# Table of Contents

In Proc. Datenbanksysteme in Büro, Technik und Wissenschaft (BTW) 1999, March 1–3, 1999, Freiburg, Germany.

**In Proc. Datenbanksysteme in Büro, Technik und Wissenschaft (BTW) '99, March 1–3, 1999, Freiburg, Germany.**

# Exploiting ORDBMS Technology to Implement the ZYX Data Model for Multimedia Documents and Presentations

Susanne Boll, Wolfgang Klas, and Utz Westermann

Database and Information Systems (DBIS), University of Ulm,
Computer Science Department, Oberer Eselsberg, 89069 Ulm, Germany
{boll, klas, westermann}@informatik.uni-ulm.de
http://www.informatik.uni-ulm.de/dbis

**Abstract.** Advanced multimedia applications require adequate support for the modeling of multimedia information but also an adequate database technology supporting the integrated management, retrieval, delivery, and even the presentation of such multimedia information. In our CardioOP project within the application domain of cardiac surgery, we are developing concepts and prototypical implementations of a database system-driven multimedia repository that gives this comprehensive support. In the project context, we have been developing the ZYX data model that meets the specific requirements of *reusability*, *interaction*, *adaptation*, and *presentation-neutral description* of the structure and content of multimedia documents. For the multimedia repository, hence, we need to select database system technology for the implementation of the ZYX model for multimedia documents and — with regard to our future work — for the retrieval, browsing, delivery, and presentation of the multimedia documents. In this paper, we present our approach to implement the ZYX data model for multimedia documents and presentations using object-relational database technology. Hereby, we exploit the features and experience the limitations of object-relational database system technology in comprehensively supporting multimedia applications.

## 1 Introduction

For an adequate support of multimedia applications not only sufficient support for modeling of multimedia information is needed but also for retrieving and delivering and even presenting this information. Multimedia information here comprises the various types of *multimedia data* like image, audio, and video, *multimedia documents* that represent multimedia presentations, and *metadata* associated with data and documents. Data modeling capabilities, consistent and persistent management of data, efficient retrieval mechanisms, and reusage of

stored data are features that a database management system (DBMS) can offer to support multimedia applications. Taking into account the particular requirements of multimedia applications, a database management system must also offer support for multimedia information like multimedia documents and multimedia data. Not only efficient data management, but also data manipulation functionality is needed for this kind of information. This includes modeling and exploiting metadata for content-based and context-based retrieval of multimedia information within the DBMS [SK98]. Another challenging issue in this context is to provide techniques for continuous media data delivery and presentation services integrated into a database system in order to achieve *really multimedia capable* database system support [BKL96,MKK95].

Background for this work is the project "Gallery of Cardiac Surgery" (Cardio-OP)[1] which aims at the development of a network-based and database-driven multimedia information system for physicians, medical lecturers, students, and patients in the domain of cardiac surgery. The overall goal is to develop a system that will serve as a common information and education base for its different types of users in the domain of cardiac surgery. The physicians, medical lecturers, students, and patients are provided with multimedia information composed according to their context, i.e., their different understanding of the selected subject, their location and technical infrastructure. Our focus here is the fine-grained reusage of multimedia material in different user and application contexts.

Within this project, our group is developing concepts and prototypical implementations of a database-driven multimedia repository. This repository *stores* the multimedia information from the application domain, supports its *content-based retrieval*, and integrates its *delivery and presentation*. As the information system is common to different user groups it will be designed to support flexible reuse of the multimedia material in different user context, with different communication media (on-line, CD-ROM, print media), and at different locations (university campus, hospitals, at home). To achieve this goal, a suitable data model for multimedia information that allows for reusage is needed in the first place.

Starting out from the need for a data model supporting modular and context-dependent composition of multimedia documents, we have analyzed interesting existing multimedia document models like HyTime [ISO92], MHEG-5 [ISO95], and SMIL [HBB+98] with respect to the project requirements. As these do not meet our requirements we have been developing the ZyX data model [BK99] for multimedia documents. In comparison to existing models it provides more adequate support for *reusability* and *flexible composition*, *semantic modeling*,

[1] Cardio-OP - Gallery of Cardiac Surgery - is partially funded by the German Ministry of Research and Education, grant number 08C58456. Our project partners are the University Hospital of Ulm, Dept. of Cardiac Surgery and Dept. of Cardiology, the University Hospital of Heidelberg, Dept. of Cardiac Surgery, an associated Rehabilitation Hospital, the publishers Barth-Verlag and dpunkt-Verlag, Heidelberg, FAW Ulm, and ENTEC GmbH, St. Augustin. For details see also URL www.informatik.uni-ulm.de/dbis/Cardio-OP/

*adaptation* and *individualization* for presentation, and *presentation-neutral storage*. Hence, support for ZYX multimedia documents must be provided by the DBMS.

For the design of the project's multimedia repository, we have been investigating DBMS technology for comprehensive support of multimedia applications especially in the project context. When selecting the DBMS technology most suitable to offer this support, we were considering relational, object-oriented, object-relational, and multimedia database technology. We discuss in this paper why object-relational database technology is our choice and how it can be exploited for modeling flexible multimedia documents.

