# A self-adaptive semantic schema mechanism for multimedia databases

Jun Yang [a,b], Qing Li [a], Yueting Zhuang [b]
[a] City University of Hong Kong, {itjyang, itqli}@cityu.edu.hk
[b] Zhejiang University, Hangzhou, China, yzhuang@cs.zju.edu.cn

## ABSTRACT

In the context of multimedia retrieval, the goal of accuracy is to a certain extent contradictory with that of efficiency. The former relies on exploiting sophisticated features, whereas the latter favors using simple features with reduced dimensionality. As an endeavor to strike the balance between these two goals, this paper presents a self-adaptive semantic schema mechanism (SSM) for multimedia databases. The SSM is implemented based on an object-oriented data model, with classes being organized into a semantic hierarchy. As its most distinguishable feature, when the conditions of certain ECA-rules are satisfied, SSM supports adaptive evolution of a schema in the form of expansion with new classes and/or compaction by removing inefficient ones. This self-adaptive evolution strategy allows a schema to optimize for the requirements of each specific application, thereby achieving a dynamic, application-specific balance between accuracy and efficiency. A prototype system for multimedia retrieval, 2M2Net, has been built based on this mechanism and validated for its feasibility.

**Keywords:** multimedia database, semantic schema mechanism, self-adaptive, schema evolution strategy, ECA-rules

## 1. INTRODUCTION

In recent years, the Computer Vision community and the Database community are working towards the common goal of multimedia retrieval. Nevertheless, the specific interests of these two communities, as well as their fundamental approaches, are significantly divergent, or to a certain extent mutually exclusive. The primary concern of the Computer Vision researchers is high retrieval accuracy, usually defined in terms of precision and recall. To this end, a great number of sophisticated data features, which are proven to be more descriptive and discriminative than simple ones, have been extensively used in retrieval approaches. The efficiency issue, however, is seldom within their consideration. Most multimedia systems from this community, to the knowledge of the authors, employ many "non-database" approaches, such as using file system for storage, conducting sequential search, etc.

Even if enough attention is paid to the efficiency issue, it is unlikely to be well addressed, since the sophisticated features for multimedia data processing pose the real obstacle of scalability. While the dimension of features can be as high as a few hundreds (e.g. the renowned QBIC system [2] uses at least a 256-D color histogram for image retrieval), the most powerful multi-dimensional indexing techniques, such as R-tree and its variants, are not scalable to dimension higher than 20. Keyword features are not less sophisticated in this regard, since the dimension of a keyword vector virtually equals to the size of the vocabulary. This sharp contrast implies that the search space cannot be effectively partitioned, and in the worst cases, the search is degraded to a sequential scan [18]. The drawback of inefficiency, which is currently somewhat underestimated due to the use of small- or medium-scale databases, will become extremely critical when dealing with large-scale data collections.

The Database people, on the other hand, are mainly concerned with the efficiency aspect of a multimedia system. For this purpose, they have investigated a broad range of techniques regarding storage (e.g. [3]), data modeling (e.g. [11]) and multi-dimensional indexing (e.g. [6][12]) for multimedia, all of which unanimously suggest the use of simple, uniform, and low-dimensional features that allow economical and efficient storage, indexing and access. The feature effectiveness in terms of retrieval accuracy, however, is in most cases taken for granted due to the traditional way of exact matching in conventional databases.

The analysis presented above gives insight into the relationship between accuracy and efficiency with regard to multimedia retrieval, which is to a great extent a tradeoff, reflected in the choices between sophisticated (but inefficient) features and simple (but less powerful) ones. On the other hand, both issues are so essential to a multimedia system, that

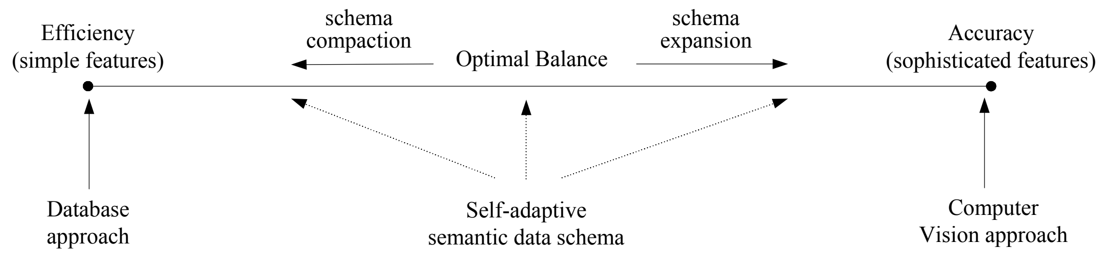

Fig.1: Relationship between self-adaptive semantic schema and conventional approaches

failure to address either of them will make the system impractical. Therefore, a fundamental question any multimedia system designer has to confront is: *what is the optimal balance point between the two extremes?* Unfortunately, to this question there does not seem to exist a uniform answer that can be applied to all the applications, nor does there exist a fixed answer even for a specific application. Rather, the "optimal" balance point should be tuned based on the characteristics of a specific application from time to time (i.e., snapshots of the application), mainly in the following aspects:

1. **Data volume.** Greater amount of data naturally imply more expensive storage and computation. Consequently, applications manipulating of a large volume of data are inclined to use highly efficient features, and vice versa.

