A Distance Learning System for Content-Based Lecture Retrieval

Guang-Ho Cha

Department of Computer Engineering, Seoul National University of Science and Technology, Republic of Korea
ghcha@snut.ac.kr

Abstract: Education and training are expected to change dramatically due to the combined impact of the Internet, database, and multimedia technologies. However, the distance learning is often impeded by the lack of effective tools and system to manage and retrieve the lecture contents effectively. This paper introduces a system that enables remote users to access specific parts of interest from a large lecture database by contents. The system includes several novel techniques to achieve the content-based lecture retrieval: (1) The XML(eXtensible Markup Language)-based semistructured model not only to represent lecture contents but also to exchange them on the Web; (2) The technique to build structural summaries, i.e., schemas, of XML lecture databases. The structural summaries are useful for browsing the database structure, formulating queries, building indexes, and enabling query optimization; (3) An index structure to speed up the search to find appropriate lecture contents.
[Cha G.-H. **A Distance Learning System for Content-Based Lecture Retrieval**. *Life Sci J* 2014;11(7):743-748] (ISSN:1097-8135). http://www.lifesciencesite.com. 109

Keywords: Distance learning; lecture model; lecture browsing; lecture querying; lecture indexing;

1. Introduction

Education and training are expected to change dramatically due to the combined impact of the Internet, database, and multimedia technologies. Besides the economic impact – online education means less traveling, hence lower cost – the expectation is that the educational process itself will change radically. Video is the most effective medium for providing remote and future students with a lecture because of its expressive power that combines images and voice. Moreover, the ability of recording and subsequently playing back live instruction sessions could significantly enhance the students' learning effectiveness because it allows them to review class lectures repeatedly. Unfortunately, however, the benefit of video-based lecture is often impeded by the fundamental difficulties with information retrieval: if one is trying to locate specific information on a video source, finding it can be a process that is time consuming and tedious. In addition, the contents of class lectures are diverse, and the same course can be given over and over again with different contents and structures by different instructors. Thus, we cannot conform the lecture content to a rigid, predefined schema. Three crucial issues that need to be addressed are: (1) the representation of lecture contents in a form that facilitates retrieval and interaction; (2) the structural summary of a lecture database that guides users to browse and query the database; and (3) the indexing scheme to expedite the search.

Browsing and *querying* in a lecture database for distance learning should provide the same ease of use as flipping through the pages of a book and scanning the table-of-contents and index pages to get ideas of the content quickly, and then gradually focusing on particular chapters or sections of interest. For a lecture database, this is not as straightforward as browsing and querying in a book. We have to identify the chapters, sections, and subsections of a lecture, and create table-of-contents and index pages for lecture, both structured and unstructured, so that we can get an overview and know where to find relevant contents.

The transformation of a simple lecture into a valuable educational tool requires five steps. First, we partition a lecture into individual lecture segments by exploiting the hierarchical structure of the lecture. A lecture segment consists of a set of lecture notes and any contiguous portion of a video clip which constitutes a digital video lecture. Each lecture segment is associated with the system-wide unique identifier. Second, we abstract the contents of lecture segments with text descriptions, meaningful attributes, and key images, and organize them into an effective structure that facilitates retrieval and interaction. Third, we need tools to aid the user for browsing a lecture database and formulating queries. Although it may be possible to manually browse a small database, in general forming a meaningful query is difficult without knowledge of the database structure. Fourth, we index all useful objects appearing in lecture segments to efficiently locate specific lecture segments of interest. Finally, we need query optimization techniques to reduce the search space and expedite the search since there are numerous query plans for each query.

In this paper, the XML-based semistructured model is introduced for content-based lecture access. It fully supports XML data and represents lecture

contents without rigid, fixed schema. Database structure summarization, i.e., schema extraction, technique for irregular lecture database is used to guide users in data-base browsing and querying. An index structure is presented to efficiently locate not only a specific lecture segment but also a collection of semantically related lecture segments.

2. Video Segmentation

While a video clip consists of a sequence of frames, it is not meaningful to use the individual frames as the units for video retrieval. Rather, it is advantageous to identify meaningful segments of video to serve as retrieval units. As defined in [3], the fundamental unit of video production is a *shot* that consists of a contiguous sequence of video frames. While the video segmentation based on image processing techniques automates the process of video parsing, it has the following problems for distance learning:

- For a video clip of a class lecture, there can be no clear visual cue for shot change detection. Therefore, video segmentation using shot change detection algorithms would be difficult.
- Shots do not capture the underlying semantic structure of a class lecture, based on which the user may wish to browse and retrieve the video lecture.

