

# Object-Oriented Modeling, Import and Query Processing of Digital Documents

Andre Zeitz  
Database Research Group  
Computer Science Department  
University of Rostock, Germany  
zeitz@informatik.uni-rostock.de

Ilvio Bruder  
Database Research Group  
Computer Science Department  
University of Rostock, Germany  
ilir@informatik.uni-rostock.de

## Introduction

Digital libraries have to manage data structures of bibliographic documents in a more complex way. The requests of librarians and library users are growing. They expect different points of view to the data and different search possibilities. The demonstration described here defines a way to integrate fine granular structures of digital documents in an object-oriented database. A generic data model was specified to provide a simple and flexible way to manage the documents.

Due to the fact that documents can be multifaceted it is quite difficult to manage bibliographic documents in a digital library. We want to provide different search and representation scenarios. The system should be efficient and also flexible managing heterogeneous documents. The building of digital library collections should be as simple as possible.

The demonstration system includes a possibility to create and import document collections, to map document structures to the internal generic model, and finally to search for documents and to present them.

## Generic Data Model

The generic data model presented in this demonstration is organized as a tree which consists of two node types: inner nodes and leaf nodes. An inner node contains a set of attributes and a set of further nodes. Attributes describe the node or the data inside that node independent of the metadata. Optionally, an inner node contains a metadata record that can occur in different representations, e.g., in Dublin Core and USMARC. A leaf node is used to store any kind of unstructured data like fulltext, e.g., a book chapter or a chapter of a DVD.

All kinds of nodes are stored inside an object-oriented database. The content of each leaf node is additionally managed outside the database, dependent on the data type of the leaf nodes. For example, fulltexts are indexed using a fulltext indexer.

## Import of Documents

Documents have a schema defined during the digitalization or during the authoring. This schema describes the logical and the physical structures of a document.

The import of documents is a two-stage process. During the first stage the schema is manually imported in the database. Then the documents corresponding to the schema are automatically imported during the second stage. Imported documents have often a specified schema, e.g., XML-DTD or XML-Schema. Documents without a schema are imported as one structure unit. Documents have also metadata which are described in other resources.

The first import step implies the definition of a mapping from schema elements onto internal data structures. Thereby, some schema elements are mapped to inner nodes of the generic model, the others are left as fulltext in the leaf nodes. If the schema element is important for search scenarios, or if it specifies a logical structure, or if it is a metadata element, it has to be defined as an inner node. Layout elements and structure elements which are not used in search scenarios are converted into leaf nodes. The result schema of the mapping is an internal data model similar to the generic data model as mentioned before.

The second import step is the document parsing to import the document data into the database. The document parsing is document type dependent. In our test case (a couple of XML-structured books) we use an XML parser for XML documents. The mapping definition of the first import step is used to import automatically the documents corresponding to the schema. The data object of one document is created by traversing the document structure. After creating the data object, its state is changed from transient to persistent.

## Searching the Data

Searching in our system is done through search forms which are derived automatically from the schemes. Because of the structuring of data, users can search for parts of objects, e.g., for particular sections of a book. Thus using our implementation you can search for a chapter that is part of a certain book and contains a given word. The result of such a query is not the whole document but the chapter you are looking for.

To search for documents that do not have a corresponding schema you can only search for metadata and fulltext. To search for structure elements a schema is essential.