2. **Distribution of data**. Data that are evenly scattered in the feature space can be easily distinguished through simple and coarse features. By contrast, if the data are concentrated in a dense region of the feature space, more discriminative and sophisticated features become indispensable in order to differentiate among them.

3. **Requirement on retrieval**. Some applications demand very precise retrieval results but can accept a moderately long time for processing (e.g. image-based automated medical diagnosis system), while others prefer queries to be responded quickly but can tolerate certain level of impreciseness (e.g. a Web-based image search engine). Obviously, the features for the applications of the first type must be sufficiently rich and powerful, while those for the second type must be efficient to access and process.

As illustrated in Fig.1, the self-adaptive semantic schema mechanism (SSM) for multimedia database, the main contribution of this paper, is a "middle way" approach between the conventional Computer Vision approaches and the Database approaches. The SSM is implemented based on an object-oriented data model, with classes being organized into a semantic hierarchy. Driven by predefined ECA-rules, the SSM supports adaptive evolution of a schema in the form of either expansion by including requested classes or compaction by removing inefficient ones. In this way, it optimizes the schema for the requirements of each application, thereby achieving a dynamic, application-specific optimal balance between accuracy and efficiency with regard to multimedia retrieval.

The rest of the paper is organized as follows. In Section 2, we present a brief review of the related techniques that contribute to our work. The self-adaptive semantic schema mechanism is proposed in two steps. Firstly, we define a schema template and describe its construction approach in Section 3. By following this template, self-adaptive schemata can be constructed in the second step using a self-adaptive schema evolution strategy. This strategy, interpreted in the form of ECA-rules governing the evolution process, is elaborated in Section 4, along with a case study demonstrating how this strategy works in an application scenario. We introduce a prototype system built upon the proposed mechanism in Section5. Summary and future works are given in the last section.

## 2. RELATED WORK

The work presented in this paper can be regarded as an effort to address the key issues of multimedia retrieval from a database perspective. For this purpose, we investigate the advantages of several database techniques, including object-oriented modeling, schema evolution with object migration, and active capability via ECA-rules. In this section, we present a brief review of these relevant techniques.

Multimedia data in the form of image, video, and audio possess properties that are not adequately supported by traditional database systems, such as huge data size, time-dependent nature, content-based retrieval, etc. To address these limitations, multimedia database has been proposed and received an extensive study on its related techniques during the last decade. The objective is to provide reliable and efficient storage, maintenance, and access of different types of media. Substantial modifications and extensions can be found in both relational databases (e.g. STARBURST system [4]) and object-oriented databases (e.g. [10][12]), as an effort to accommodate “multimedia features”. Some emerging multimedia systems, such as QBIC [2] and Informedia [5], manifest certain characteristics of a multimedia database. Nevertheless, even till now there is no widely accepted definition of multimedia database and its architecture, and many relevant problems have not been successfully addressed.

Object-oriented data modeling is an established technique developed to meet the advanced modeling requirements of complex applications. In an object-oriented model, each real-world entity is modeled as an *object* (alternatively called as an *instance*) of a certain *class*, which can have some *attributes* (properties) and *methods* (operations) that are applicable to its objects. The fundamental modeling techniques of an object data model include *instantiation* (creating an instance of a class), *inheritance* (defining a new class by inheriting attributes and methods from other classes), and *aggregation* (a class subsuming other classes as its components). An object-oriented model has advantages over its relational counterpart in modeling capability of complex objects, type reusability, economical schema evolution, polymorphism, and extensibility, etc. This can account for the general understanding that an object-oriented model, amongst other data models, is most suitable to be used for a multimedia database [1][10][12].

Active databases, as opposed to passive ones, are the database systems augmented with reactive behaviors that allow them to respond independently to certain events. Typically this reactive behavior is implemented in the form of either "triggers" or *event-condition-action* (ECA) rules[1]. A survey on active databases has been conducted in [14].

As will be seen later, we exploit the power of the above database techniques in a synergic manner to propose a self-adaptive semantic data schema for multimedia databases. Notice that we do not intend to (also impossible to) address, in the scope of this paper, all the open issues in the area of multimedia databases. Our main interest resides in the retrieval aspect, especially in how to get a good tradeoff between accuracy and efficiency through mechanisms enforceable by the database system.

## 3. SCHEMA TEMPLATE

As the first step towards the self-adaptive schema, a schema template is created, which from definition is exactly an object-oriented schema. However, it is not a schema by itself, since it is never stored in the database catalog or used for data typing in real applications. Rather, it provides the background knowledge (template) based on which the self-adaptive schema is constructed in a later stage. In this section, we firstly describe the template from an object-oriented perspective, and then suggest a method to create the template automatically from a lexical thesaurus.

### 3.1 Template Definition

As illustrated in Fig.2(a), the schema template is defined based on an object-oriented data model, with classes organized into a hierarchical structure. The link between two classes denotes the inheritance relationship between them, by means of which a class (*subclass*) can inherit common attributes and methods from a more generalized class (*superclass*). Each class can also define local attributes and methods that do not appear in its superclass. As shown in Fig.2(b), the class *Bird* inherits from its superclass *Animal* attributes *Habitat*, *Age*, *ScientificName,* and method *SimilarTo*. It also defines new attribute *FeatherColor* and method *AbleToFly*, which are unique of its semantics. The template is constructed by recursively using subclasses to derive more specialized classes, and therefore, it is in essence an *inheritance* (or *"is-a"*) *hierarchy* of an object-oriented data model.