The paper is organized as follows: Section 2 discusses the different DBMS technologies with regard to the project's specific needs. Section 3 introduces the reader to our ZYX data model for multimedia documents. Section 4 presents an object-oriented model for ZYX. Section 5 presents those features of the selected ORDBMS, the Informix Dynamic Server / Universal Data Option, that allow for the extension of the DBMS and, hence, the modeling of ZYX documents in the DBMS schema. Section 6 presents the mapping of the object-oriented model to the object-relational database schema, the ZYX DataBlade. Section 7 summarizes our practical experiences with the selected ORDBMS. The paper concludes with a summary and an outlook to ongoing and future work.

## 2 Evaluation of DBMS technology

When selecting the DBMS technology most suitable to offer support for the project requirements mentioned in Section 1, we were considering relational, object-relational, object-oriented, and multimedia database technology.

With relational database technology we do not find sufficient modeling capabilities for multimedia data [KA97,CC96]. These systems stay with the introduction of untyped Long Fields or Binary Large Objects into the system to integrate, e.g., audio and video data. They do not offer built-in modeling features for complex data types as they occur with multimedia data. Data delivery features or even presentation features cannot be found with relational database systems. The support for multimedia applications is limited to the management of untyped media data that cannot be exploited for query optimization and optimized indexing.

Object-oriented DBMS technology on the other hand has often been proposed to be a good approach to model the complexity of multimedia data [KA97,Paz97]. It allows for the specification and management of arbitrary complex media objects and documents. The multimedia data types, however, are not built-in data types of the system but modeled in the object-oriented database schema. Therefore, indexing, querying, and transaction management are not directed to exploit multimedia specific features as the internal structure and semantics of the multimedia data types are not understood by the OODBMS. Moreover, object-oriented database systems most often do not offer a standardized ad-hoc query language allowing for the retrieval of multimedia information.

And scalability of the existing object-oriented database systems is still in question.

With the integration of object-oriented approaches into relational DBMS technology one keeps the well-known features of relational DBMSs but integrates the modeling of extensible, complex data types. Hence, the ORDBMSs stay with proven features of relational DBMSs and adjust object-oriented features to the relational framework. ORDBMS systems stay with the SQL ad-hoc query language which is extended by constructs that are introduced with the new data types. The extensibility features of an ORDBMS can be exploited to introduce multimedia data types into the system. However, when considering object-relational database technology we must distinguish the extent to which object-oriented features are introduced to the ORDBMS. For example, IBM's DB2 [IBM98,Cha98] offers only used-defined functions but no user-defined complex data types. Oracle8 [Ora98,KL97] offers no support for inheritance and user-defined indexing techniques. With the Informix Dynamic Server / Universal Data Option (IDS/UD) [Inf97] we find a system that provides a high degree of extensibility with complex data types, inheritance, user-defined routines, and user-defined indexing techniques.

The approaches to integrate multimedia capabilities with DBMS technology to build a Multimedia DBMS (MMDBMS) are mainly based on object-oriented database technology [NBT96,Chu96,SJ96,RNL95]. Some systems extend the OODBMS's inherent features, others just make use of the underlying OODBMS and introduce the MMDBMS features in a specific database schema. The AMOS system [RNL95,RKN96] still appears to be the most advanced MMDBMS [Paz97]. For instance, AMOS features a new multimedia data type for audio, integrated in the underlying OODBMS VODAK, so that it is a built-in multimedia data type that can be used in the system's DML like any other "simple" data type. It also supports continuous media data delivery integrated with the MMDBMS [MKK95]. As we were involved in the development of the AMOS system which offers comprehensive support for multimedia, we know that it remains a research prototype though. An OODBMS called Jasmine which features built-in multimedia data types has been released by Computer Associates recently. Jasmine must be considered in future but it still has to receive its baptism of fire from the market demanding industrial-strength systems. We do not find OODBMS-based systems that offer sufficient multimedia support and at the same time are ready for the commercial market. Currently multimedia support is moving into ORDBMSs whose extensibility features provide a means for the integration of multimedia capabilities into the DBMS. The commercially available systems offer industrial-strength but, as mentioned above, have to be distinguished carefully by the way and the degree of multimedia support.

The Cardio-OP project — of which results are to be exploited commercially by the project partners from the publishing domain — requires an industrial-strength DBMS. Due to performance reasons and query functionality we were in favour of an ORDBMS. As we think that extensibility of such a system makes sense only if the extensibility is known and supported by the DBMSs traditional

components like query optimization, indexing, and transaction management we came to the conclusion that IDS/UD is that out of all commercially available systems that offers sufficient support for creating complex new data types within the database system's kernel — with all the respective consequences as we will show in this paper. However, IDS/UD offers still no native support of multimedia data types within the database system itself but the possibility to extend the system by new complex data types and user defined routines by means of *DataBlades* that are insertions into the DBMS's kernel. Hence, the new multimedia data types are known to the DBMS. To support the different media types there are already Third-Party and Informix DataBlades available or at least under development that give a sufficient initial support for multimedia data. Support of multimedia documents is possible by the use of complex data structures. The integrated support for multimedia data delivery and presentation is also still lacking and again part of the necessary extension of the ORDBMS.

## 3 The ZYX Model

In this section, we introduce the reader to the basic concepts of our ZYX data model as formally defined in [BK99] in order to provide a basic understanding for the contributions in the subsequent sections.

The ZYX model describes a multimedia document by means of a tree, a so called *specification*. The nodes of the specification are the *presentation elements* and the edges of the specification, called *bindings*, bind the presentation elements together in a hierarchical fashion. Each presentation element has one *binding point* with which it can be bound to another presentation element. It also has one or more *variables* with which it can bind other presentation elements to it. Figure 1 illustrates these basic elements of the model in a graphical representation.