On the ground of above motives, we do not pay special attention to the problem of the video segmentation based on image processing. Rather, we automatically extract descriptive text information from the instructor's lecture notes, and manually describe the necessary semantic video units and their contextual information. After that, the video lectures are automatically indexed, converted to a Web-ready format, and made available to end users through the Internet.

A lecture is organized into *presentation slides* (i.e., lecture notes) and video segments. Each slide corresponds to a single page course note assumed to be written in XML. Instructors lecture by showing electronic course slides, and recording of lectures is expected to capture the video of live lecture sessions. In our work, we define a *video shot* as the video segment synchronized with a single slide. The synchronization of slides with video segments can be easily made because instructors are required to explicitly switch slides during live lecture session. When students access a particular lecture in a course, they see the presentation similar to Figure 1. By allowing remote or future users to not just view presentation slides but also to see and hear the presenter, the instructor achieves a broader reach and increased productivity and the audience gets a richer experience that enables them to retain more

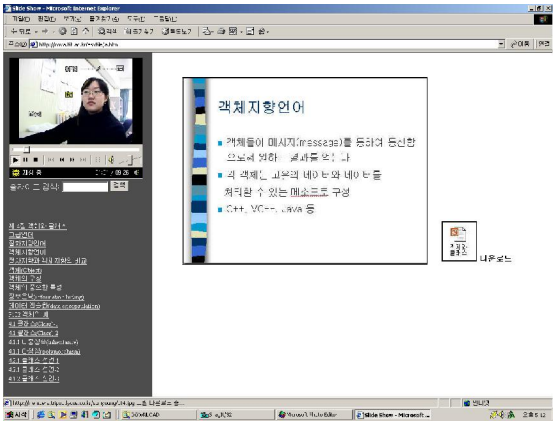

Figure 1. Online presentation window

information and saves on travel costs. However, the more important things we need for is to locate and retrieve a particular piece of the video lecture because watching the whole video is time consuming.

3. Lecture Database Model

Data modeling deals with the problem of how to represent the data to facilitate users' access. To the best of our knowledge, there have been no efforts to model the lecture database. Most of early research effort has been devoted to the shot-based video segmentation and each video shot is described using text descriptions and cinematic attributes. For a video clip of a class lecture, however, there can be no clear visual cue for shot change detection. Therefore, video segmentation using shot change detection algorithms would be difficult. Moreover, shots do not capture the underlying semantic structure of a class lecture. On the ground of above motives, we extract descriptive information from the instructor's lecture notes, and describe it with XML. After that, the lectures are indexed and converted to a Web-ready format.

To represent the descriptive information extracted from the lecture notes, we adopt the semistructured data model [1, 2, 6], specifically, XML-based semistructured model. The motivation to employ the semistructured model comes from the need to provide the lecture content description with flexibility and diversity. Because the lecture contents are diverse and rich, we cannot conform the lecture database to a rigid, predefined schema. Moreover, the motivation to fully support XML data is to exchange lecture data on the Internet. By semistructured data we mean data that has no absolute schema fixed in advance, and whose struc-ture may be irregular or incomplete. Like in the standard model [1, 2, 6] for semistructured data, a lecture database is thought of as a labeled directed graph. For example, Figure 1