[1] The meaning of an ECA rule states that “when a specified event occurs, evaluate the given condition and if it holds true, execute the action” [14].

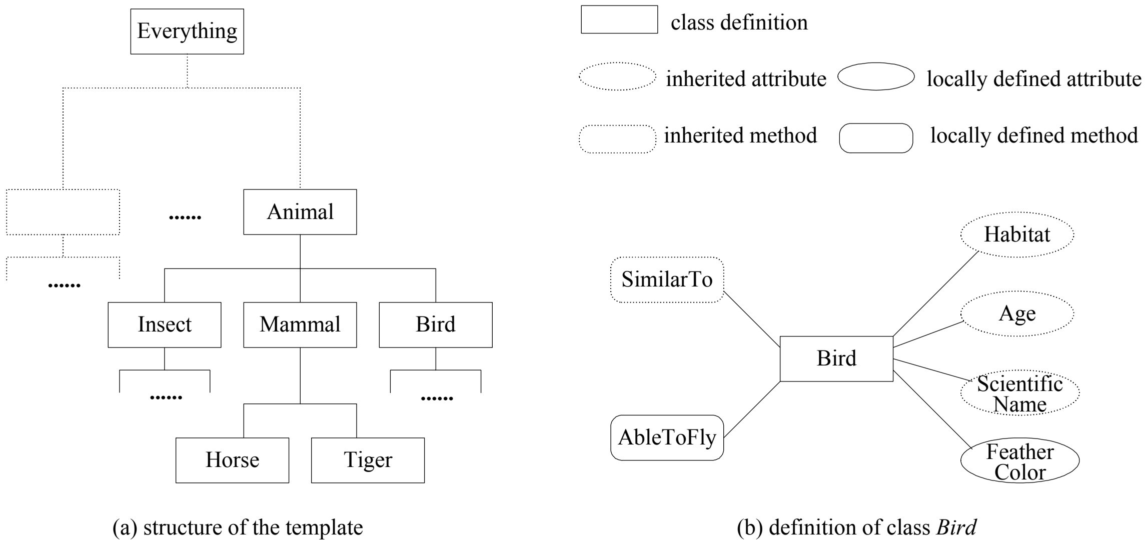

Fig.2: Definition of the schema template

The schema template can be also interpreted from a semantic perspective: each class in the schema corresponds to a *semantic concept*, and the inheritance relationship indicates that one concept is a specialization of another. Viewed as a whole, the template starts with generic concepts at higher layers and extends downwards to more specialized concepts at lower layers, which forms an intuitive classification (or taxonomy) that captures the human being's commonsense. In fact, what is revealed in Fig.2 (a) is only a small fraction of the whole template, which, starting with root class *Everything*, is supposed to cover all the semantic concepts that are assignable to media objects.

In practice, media objects of various types, such as still images, graphics, video clips, audios, are attached to different classes in the template (or the schema created from the template) as their instances, like the decorations hung on a Christmas tree. For each data type, there exists a foundation class (not shown explicitly in Fig.2(a)), such as *Image*, *Graphic*, or *Video*, which is named as *type class* as opposed to *semantic class* that is shown explicitly in Fig.2(a). Each media object must be at the same time an instance of a semantic class as well as of a type class, e.g., an image showing a horse is an instance of both class *Horse* and class *Image*. This lends considerable flexibility and convenience to the programming task, since each object can inherit attributes and methods that are characteristic either of its data type or of its semantic meaning. However, currently such multiple inheritances exist only between a semantic class and a type class, rather than between two semantic classes. Put it in another way, multiple inheritance is not allowed in the scope of the hierarchy shown in Fig.2(a), which is therefore a tree structure in the strict sense.

In our preliminary implementation, aggregation relationship between classes is not currently included in the schema, i.e., a class can neither subsume nor refer to other classes in its definition. Although this admittedly undermines the modeling capacity of the schema for composite objects, it can be explained in the context of multimedia retrieval: each media object (e.g., a photograph) is an integral semantic unit which makes sense to users only when it is retrieved as a whole instead of as bits-and-pieces components.

By definition, there is no difference between the schema template and an object-oriented schema. In practice, however, the template is only used as guidelines for the definition and organization of classes, which are followed by the self-adaptive schema during its construction and evolution. The template is stored as background knowledge in some places other than the database catalog.