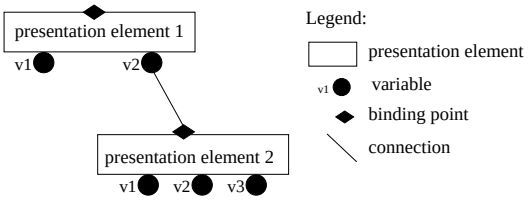


**Fig. 1.** Graphical representation of the basic document elements

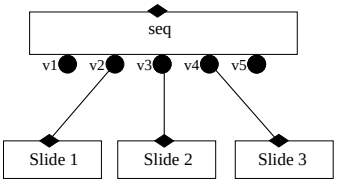


**Fig. 2.** Simple specification with *seq* element

Presentation elements are the generic elements of the model. They can be categorized into *fragments*, which are reusable building blocks that can range from mere media elements to whole multimedia documents, and *operator elements* that model the temporal, spatial, layout, and interactive semantic relationships

between the elements of a multimedia document. Consider the example in Figure 2. A temporal operator element, the sequential element *seq*, binds the media elements $slide_1$, $slide_2$, and $slide_3$ to its variables $v_2$, $v_3$, and $v_4$. This represents the semantic relationship of the sequential presentation of the three slides. With the *seq* element's binding point this sequential slide presentation can be bound to another presentation element in a more complex multimedia document specification.

In the following, we explain the modeling capabilities of our model with regard to our specific requirements of reusability, adaptation, interaction, and presentation-neutral representation.

### 3.1 Reusability

Basically, any fragment can be reused. A fragment can be an *atomic media element*, which represents media data of a single media type like a video, a *complex media element*, which encapsulates a specification in ZYX , or an *external media element*, which encapsulates a document specified in an external document format.

It is possible to reuse complete or partial ZYX specifications. In the ZYX model, not all of the variables of a presentation element must be bound at authoring time. In Figure 2 the variables $v_1$ and $v_5$, e.g., the title and the summary of the slide presentation, are still unbound. This means that the slide sequence can later be completed by binding presentation elements to the *free variables*. The simple sample specification in Figure 2, hence, forms a "template". This is an important feature for building fragments that can be reused in different multimedia documents by binding the free variables differently corresponding to the context. A more complex specification is shown in Figure 3. Here, on different "levels" of the specification tree variables are left unbound. To make later reuse easier a specification can be encapsulated by a complex media element. This means that a specification appears like a single presentation element in the specification tree with one binding point and a set of free variables. The free variables of the encapsulated specification are *exported*. Figure 3 illustrates how a complex media element encapsulates a specification. The complex media element somehow is the black box view to a complex specification.

Analogously, an external media element encapsulates a specification that was composed in an external document format. This allows for the easy inclusion of existing documents into our model. What is encapsulated by the external media element is dependent on the external document format. However, external media elements should be employed with care as, e.g., each encapsulated format must be supported by presentation software.

The concepts of free variables and complex media elements guarantee reusability on the level of presentation fragments. External media elements allow for document format comprehensive reusability.

Finding a suitable component in a huge collection of reusable components is a difficult and time consuming task [Kru92]. In order to support the search and retrieval of reusable fragments, ZYX allows for the annotation of fragments

with *metadata*. Metadata is specified in form of key-value pairs which describe the content and technical characteristics of fragments. Metadata are exploited for adaptation (see below) and can be used by the repository to support authors in their search for reusable fragments.

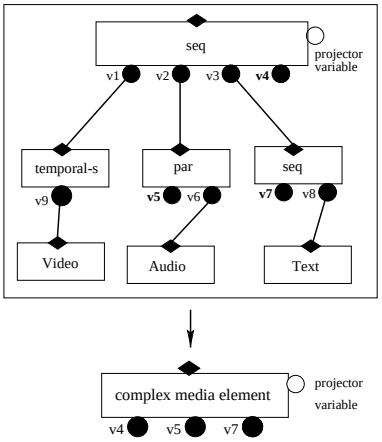


**Fig. 3.** Specification encapsulated in a complex media element

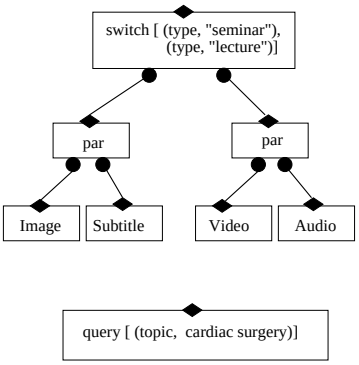


**Fig. 4.** Specification of presentation alternatives with the *switch* and the *query* element

Another means of reusability is provided by *selector elements*. Selector elements are operator elements that determine *what*, i.e., which part of a fragment, is presented. They can be used to select a specific temporal interval of a fragment using the *temporal-s* selector, e.g., a temporal interval of a video (see Figure 3), or a specific rectangular area by the *spatial-s* selector, e.g., a rectangular area of an image.

Besides the selector elements, the ZyX data model offers *projector elements* that influence the visual and audible layout in a presentation of a fragment. Projector elements determine *how* a specification is presented. They determine, for example, the presentation speed of a video or the spatial position of a video on the screen. Projectors can only be bound to *projector variables*. Each presentation element can have projector variables to which projector elements (and only projector elements) can be bound. This provides a clear separation between layout and structure of a fragment. By leaving projector variables unbound and using the export mechanism of complex media elements (see Figure 3), one can add different layouts to the same fragment at a later point in time. This allows for reusability of the same fragment in different presentation contexts.