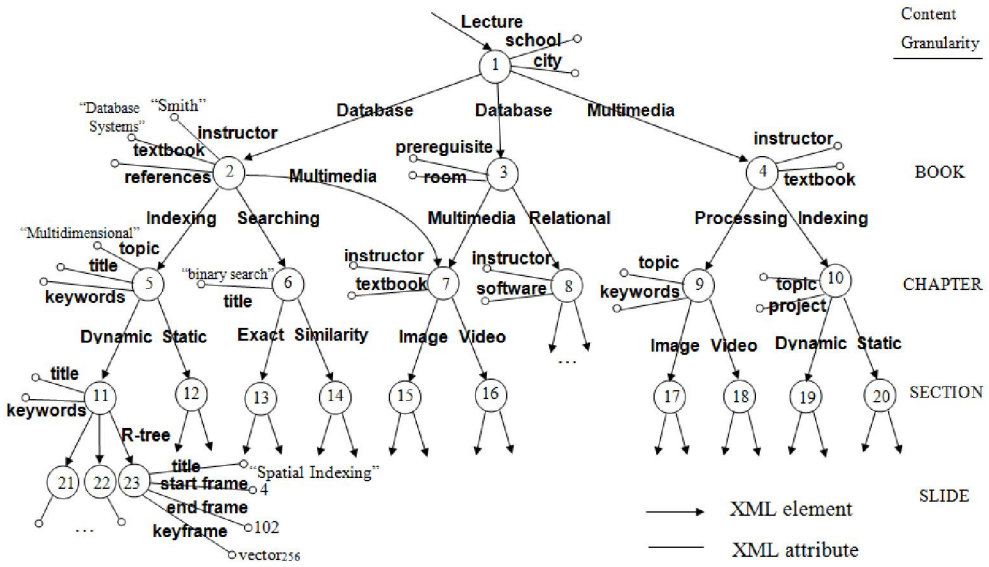

Figure 2. An example lecture database (Some nodes are omitted and only a few values of attributes are shown)

depicts a portion of a lecture database containing three class lectures (two for a database course and one for a multimedia course). Each node corresponds to an XML element and can have attributes depicted as small circles in Figure 2. Our example database is almost tree-structured because of the hierarchical nature of the book for lecture even though the semistructured model permits arbitrary graph-structured databases. Each level of our example lecture database represents the level of content granularity. For example, we can assume that nodes 2 through 4 are in the level of book, nodes 5 through 10 are in the level of chapter, nodes 11 through 20 are in the level of sec-tion, and so on.

Unlike the standard semistructured model, our data model fully supports the XML data. In other words, it allows us to associate attributes with graph nodes (XML elements). In our data model we call the nodes lecture objects (LOs) in which the video segments and presentation slides for lecture are associated. An LO can be viewed as a 6-tuple (PID, OID, a set of video segments, a set of presentation slides, a set of sub-elements, a set of attributes). We should note that the elements and the attributes attached to LO are not pre-fixed. Each LO has a unique object identifier (OID), such as 1 to 23 in Figure 1, and outgoing edges that correspond to its sub-elements. Every LO belongs to a certain type and the type is identified by a path identifier (PID). In our model, a type is defined by a path on the extracted schema graph, which will be described in the next section. Labels are attached to the edges and they serve as names for LOs or attributes. Our example database in Figure 2 contains one root LO which represents the Lecture database and contains three sub-LOs, two Databases and one Multimedia. Database LO 2 has three attribute-value pairs describing its instructor, textbook, and references, while Database LO 3 has two attributes prerequisite and room. Unlike the standard semistructured model, sub-LOs under an LO in our model are ordered to reflect the timing sequence of the video segments associated with them. We can see that the database structure based on the semistructured model is irregular since, for example, two Database objects (LO 2 and LO 3) have different structures.

4. Summarizing Lecture Database

Two completely different types of lecture retrieval requests can be expected from the end-user:

- *Querying*: The user retrieves particular lecture objects for viewing or reuse.
- *Browsing*: The user traverses a lecture database along the semantic links.

A query processor should respond to both types of retrieval requests by providing the user with query formulation tools for querying and optimal starting points for browsing. When we model a lecture retrieval request as an iterated sequence of querying and browsing, each step should act as an information filter reducing the search space and give a more refined search space to the next step. In a small database, although it may be possible to browse the whole database, in general it is difficult and tedious to browse a large database. It is reasonable to pose a query at the start by using some attributes. However, since our lecture database is based on the XML-based semistructured model, i.e., it is schemaless, it needs a tool that assists users in query formulation by providing the information (i.e.,