## 3.2 Template Construction Approach

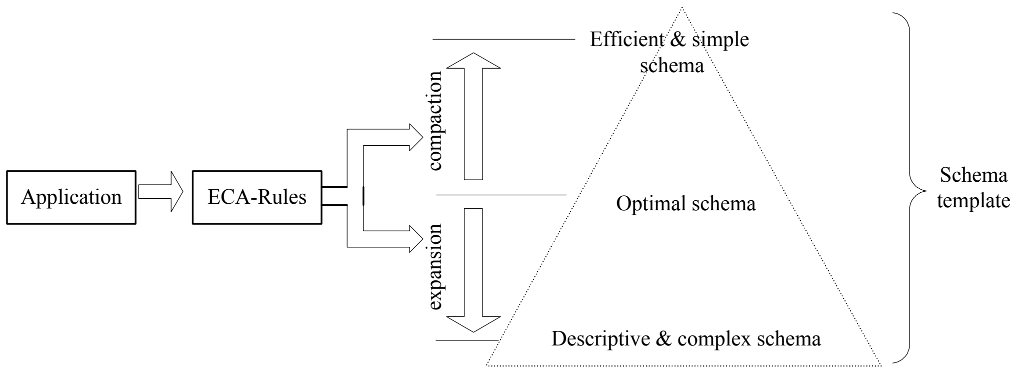

Fig.3: Self-adaptive schema evolution strategy

It is naturally followed by the question of how to define the structure of the schema template, i.e., why we organize classes in such a structure rather than another, which is actually the issue of semantic modeling. In many applications, the structure of data schema exists as a kind of ontology, which can be acquired by consulting domain knowledge. To avoid this time-consuming process, in our approach a lexical thesaurus, WordNet, is utilized to construct the structure of the schema template.

WordNet [8] is an electronic thesaurus that is organized around the concept of synset as a class of closely related words with the same sense. There exist various types of semantic links among synsets, which interconnect them into a huge network. The noun portion of WordNet, in particular, has two major types of semantic links between synsets: hypernym/hyponym ("*is_a*") relationship, and meronym/holonym ("*is_part_of*") relationship. If only the first type is taken into account, all the noun synsets constitute a hierarchical structure, which is otherwise an acylic graph.

In our work, we map the hierarchy embodied in WordNet to the hierarchical structure of the template, with each synset corresponding to a class. This mapping can be explained based on the following observations: firstly, the hierarchy of WordNet is so comprehensive that it covers all nouns; secondly, the definition of synset is compatible with that of a class in that both of them represent a semantic concept; last but not the least, the WordNet hierarchy is built upon the "*is_a*" relationship, which corresponds to the inheritance relationship among classes. Some adjustments are performed during the mapping of the WordNet hierarchy to the template, e.g. pruning out some "big words" that rarely appear in ordinary documents. Interested readers can refer to [13] and [15] for further details of this mapping.

## 4. SELF-ADAPTIVE SCHEMA EVOLUTION STRATEGY

A key observation made on the schema template (or schemas with the similar structure) is that, the complexity of a schema, defined as the number of classes in it, is closely related to the tradeoff between accuracy and efficiency discussed in Section 1. When the schema is complicated, it contains a vast number of classes, most of which are specialized classes at the bottom of the inheritance hierarchy. As a result, there are more specialized and consequently more descriptive attributes and methods available for modeling multimedia data. In this case, however, there will be a substantial maintenance cost for an increasing number of classes in the database catalog, and the efficiency of query processing will also decline due to the involvement of more classes and longer inheritance path. On the contrary, if the schema is simple and has only a few generic classes, efficiency will definitively improve. However, the efficiency yield comes at the cost of descriptive power, since in a simple schema specialized objects have to be fit into generic classes with coarse attributes (e.g. a *Parrot* object having only the attributes and methods as an *Animal* object).

Therefore, the tradeoff between accuracy and efficiency can be interpreted as the tradeoff between a descriptive schema and an efficient one. The optimal schema in this tradeoff, like the optimal balance discussed in Section 1, is again determined by the requirement of a specific application. Applications manipulating large-scale, widely distributed data would prefer efficient and simple schema, and vice versa. Within each schema, applications focusing on a specific domain require the corresponding part of the schema to be fully developed, but does not need other parts. Typically, it is

**Notation**:
$S$ -- self-adaptive schema     $C$ -- class to be inserted or deleted     $C_p$, $C_q$ -- variables that can used to denote any class

**Procedure:**

```
Step 1: Set $C_p$ to $C$
        while $C_p$ does not exist in $S$ then
            Set $C_p$ to the superclass of $C_p$
        wend
Step 2: Set $\{C_1,\ldots,C_M\}$ as the children nodes of $C_p$
        for $i=1$ to $M$
            Set $C_q$ as the lowest common ancestor class of $C_i$ and $C$
            if $C_q = C$, then go to Step 4
            if $C_q <> C_p$, then go to Step 5
        end for
Step 3: Insert $C$ into $S$ as a child node of $C_p$
        exit
Step 4: Insert $C$ into $S$ as a child node of $C_p$ and parent node of $C$
        exit
Step5: Insert $C_q$ into $S$ as a child node of $C_p$
       Insert $C$ into $S$ as a child node of $C_q$
       Reconnect $C_i$ as a child node of $C_q$
       exit
```

(a) Schema expansion with class $C$

**Pre-Processing:**

```
Set $C_p$ to the parent node of $C$
Migrate all the instances of $C$ to $C_p$
```

**Procedure:**

```
Step 1: if $C$ is not a leaf node in $S$ then
            Set $\{C_1,\ldots,C_M\}$ as the children nodes of $C$
            for $i=1$ to $M$
                Connect $C_i$ to $C_p$ as its child node
            end for
        end if
Step2: Remove $C$ from $S$
```

(b) Schema compaction by removing class $C$

Fig.4: Schema evolution algorithms

the responsibility of the schema designer to recognize such requirements and optimize the schema accordingly. As an alternative approach, the self-adaptive schema evolution strategy allows an application to autonomously adapt (expand or compact) its underlying schema to its requirement through a set of predefined ECA-rules, without the need for any human supervision and intervention. The basic principle of the strategy is clearly illustrated in Fig.3, which is another version of Fig.1 interpreted in terms of schema complexity. In this section, we start by describing the schema expansion and compaction algorithm, proceed with formulating the ECA-rules that govern the evolution, and end up with a case study demonstrating the functioning of the strategy.

