-E, 1997. A Propagation Mechanism for Populated Schema Versions. In *IEEE Int. Conf. on Data Engineering*, pp. 67-78.

Miller, R. *et al.*, 1993. The Use of Information Capacity in Schema Integration and Translation. In *VLDB'93*, pp. 120-133.

Monk S. and Sommerville I., 1993. Schema Evolution in OODBs Using Class Versioning. In *SIGMOD RECORD (22)*, pp. 16-22.

Ra Y.G. and Rundensteiner E.A., 1997. A Transparent Schema Evolution System Based on Object-Oriented View Technology. In *IEEE TKDE*, pp. 600-624.

Rundensteiner E. *et al.*, 1998. Capacity-Augmenting Schema Changes on Object-Oriented Databases. In *OOIS'98*, pp. 349-366.

Samos J. and Saltor F., 1997. External Schemas in a Schema-Evolution Enviroment for OODBs. In *DEXA'97*, pp. 516-522.

Torres M., 2002. Definición de Esquemas Externos en Bases de Datos ODMG. PhD Thesis, Univ. of Almería, Spain.

Tresch M. and Scholl M.H., 1993. Schema Transformation without Database Reorganization. In *SIGMOD RECORD*, pp. 21-27.