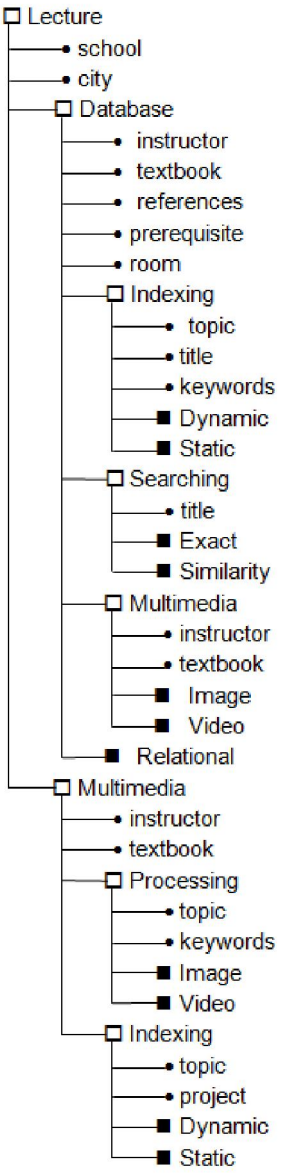

Figure 3. Structural summary of the example database in Figure 2

schema) summarizing the database structure. The schema allows users to browse and query easily through the database. Also, it improves the system performance greatly by enabling to take advantage of indexes and query optimization.

Figure 3 shows the structural summary of the lecture database given in Figure 2. A rectangle corresponds to an XML element in the database, and small black circles denote XML attributes in the XML element. Every XML element of an original database is described exactly once in the structural summary, regardless of the number of times it appears in that database. There is no XML element that does not appear in the original database. From the structural summary, a user can interactively query

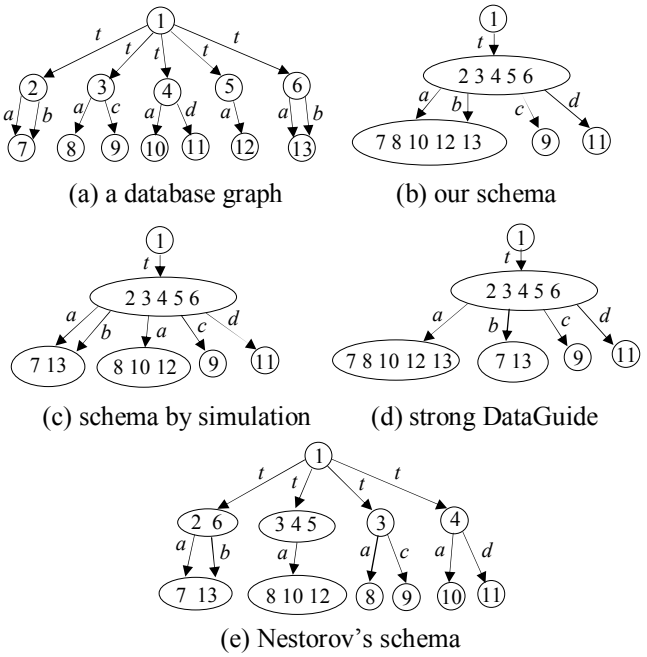

Figure 4. Comparison of database schemas

and browse the graph-based database. Clicking on a rectangle on the structural summary expands or collapses LOs. The white rectangle indicates that the LO has been expanded and the black rectangle indicates that the LO has not. For example, Database and Multimedia LOs have been expanded, while Dynamic and Static LOs have not.

We develop a new technique to improve the running time to build the structural summary of a database. In our method, the structural summary does not have any redundancy in nodes and edges in a schema graph. We can best explain the difference among our technique, Buneman et al.'s [2], DataGuide [4], and Nestorov et al.'s [5]. Figure 4(a) illustrates a database graph *DB* and 4(b) is our summary of *DB*. Figures 4(c), 4(d), 4(e) show Buneman et al.'s schema based on simulation, strong DataGuide, and Nestorov et al.'s minimal perfect typing, respectively. We compare the schemas by their size. DataGuides require a powerset construct over the underlying database, which in the worst case can be of exponential cost. As you can see in Figure 4(d), elements 7 and 13 are replicated in nodes in the DataGuide. The schema based on simulation guarantees its size, that is, the size of the schema is at most linear in that of the database. However, as we can see in Figure 4(c), edge *a* outgoing from the same node is replicated. Figure 4(e) also shows redundancy in nodes and edges. On the other hand, our database schema in Figure 4(b) does not have any redundancy in nodes and edges. The compactness of our schema

key value	overflow page pointer	no. of PIDs	PID_1	no. of OIDs	$\{OID_{11},\ldots, OID_{1i}\}$	…	PID_n	no. of OIDs	$\{OID_{n1},\ldots, OID_{nk}\}$

Figure 5. An entry of a leaf node of the P-index

results in efficiency in query evaluation as well as in database summarization.