## 4.1 Schema Expansion and Compaction Algorithms

The self-adaptive semantic schema is defined based on the schema template by introducing the self-adaptive schema evolution strategy. In the initial state, this schema has only the root class (i.e. *Everything*) or several generic classes available; then, class definitions can be dynamically inserted into or removed from the schema, implemented as schema evolution in the form of expansion or compaction. Both schema expansion and compaction must follow the structure of the schema template, i.e., the self-adaptive schema conforms to the template in terms of structure. In fact, the self-adaptive schema is a concise and dynamic subset of the schema template, tailored to a specific application. Its conciseness is guaranteed by the evolution strategy, which only inserts requested (by the application) classes and regularly gets ride of inefficient classes.

The algorithms[2] for schema evolution are shown in Fig.4. The expansion algorithm (Fig.4 (a)) focuses on seeking for the appropriate position in the current schema to insert the new class. To illustrate this complicated algorithm, we

[2] In order to differentiate between the *self-adaptive schema* and *schema template* contexts, we use different notations when referring to the relationships between classes in the algorithm and the rest of the paper: keywords "superclass", "subclass", "ancestor class" and "descendent class" refer to the relationship between classes in the schema template, while "parent node/class" and "child node/class" are used in the self-adaptive schema as the equivalent of superclass and subclass.

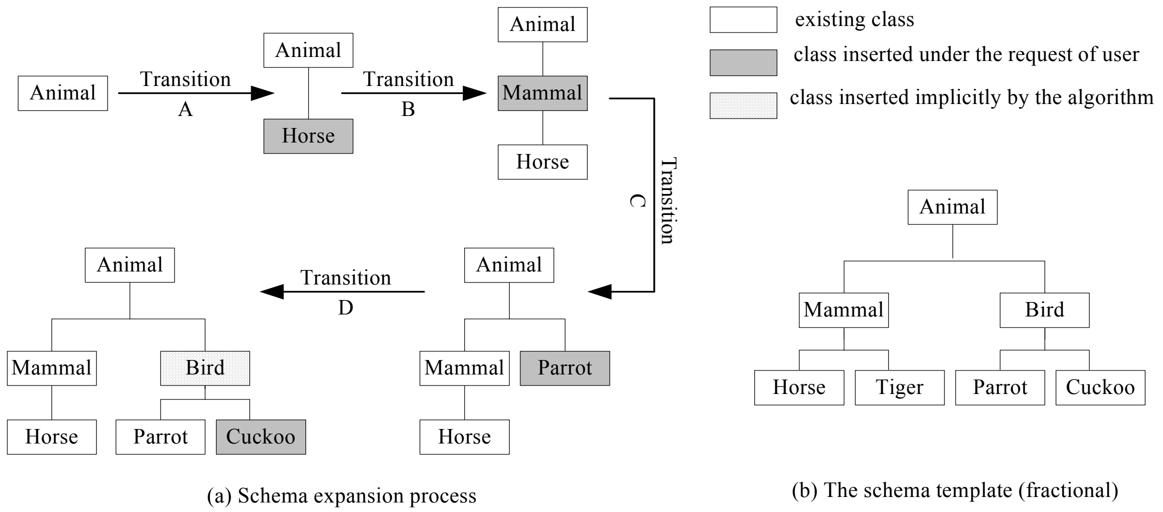

Fig.5. Example of schema expansion

provide an example of schema expansion process in Fig.5. Suppose we use the same notation as in the algorithm. In *Step 1*, we locate class *C*'s nearest ancestor class that already exists in *S* and denote it as $C_p$. Then it comes with four possibilities, each of which corresponds to one of the four transitions indicated in Fig.5(a). First, if $C_p$ has no child node, we go to *Step 3* to insert *C* into *S* as a child node of $C_p$, as exemplified by Transition A in Fig.(b)); Second, if one of $C_p$'s child node $C_i$ is a descendent class of *C*, *Step 4* is performed to insert *C* between $C_p$ and $C_i$ (Transition B); In the third case, if every child node of $C_p$ is in different sub-trees under $C_p$ (defined in the context of the schema template) with *C*, we insert *C* as a child node of $C_p$ in *Step 3* (Transition C); In the last possibility, one of $C_p$'s child node $C_i$ and *C* are in the same sub-tree under $C_p$, we execute *Step 5* to insert the root of this sub-tree into *S* as a child node of $C_p$ , and link *C* and $C_i$ as its children nodes (Transition D). Note that in this case, an additional class other than the user requested one is inserted in order to balance the structure of the schema. For all the four cases, if the inserted class is not a subclass of its parent node (but must be its descendent class), we convert its attributes and methods that are implicitly inherited along the inheritance path (below its parent node) into explicitly defined attributes and methods.