### 3.2 Adaptation

Adaptation in the context of the ZYX model means that the multimedia document delivered to a user for presentation in response to a user query should best match the user's interest and the user's system environment collected in a user *profile*. This profile, i.e., metadata described by key-value pairs, captures characteristics that describe a user, topics of interest, presentation system environment, network connection, etc. Likewise, as mentioned above, each fragment is assigned metadata that describe its content.

The ZYX data model offers presentation elements to support adaptation to a user's profile by means of *switch elements* and the *query elements*. The switch elements allow to specify different alternatives for a part of a document. The alternative best matching the user profile is selected. An illustration of the switch element is given in Figure 4. At the time of presentation, either the left or the right subtree is presented depending on the type of teaching. A switch element can be employed if all alternatives are known to the author of the fragment.

However, it might be the case that the alternatives can be determined only at presentation time. For example, an author wants to determine that at a specific point in a medical presentation a digression to cardiac surgery is to be made but does not specify which fragments are relevant to this. This can be specified with a *query* element. The query is represented by a set of metadata. When the document is selected for presentation the query element is evaluated and replaced by the fragment best matching the metadata of the query. An illustration of the query element is given in Figure 4. The sample query element is the place holder for the fragment best matching the query with topic cardiac surgery.

### 3.3 Interaction

The mandatory requirement to support the modeling of interactive multimedia presentations is met by the data model's *interaction elements*. The model offers two types of interaction elements, *navigational interactive elements* and *design interactive elements*. Example for a navigational element is the *link* element that allows to specify hypertext structure. A *menu* element supports to interactively follow one presentation out of a set of presentations paths. The design interactive elements are the interactive versions of the projector elements. For example, for the typographic projector that allows to specify font, size, and style of a text, the *interactive typographic projector element* specifies that these settings can be carried out interactively when the document is presented.

### 3.4 Presentation-neutral representation

The usage of projectors does not imply that the layout is statically anchored in the document. As outlined before not all variables of a presentation element must be bound in the first place. They can be bound when the document is selected for presentation. This is also the point in time when the projector variables of a document can be bound to a set of projectors. This follows the idea of separating structure from layout information as can be found with SGML and XML.

# 4 The Object-Oriented Data Model for ZyX

The previous section has introduced the ZyX multimedia document model as a means to describe multimedia documents with regard to reuse, interaction, adaptation to user specific needs, and presentation neutral representation. In the following, we present an object-oriented, DBMS-independent conceptual model of ZyX using the notation of the Object Modeling Technique (OMT)[RBP+91], which serves as the basis for the IDS/UD-dependent logical database design.

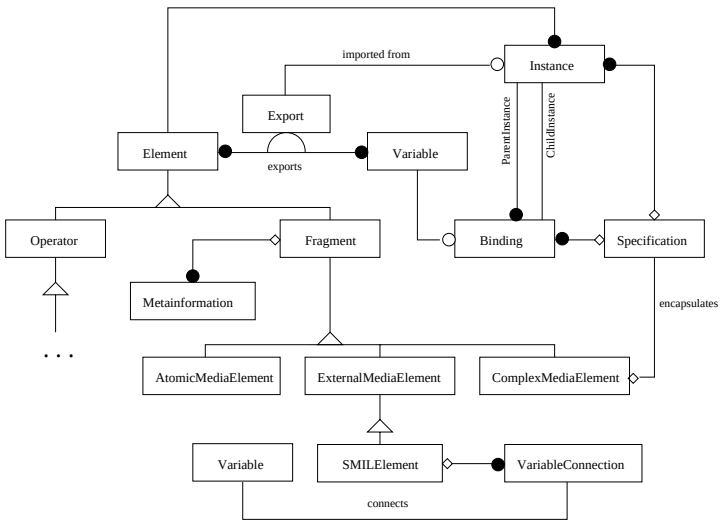


**Fig. 5.** Central classes of the conceptual design (OMT notation)

Figure 5 shows the central classes of the conceptual design. The classes and associations shown there are explained in the following subsections, advancing along the central classes `Element`, `Specification`, `Operator`, and `Fragment`.

## 4.1 Element

The class `Element` is the base class of the conceptual design. Every presentation element of the ZyX model is represented by a class which is derived from `Element`. `Element` defines attributes which allow for the unique identification of a presentation element. As explained in Section 3, variables and projector variables – modeled by `Variable` – can belong to a presentation element. This is modeled by the association `exports`. The cardinality of this association is $(m : n)$ because a variable can belong to several presentation elements. This happens if an unbound variable of a presentation element is exported by a complex media element encapsulating that presentation element. In this case that variable belongs to both the presentation element and the complex media element.

The association class `Export` is introduced to define an order on a presentation element's variables. Such an order is desirable to allow for selectively addressing the single variables and projector variables of a presentation element. In conjunction with the association `imported from` the class `Export` is also used to specify from which presentation element an exported unbound variable or projector variable of a complex media element originates.