4.1 The Algorithm

We define some terminologies before proceeding.

Definition 1. A *data object* is a node, i.e., LO, in a database graph.

Definition 2. A *target set* for a path *l* is a set of data objects that can be reached by traversing a path *l* in a database graph.

Definition 3. A *schema object* is a node in the schema graph that corresponds to a target set of a path *l* in a database graph.

The schema extraction is easy to implement with our algorithm. The root data object becomes a root schema object. In a depth-first fashion, we extract all child schema objects reachable by all unique paths outgoing from a schema object. Each time we encounter a new target set for a unique path l, we create a new schema object s. If we reach a schema object s via a path l and a data object o is already included in the schema object s with a different path m, rather than creating a new schema object we instead add an edge l to the schema object s. The algorithm is specified as follows.

Algorithm ExtractSchema(*o*)
// Input: root oid *o* of a database
// Output: database schema *s*
{
  s := CreateSchemaObject();
  Insert {*o*} to *s*;
  RecursiveMake(*s*);
}

Algorithm RecursiveMake(*s*)
{
  Let *S* be a set of current target sets under *s*;
  Let S_j denote a certain target set included in *S*;
  For each unique label l_i outgoing from *s* {
    o := target set reachable by l_i;
    If (*o* and S_j have data objects in common) {
      Add an edge l_i from *s* to the schema object corresponding to S_j;
      $S_j := o \cup S_j$;
    }
    Else {
      s_2:= CreateSchemaObject();
      Insert s_2 to *s*;
      Add an edge l_i from *s* to s_2;
      RecursiveMake(s_2);
    }
  }
}

5. Indexing

In this paper, we propose a new index structure called P-index (path index) to index paths on the database graph. The structure of the P-index is based on the B^+-tree. The internal node of the P-index has the same structure as that of the B^+-tree. The leaf node has a format different from that of an internal node. Each index entry in the leaf node has a form shown in Figure 5. For path indexing, the P-tree maintains in a leaf node the lecture objects on the label paths from the qualifying objects to the root. The P-index is somewhat similar to the class-hierarchy indexing [7] used in object-oriented databases. The class-hierarchy indexing maintains one index on a common attribute for a hierarchy of classes. On the other hand, there is no concept of a common attribute in the irregular semistructured database. Instead, the P-index maintains one index on every path from the qualifying objects to the root.

6. Conclusions

We presented a new approach to the distance learning based on the XML-based semistructured model. By employing this model, we could provide the lecture contents with flexibility and diversity as well as exchange them conveniently on the Internet. Based on this model, we described the technique to extract schemas from a graph-based database. In irregular semistructured database, without schema, it is difficult to query and browse the database, to construct indexes, and to perform query optimization. An index structure for path query was also introduced to speed up the search. Read-intensive lecture database applications justify the extensive use of index structures to speed up the query processsing.

Acknowledgements:

This study was financially supported by Seoul National University of Science and Technology.

Corresponding Author:
Dr. Guang-Ho Cha
Department of Computer Engineering
Seoul National University of Science and Technology, Seoul 139-743, Republic of Korea
E-mail: ghcha@snut.ac.kr

References

1. S. Abiteboul, "Querying Semistructured Data," Proc. of ICDT, pp. 1-18, 1997.
2. P. Buneman, "Semistructured Data," Proc. of ACM PODS, pp. 117-121. 1997.
3. G. Davenport, T.A. Smith, and N. Pincever, "Cinematic Primitives for Multimedia," *IEEE Computer Graphics and Applications*, pp. 67-74, 1991.
4. R. Goldman and J. Widom, "DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases," *Proc. of the International Conference on Very Large Data Bases*, pp. 436-445, 1997.
5. S. Nestorov, S. Abiteboul, R. Motwani, "Extracting Schema from Semistructured Data," *Proc. of ACM SIGMOD*, pp. 295-306, 1998.
6. Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, "Object Exchange Across Heterogeneous Information Sources," Proc. of ICDE, pp. 251-260. 1995.
7. W. Kim, K.-C. Kim, and A. Dale, "Indexing Techniques for Object-Oriented Databases," Object-Oriented Concepts, Databases, and Applications, pp. 372-394, ACM Press, 1989.