Compared with expansion, schema compaction seems to be much more straightforward, which simply removes the class and reconnect its children nodes (if any) to its parent. However, since there is a possibility that a class is deleted even when it has several native objects (c.f. Section 0), housekeeping work is necessary to fit these objects into other data types before their original class is removed, otherwise they will be purged with the original class. In this case, we apply the *object migration* technique [7] to migrate these objects to the parent node (denoted as $C_p$) of their original class (denoted as *C*). During migration, the object identity of a migrating object is preserved, and the values of the attributes that are inherited from $C_p$ are also retained. However, the attributes that locally defined in class *C* can no longer exist, since such attributes cannot be fit into the definition of $C_p$. Moreover, the objects that are migrated to $C_p$ will become instances of $C_p$, and completely "forget" their original class membership. As a result, even if class *C* is reconstructed sometime later, there is no way for these objects to be reclaimed by *C*.

The compaction algorithm, although resulting in a more concise schema, causes the loss of information of the migrating objects, in terms of their locally defined attributes and their class membership. However, there is no cost-effective solution to either of these two drawbacks. On the one hand, the locally defined attributes cannot propagate to the superclass (or parent class); otherwise, the schema consistency is in risk if the superclass has other subclasses besides the one being removed. On the other hand, attempts made to memorize the original membership of a migrating object might open up more problems, especially when the migration is transitive. Such information loss, which implies the loss of descriptive power of the schema, is the cost of efficiency.

| **Rule 1:** (for schema expansion) | **Rule 2:** (for schema compaction) |
|---|---|
| **Event 1**:<br>A request is received to create an object *O* of class *C*.<br>**Condition 1**:<br>*C* does not exist in the current schema *S*.<br>**Action 1**:<br>Execute the schema expansion algorithm to insert *C* into *S*. | **Event 2**:<br>A certain time interval is elapsed.<br>**Condition 2**:<br>**2.1**: Class *C* is a leaf node.<br>**2.2**: The objects of *C* have not been manipulated during the last time interval.<br>**2.3**: The ratio of *C*'s cardinality against the cardinality of *C*'s parent class is below a threshold.<br>**Action 2**:<br>Execute the schema compaction algorithm to remove *C* from the current schema *S*. |

Fig.6: Examples of ECA-rules

### 4.2 ECA-Rule Driven Schema Evolution

Schema evolution in our approach is not executed under the explicit requests of applications. Instead, it is controlled by a set of ECA-rules, i.e., it is triggered when the condition holds true on occurrence of certain events. The ECA-rules are defined by the system designers, which reflect the requirement of a specific application. Although the practical ECA-rules can be very sophisticated and application-specific, we suggest two general and exemplary ECA-rules in Fig.6 to convey the basic idea of the strategy.

The event that may induce schema expansion is the request to create an object *O* of a certain class *C*. In this case, if the *C* already exists in the current schema *S*, we create *O* as an instance of *C*; otherwise, we execute the schema expansion algorithm (c.f. Fig.4 (a)) to insert *C* into the schema, and then create O as an instance of *C*. On the other hand, we consider schema compaction at certain time interval with predefined length (which can be as long as days). The condition part for class *C* is the "intersection" of three conditions: (1) *C* is a leaf node, (2) objects of *C* have not been manipulated (created, modified, accessed, deleted) in the last time interval, and (3) the ratio of *C*'s cardinality against the cardinality of *C*'s parent class is below a threshold. (Cardinality of a class is defined as the total number of objects belonging to this class and all its descendent classes.) If all conditions hold true, the schema compaction algorithm (c.f. Fig.4 (b)) is called to remove *C* from the schema with its objects migrated to its parent class.

The aim of the exemplary ECA-rules is to maintain a concise set of classes in the schema with adequate descriptive power, which is tailored to a specific application. For this purpose, schema expansion is executed in an "on demand" manner, which augments the descriptive power of the schema by adding new classes requested by the application. Schema compaction, on the other hand, is much like a "garbage collection" process performed regularly to remove the inefficient classes, whose maintenance cost cannot be justified by its number of objects and frequency of access. It can be expected that even more sophisticated ECA-rules will serve the same purpose, i.e., tailoring the schema towards the requirements of the application.

Admittedly, ECA-rules entails significant performance overhead, which may partially offset the efficiency gained from the strategy. However, due to the following reasons, we argue that using ECA-rules does bring more benefit that justifies its cost. First of all, the ECA-rules approach automates the schema evolution process, which relieves considerable human effort of monitoring database status and performing evolution. Such a task can be even impossible for database administrators to accomplish when the database contains a large volume of data and accepts highly frequent access requests. As another consequence of automation, our approach is free of subject errors, which may become as serious as deleting necessary classes and objects. Further, the ECA-rules enforce the schema to conform to the semantically structured template, which may otherwise be disordered by supervised evolution. Compared with application-level mechanisms, ECA-rule has a broader access and greater control of the database, which allows the implementation of more sophisticated and powerful rules.