### 4.2 Specification

The class `Specification` models ZYX specifications. The presentation elements used in such a specification and the bindings between them are modeled by aggregation of the classes `Instance` and `Binding` respectively. A direct aggregation of `Element` in `Specification` is not sufficient because one presentation element can occur several times in the same specification and each of these occurrences must be distinguishable from each other. Thus, `Instance` models the occurrence of a presentation element in a specification. A binding between the binding point of an instance of a presentation element and a variable associated with another presentation element which occurs in the same specification is modeled by the associations in which `Binding` participates.

### 4.3 Operator

The ZYX model divides presentation elements into operator elements and fragments. For each of these groups of presentation elements an abstract base class is defined. The class `Operator` forms the base class of the operator elements. For each operator element of the ZYX model a class is defined that is derived from `Operator`. This class has to define attributes specific to the operator element's parameters. For example, the class `Par` models the *par* operator element and defines the attributes `LipSync` and `Finish`.

### 4.4 Fragment

The class `Fragment` is the abstract base class of the fragments of the ZYX document model. As shown in Section 3, metadata organized by means of key-value pairs can be associated with fragments. This is modeled by the class `Metainformation` which represents key-value pairs. `Fragment` further defines means to name and describe a reusable fragment. Atomic media elements are represented by the class `AtomicMediaElement`. This class defines attributes to reference the media data encapsulated by an atomic media element and to describe the kind and format of the media data. The media data itself is not represented in the conceptual design. Hence, it does not matter where media data is stored. It can be stored in the database, in a file system, on a web server, or even in a remote database. This allows for an easy integration with a media server in future.

The class `ComplexMediaElement` models the complex media elements of ZYX. A complex media element encapsulates a ZYX specification. This is reflected in the aggregation `encapsulates`.

Finally, the class `ExternalMediaElement` models external media elements. Like `AtomicMediaElement` the class `ExternalMediaElement` defines means to specify the storage location of the external specification and to describe its format. As outlined in Section 3, the integration of external media elements in a ZYX specification depends on the external document format. Thus, for each supported document format a subclass of `ExternalMediaElement` has to be defined. At the moment, we support external fragments specified in SMIL modeled using the class `SMILElement`. A SMIL element can define so called *variable connections* which connect link targets of SMIL links to ZYX variables. If a SMIL link is activated during the presentation of a SMIL element and its link target is connected to a ZYX variable, the presentation element bound to the ZYX variable is presented. This is modeled by the class `VariableConnection`.

## 5 Features of IDS/UD

So far, we have performed a DBMS-independent object-oriented modeling of the ZYX model. In this section, we introduce the Informix Dynamic Server / Universal Data Option together with its advanced data modeling and extension concepts which we want to make use of during the logical database design phase described in Section 6.

IDS/UD is a so-called object-relational DBMS. Compared to a relational DBMS, IDS/UD includes object-oriented concepts and means to extend the DBMS with user-defined functionality. The most important extensions are presented in the following subsections.

### 5.1 Extended Data Types

In addition to the built-in atomic data types, like `INTEGER`, `DATE`, and `VARCHAR`, IDS/UD allows for the definition of so called *extended data types*. Four different kinds of extended data types can be distinguished: *collection data types*, *distinct data types*, *row data types*, and *opaque data types*.

A collection data type collects elements of the same data type in a group. Such a group can be a set, list, or a multiset. A distinct data type is basically an alias for a source data type and has the same storage representation but it is distinguished from the source data type by its name and cannot be substituted for the source data type without explicit type casting.

A row data type is a complex type which contains a group of data fields. Thus, a row data type closely resembles a record or structure in a programming language. A row data type can be named, called *named row type (NRT)*, or unnamed, called *unnamed row type (URT)*. An NRT has a unique name by which it can be globally identified while an instance of an URT corresponds

to a nameless tuple. NRTs can be associated with each other in a supertype-subtype relationship. A subtype inherits all data fields of its supertype and can define additional data fields. An instance of the subtype can be used anywhere an instance of the supertype is expected. Such a relationship is a kind of *intensional specialization*.

Finally, an opaque data type is a fundamental atomic data type. An opaque data type cannot be interpreted directly or subdivided into further components by the DBMS. An opaque data type is defined by a C structure. In order to meaningfully access an instance of an opaque data type, *access routines* must be defined, for example to write or read fields of the C structure or to compare two instances of the opaque data type.

### 5.2 Typed Tables

A typed table is a database table that has an NRT associated with it. For each data field of the NRT, a typed table defines a column of the same name and type. Analogously to the intensional specialization of NRTs, typed tables can be associated with each other to form a subtable-supertable relationship called *extensional specialization*. This is possible if the NRT associated with a subtable is a subtype of the NRT associated with its supertable. The effects of a subtable-supertable relationship are the following: The scope of a `SELECT` statement on typed table $t$ automatically includes all of its subtables if not explicitly prohibited by the `ONLY` keyword. Result tuples of any subtable are reduced to the columns of $t$. Likewise, the scope of `DELETE` or `UPDATE` statements on $t$ includes all of its subtables if not explicitly prohibited.

### 5.3 Routines

The functionality of IDS/UD can be extended by defining server-based routines. Such a routine resembles procedures or functions of imperative programming languages. Parameters of any data type can be passed to routines and a routine can return data of any type. At the moment, it is possible to implement routines using C and the *DataBlade API* or the *Informix Stored Procedure Language (SPL)*. A routine written in SPL is compiled by the DBMS to an intermediate format and stored at the server. Whenever the routine is invoked the DBMS retrieves and interprets the intermediate format. C routines are the only way to implement access routines for opaque data types. The C code of a routine is compiled using an ordinary C compiler and linked into a shared library. The path where this library is stored is registered at the DBMS. If the routine is invoked the DBMS loads the shared library and executes the routine. As the routine is executed in the address space of the DBMS this allows for a high execution speed but also for a misbehaving routine to crash the whole DBMS.

### 5.4 DataBlade Modules

A DataBlade module is a mean to flexibly extend the functionality of the IDS/UD at runtime. A DataBlade module collects database objects to forge an exten-

sion unit which can be installed (*registered*) and uninstalled (*de-registered*) in a database during operation of IDS/UD. Among the database objects that can be collected in a DataBlade module are: extended data types, routines, tables, indices, and error codes.

Informix offers extensive tool support for the creation and registration of DataBlade modules. *BladeSmith* allows to define the database objects included in a DataBlade module using a graphical user interface. From these definitions BladeSmith generates SQL scripts for registration and de-registration of the DataBlade module. The tool *BladeManager* visually handles the registration and de-registration of DataBlade modules in a database without interrupting the operation of the DBMS.

## 6 DataBlade Design

This section describes in detail our experiences with the mapping of the object-oriented model to the object-relational database schema of the IDS/UD, the implementation, stability, and extensibility of the solution.

So far, we have sketched the conceptual design of a database capable of managing ZYX documents and fragments in Section 4. Moreover, we have introduced the most important features IDS/UD offers in excess of relational technology in Section 5. Due to the flexibility of DataBlade modules we are now concerned with the exploitation of these features to map the conceptual design onto a logical design for a ZYX DataBlade module. In the following, we explain the mapping of the classes and associations of the conceptual design and the definition of routines.

### 6.1 Mapping of Classes to the Logical Design

Basically, opaque data types and named row types are suited to model classes of the conceptual design as both represent record-like structures. In order to decide between these two options we should consider that an opaque data type encapsulates its structure in a very strict way. In order to allow the DBMS access to any of its data fields access routines must be implemented. This results in a high implementation effort. Moreover, this can lead to performance problems if access to the data fields of opaque types occurs frequently because each access results in a routine call. However, in the case of the ZYX database, frequent access to attributes of classes is very likely. In contrast to opaque data types, named row types allow for direct access to their data fields by the DBMS itself.

Furthermore, the conceptual design includes a quite complex specialization hierarchy. While it is not possible to directly model specialization with opaque data types NRTs support the definition of intensional specialization relationships. In conjunction with typed tables extensional specialization can be modeled as well.

Regarding these facts, we have decided to use NRTs and typed tables as the means to directly map the classes of the conceptual database design onto the

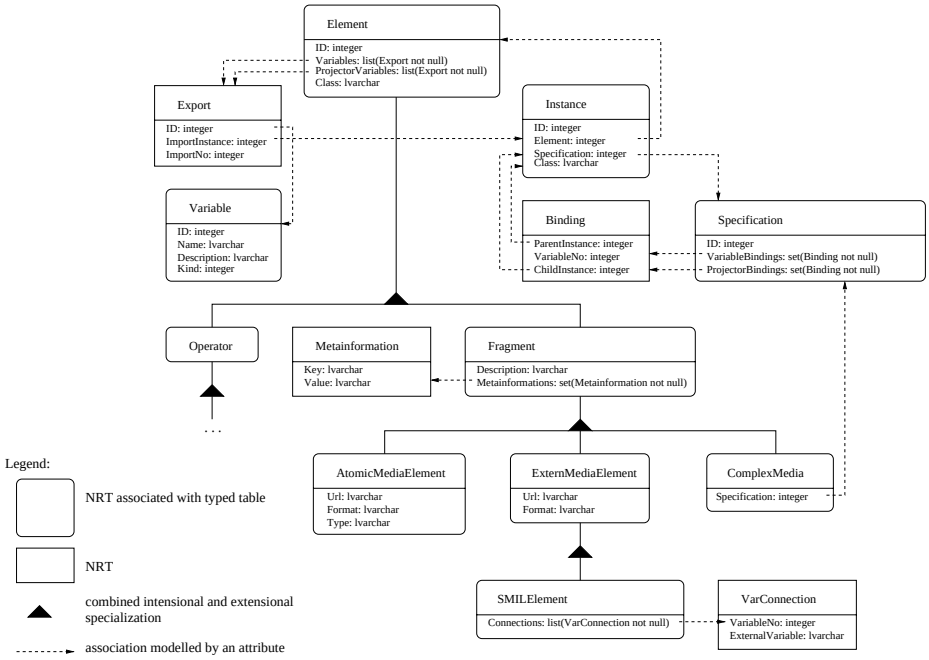


**Fig. 6.** Central logical design of the IDS/UD schema

logical design. Every class of the conceptual design is represented by an NRT bearing the same name as the class (see Figure 6). In addition to that, most named row types are associated with a typed table.

### 6.2 Mapping of Associations to the Logical Design

In the previous subsection, we have explained how the classes of the conceptual design are mapped to NRTs and typed tables of IDS/UD that form the logical design. In this subsection, we are concerned with the mapping of the associations of the conceptual onto the logical design. As IDS/UD offers no direct support for the modeling of associations, these have to be simulated by the use of data fields or additional tables.