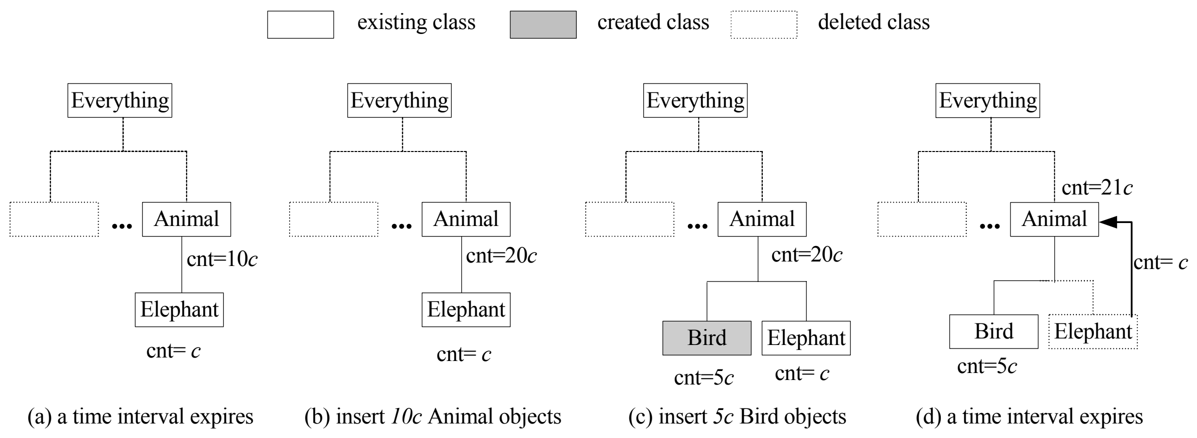
existing class
created class
deleted class
Everything
Animal
Elephant
Bird
cnt=10c
cnt= c
cnt=20c
cnt=5c
cnt=21c
(a) a time interval expires
(b) insert 10c Animal objects
(c) insert 5c Bird objects
(d) a time interval expires

Fig.7: A Case Study

Since the particular object-oriented database management system (OODBMS) currently in use does not support active capability, the ECA-rules are encoded using the persistent object programming language provided by the database system as its additional function module. Rule management (creation, deletion, and modification) is not realized in this preliminary implementation. There are also other critical issues related to ECA-rules [14], such as consistency, conflict resolution, termination problem, which are not discussed in the scope of this paper.

## 4.3 A Case Study

In this subsection, we present a simple case study to demonstrate the functioning of the strategy in the scenario of a multimedia application, which focuses on managing animal pictures. Commonly, the application requests to insert objects into or delete them from the underlying database during its operation. On occurrence of each request (Fig.7(b) and (c)), as well as the expiration of a time interval (Fig.7(a) and (d)), the corresponding ECA-rules are evaluated, and the appropriate schema evolution is performed when the condition is true. In the following, we go through the four steps in Fig.7 with an analysis of the rule processing on each step[3]. We assume the threshold employed in *Condition 2.3* is set to 0.05.

(a) As the triggering event of *Rule 2*, expiration of a time interval causes *C2* to be evaluated for both class *Animal* and *Elephant*. At this moment, *C2.1* is false for the non-leaf class *Animal*, while *C2.3* is false for *Elephant* whose cardinality ratio is above the threshold. Therefore, no action is performed in this stage.

(b) Creation of *Animal* objects leads up to the occurrence of *E1*. Consequently, we check *C1* for class *Animal*, which holds false in this case since *Animal* is already in the schema. Again, no action is taken, except the requested objects are created as instances of class *Animal*.

(c) At the moment when objects of *Bird* are to be created, *Bird* is absent from the schema, which satisfies *C1*. Consequently, *A1* is executed to insert the class definition of *Bird* into the schema using schema expansion algorithm, and then the objects are created as instances of *Bird*.

(d) When another time interval is elapsed, class *Animal*, *Bird*, and *Elephant* are considered for *C2*. Non-leaf class *Animal* fails to satisfy *C2.1*, while newly created class *Bird* fails on *C2.2*. However, class *Elephant* satisfies all three conditions, because it locates at leaf node, stay untouched during last time interval, and occupies a too small proportion of objects under its parent class. Therefore, it is removed by the schema compaction algorithm with its objects migrated to class *Animal*.

[3] In the case study, the initials *E*, *C*, and *A* stand for Event, Condition, and Action of an ECA-rule, respectively.

## 5. PROTOTYPE SYSTEM

The semantic schema mechanism (SSM) proposed in this paper has been implemented as the data model of our prototype system, *2M2Net*, a search engine for multi-modality data in digital libraries [16]. The details of the architecture of *2M2Net* system can be found in [17]. In *2M2Net*, the proposed schema is built upon a small-scale OODBMS (namely, *NeoAccess* [9]). Because our schema requires no modification to the kernel of an object model, *NeoAccess* can take care of most of the issues as to modeling, storage, management and retrieval (using a SQL-like language) of multimedia data. Furthermore, since *NeoAccess* interfaces with users through a persistent C++ programming language, we can easily encode additional functionalities such as ECA-rules into the schema.