For IDS/UD is an extended relational DBMS, associations of course can be modeled as in traditional relational DBMS. In particular, $(1:n)$- and $(1:n)$-associations can be mapped to additional data fields that act as foreign keys while a $(m:n)$-association can be mapped to an additional table containing two foreign keys, each one identifying an instance of the NRT which takes part in this association. During the logical design step of the ZyX DataBlade, we have followed this traditional relational approach when modeling bidirectional associations. Examples for such associations (see Figure 5) are the $(1:n)$-association between `Element` and `Instance` and the $(1:1)$-aggregation between `ComplexMediaElement` and `Instance`.

As already explained in Section 5, IDS/UD features collection data types. These can be used to model associations as well. An association is mapped to a collection typed data field of an NRT which takes part in that association. Considering an instance $i$ of that NRT, the collection typed data field contains all those instances of the NRT forming the other end of the association that are associated to $i$. The advantage of using collection data types over traditional relational modeling of associations is that a join is not needed to follow an association. However, due to the lack of a reference data type in IDS/UD, the elements of a collection-typed data field are stored in the data field and not in a separate table. Hence, this kind of association modeling is suited for unidirectional associations only. An example for such an association is the $(1:n)$-aggregation between `Fragment` and `Metainformation` which is modeled by an additional data field `Metainformation` of type `set{Metainformation not null}` in `Fragment`. Figure 6 summarizes the modeling of the associations of the conceptual design.

### 6.3 Definition of Routines

As mentioned above, IDS/UD allows for the definition of server-based routines. This subsection explores how this ability of IDS/UD can be favourably used in the design of the ZYX DataBlade. The definition of routines is motivated by several reasons. First of all, the logical design developed in the previous subsections is quite large. This means that even standard tasks prove to be complex. For example, in order to create a presentation element, an instance of the appropriate named row type has to be created. Then, for each variable associated with the presentation element, an instance of the NRT `Variable` must be created. Furthermore, for each of these variables an instance of the NRT `Export` must be constructed and inserted into the list of exported variables of the presentation element. The creation of a complex media element is even more difficult because its list of exported variables is not known in advance but rather calculated from the unbound variables of the encapsulated specification. The definition of routines that handle the creation of presentation elements can significantly reduce the complexity and error-proneness of this task. Other reasons to define routines are introduced by the ZYX model itself. There are some presentation elements which have been defined with a server-based implementation in mind. For instance, the *query* operator element models server-based adaptation. Thus, it is reasonable to define a routine providing this kind of adaptation. Moreover, the ZYX model has been defined to allow for a presentation-neutral description of documents. With such a description it is possible to convert a document to other document formats like SMIL or MHEG-5. The conversion functionality can be implemented using server-based routines, too. Finally, the architecture of a multimedia database system as proposed by [KA97,BKL96] calls for the integration of the presentation and playout of media content in the DBMS itself. As Cardio-OP evolves, we will evaluate how server-based routines can be used to contribute to this integration.

The routines developed in our project so far can roughly be divided into three groups: *construction/ destruction* of presentation elements and specifications,

*navigation* through specifications, and *conversion* to external document formats. In the following, these groups are described in more detail.

**Construction/ Destruction.** Basically, routines that support the creation or deletion of presentation elements, and the construction or modification of specifications belong to this group. For each presentation element of the ZYX document model a construction routine is provided. Parameters specific to an presentation element are passed as arguments to its construction routine. For example, the routine `createPar` creates a *par* operator element in the database including all variables. As presentation element specific parameters, `createPar` takes the arguments `LipSync` and `Finish`.

The user-defined routines `createInstance`, `addInstance`, `bindVariable`, and `bindProjVariable` can be employed to create ZYX specifications while the routines `deleteElement` and `deleteInstance` serve to delete a specific presentation element or an instance of a presentation element from the database.

**Navigation.** Routines that allow to navigate through a specification fall into this category. For instance, the routine `getRoot` retrieves the root instance of a specification. Using the routines `followVariable` and `followProjVariable` one can follow a binding of variable or a projector variable to another instance of a presentation element in a specification.

A navigation routine in a broader sense is `query`. Given an instance of a *query* operator element this routine retrieves the best matching fragment from the database according to the metadata specified by the *query* operator element.

**Conversion.** The last group of routines is concerned with the conversion of ZYX documents to external document formats. So far, a routine named `exportXML` has been realized to provide an export facility to an XML-based [BPSM98] representation of ZYX documents that will be used by viewer software currently under development. A similar export facility to the SMIL multimedia document format is under development.

## 7 Experiences

This section deals with the practical experiences we gained during the implementation of the logical design of the ZYX DataBlade module. The implementation involved two major steps. First, the database schema including all NRTs, typed tables, and routine signatures was defined using the tool BladeSmith. In the second step, we implemented the routine bodies using the language C.

From the experiences we made during the first step we can safely, say that BladeSmith significantly reduces the complexity of creating DataBlades. DataBlade developers are relieved to a great extent of the burden of manually defining registration and de-registration scripts in SQL which is a complex and error-prone task. Not only the creation and deletion of the database objects included

in a DataBlade has to be specified but also the dependencies among these. However, during the development process, we encountered some situations in which BladeSmith[2] did not generate correct registration and de-registration scripts. In particular, BladeSmith has got difficulties in correctly ordering the definitions of named row types. At several points in the registration script, named row types which have not been defined yet are used in the definition of other data types. Furthermore, BladeSmith generates wrong SQL syntax when faced with NRTs that have a supertype but do not define additional data fields. Thus, we had to manually modify the generated SQL scripts. Because it is not possible to incorporate these modifications back into BladeSmith, iterative development is difficult to perform. The problems with BladeSmith do not seem to be unsolvable, however, so future versions of BladeSmith are likely to fix them.