All the experiments conducted on *2M2Net* so far have demonstrated its effectiveness and efficiency [16], which show that the proposed mechanism is at least feasible. A thorough study of the schema performance, evaluated in terms of its descriptiveness and efficiency, is yet to be conducted. The primary difficulty is the lack of viable approaches to estimate the descriptive power of the schema. Although retrieval accuracy is certainly an indication of descriptiveness, it is also influenced by many other factors that can be hardly precluded. Another difficulty comes from the trouble of collecting tremendous amount of data needed to examine the schema efficiency. Data collections of small or medium size are unlikely to produce any discernible difference in efficiency as the result of using different schemas.

## 6. CONCLUSIONS

This paper has presented a self-adaptive semantic schema mechanism (SSM) for multimedia databases, which takes the advantages of several database techniques including object-oriented data modeling, schema evolution with object migration, and ECA-rules. This mechanism features a self-adaptive evolution strategy, which exploits a prototypical schema template to automatically optimize towards the requirement of a specific application. Multimedia applications built on the proposed schema are able to achieve a dynamic, application-specific optimal balance between accuracy and efficiency with regard to data retrieval.

As a response to the shortcomings mentioned in Section 3, we plan to extend the modeling capacity of the current schema to support aggregation relationship as well as multiple inheritance. Aggregation relation between objects can help to construct composite objects and conduct complex queries, while multiple inheritance is used to model objects with several applicable semantic concepts. Another possible direction of future work is to improve the rule-based schema evolution strategy by including the rule management, analysis, and trace capabilities. Before these modifications and extensions are carried out, their costs and gains will be thoroughly studied.

## ACKNOWLEDGEMENT

The work described in this paper was supported, substantially by a grant from CityU (Project No. 7001384), partially by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 1119/99E], and partially by a grant from the Doctorate Research Foundation of the State Education Commission of China.

## REFERENCES

1. Apers, P. M. G., Blanken, H. M., Houtsma, M. A. W., *Multimedia Databases in Perspective*, Springer-Verlag, 1997.
2. Flickner, M., Sawhney, H., Niblack, W., Ashley, J., "Query by image and video content: The QBIC system", *IEEE Computer*, 28(9): 23--32, 1995.
3. Fox, E. A., "Advances in interactive digital multimedia systems", *IEEE Computer*, 24(10): 9-21, 1991.
4. Haas, L. M., "Supporting multi-media object management in a relational database management system", Technical report. IBM Almaden Research Center, 1989.
5. Hauptmann, A., Smith, M., "Text, speech, and vision for video segmentation: The Informedia project", *AAAI Fall 1995 Symposium on Computational Models for Integrating Language and Vision,* 1995.

6. Jagadish, H. V., “Spatial search with polyhedra”, *Proc. of Sixth IEEE Int. Conf. on Data Engineering*, pp. 311-319, 1990.

7. Li, Q., Dong, G. Z., “A framework for object migration in object-oriented databases”, *Data and Knowledge Engineering*, 13: 221-242, 1994.

8. Miller, G. A., Beckwith, R., Felbaum, C., Gross, D., Miller, K. “Introduction to WordNet: an on-line lexical database”, *International Journal of Lexicography*, 3(4): 235-244, 1990.

9. NeoAccess, http://www.neologic.com

10. Oomoto, E., Tanaka, K., “OVID: design and implementation of a video-object database system”, *IEEE Transactions on Knowledge and Data Engineering*, 5: 629-641, 1993.

11. Orenstein, J. A., Manola, A., “PROBE spatial data modeling and query processing in an image database application”, *IEEE Transactions on Software Engineering*, 14 (5): 611-629, 1988.

12. Orenstein, J. A., “A comparison of spatial query processing techniques for native and parameter spaces”, *Proc. of ACM SIGMOD Conf.,* pp 343-352, 1990.

13. Smeaton, A. F., Quigley, I. “Experiments on using semantic distance between words in image caption retrieval”, *Proc. of the 19th International Conference on Research and Development in Information Retrieval,* pp. 174-180, 1996.

14. Widom. J., Ceri, S., *Active Database Systems: Triggers and Rules for Advanced Database Processing*. San Francisco, California: Morgan Kaufmann, 1996.

15. Yang, J., Liu, W. Y., Zhang, H. J., Zhuang, Y. T., “Thesaurus-aided approach for image retrieval and browsing”, *Proc. of the 2nd IEEE International Conference on Multimedia and Exposition*, pp. 313-316, 2001.

16. Yang, J., Zhuang, Y. T., Li, Q., “Search for multi-modality data in digital libraries”, *Proc. The 2nd IEEE Pacific-Rim Conference on Multimedia*, pp. 482-489, 2001.

17. Yang, J., Zhuang, Y. T., Li, Q., “Multi-modality retrieval for multimedia digital libraries: issues, architecture, and mechanisms”, *Proc. 7th International Workshop on Multimedia Information Systems,* pp. 81–88, 2001.

18. Zhou, Q.Q., Li, Q., Liu, W.Y., Zhang, H.J., “A practitioner's approach to multi-dimensional indexing”, *Proc. 7th International Conference on Distributed Multimedia Systems (DMS'2001)*, Vol.2, pp. 257-265, 2001.