From the second phase, we have learned that the efforts needed to implement C routines for use with DataBlades should not be underestimated for several reasons. As already mentioned, the DataBlade API offers access to IDS/UD at a very low semantical level. On the one hand, this low semantical level allows to control many aspects of IDS/UD. As an overview, this API offers support for querying, the smart handling of large objects, the definition of custom secondary access methods, file handling, thread handling, collection data type handling, exception handling, callbacks, and memory management. On the other hand, due the low semantical level even routines performing simple tasks grow large. Moreover, it is possible for routines to crash the DBMS. Thus, extra care must be taken to test and validate them.

Summarizing, we can say that though IDS/UD still needs some improvement in terms of tool and debugging support it is well-suited to model complex data structures as given in most multimedia applications. From the fact that the DataBlade API allows to control many low-level aspects of IDS/UD, we are confident that we can extend IDS/UD by support for efficient content-based retrieval, delivery, and presentation of multimedia information.

## 8 Conclusion and Future Work

Starting out with the requirements of the Cardio-OP project, which calls for the support of *reusability*, *adaptation*, and *presentation-neutral description* of the structure and content of multimedia documents, we first introduced the overall project idea.

With the needs for managing, retrieving, and delivering multimedia data and the multimedia documents we discussed the options of selecting suitable database system technology. We presented an introduction to the ZYX multimedia document model that meets the project specific requirements. As we come to the conclusion that we will follow an object-relational approach, we presented the specific modeling features of the selected object-relational database system Informix Dynamic Server / Universal Data Option. We presented the mapping of

[2] Version 3.40.TC1

the object-oriented conceptual model of the ZYX model to the object-relational database schema exploiting the specific modeling features of the ORDBMS.

We described the experiences we gained during the implementation of the ZYX model as a DataBlade module of the object-relational database system IDS/UD under Sun Solaris, following the architectural framework initially presented in [KA97,BKL96]. Due to the fact that we can describe the ZYX model in terms of an XML DTD, we could also think about storing ZYX documents in an SGML/XML-capable database system in the future, following the approach taken in [BAK97].

Ongoing and future work includes the identification and realization of possible optimizations of the implementation of the ZYX DataBlade and the integration of content-based retrieval, delivery of continuous media data and interactive, adaptive presentation of ZYX documents. Currently, we are working on a generic Java-based presentation engine for ZYX documents supporting continuous MPEG video delivery based on an extension of the L/MRP buffer management technique [MKK95]. Further work is needed to extend the representation of metadata, its usage for querying the Cardio-OP repository taking into account the various approaches discussed in, e.g., [SK98], and to exploit appropriate indexing techniques. We also work on the flexible, on-the-fly composition of multimedia fragments in order to create individualized multimedia documents, and for the realization of adaptation and personalization of multimedia presentations depending on the user environment specified by means of user profiles.

*Acknowledgments.* We would like to thank the anonymous reviewers for their valuable comments to improve the paper.

## References

[BAK97] K. Böhm, K. Aberer, and W. Klas. Building a Hybrid Database Application for Structured Documents. In *Multimedia - Tools and Applications*, volume 5, pages 275–300, Dordrecht, 1997. Kluwer Academic Publishers.

[BK99] S. Boll and W. Klas. ZYX — A Semantic Model for Multimedia Documents and Presentations. In *Proceedings of the 8th IFIP Conference on Data Semantics (DS-8): "Semantic Issues in Multimedia Systems"*. Kluwer Academic Publishers, Rotorua, New Zealand, 5-8 January 1999.

[BKL96] S. Boll, W. Klas, and M. Löhr. Integrated Database Services for Multimedia Presentations. In S. M. Chung, editor, *Multimedia Information Storage and Management*. Kluwer Academic Publishers, Dordrecht, 1996.

[BPSM98] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. *Extensible Markup Language (XML) 1.0 – W3C Recommendation 10-February-1998*. W3C, URL: http://www.w3.org/TR/1998/REC-xml-19980210, February 1998.

[CC96] S. T. Campbell and S.M. Chung. Database Approach for the Management of Multimedia Information. In *[NBT96]*, 1996.

[Cha98] D. Chamberlin. *A complete guide to DB2 Universal Database*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1998.

[Chu96] S. M. Chung, editor. *Multimedia Information Storage and Management*. Kluwer Academic Publishers, 1996.

[HBB+98] P. Hoschka, S. Bugaj, D. Bulterman, et al. *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification – W3C Recommendation 15-June-1998.* W3C, URL: http://www.w3.org/TR/1998/REC-smil-19980615, June 1998.

[IBM98] IBM. *DB2 Universal Database*, 1998. URL http://www.ibm.com.

[Inf97] Informix. *DataBlade Developers Kit User's Guide, DataBlade API Programmer's Manual, Informix Guide to SQL: Tutorial.* Informix Press, Menlo Park, 1997.

[ISO92] ISO/IEC. *Information Technology - Hypermedia/Time-based Structuring Language (HyTime), ISO/IEC IS 10744*, 1992.

[ISO95] ISO/IEC JTC1/SC29/WG12. *Information Technology – Coding of Multimedia and Hypermedia Information – Part 5: Support for Base-Level Interactive Applications, ISO/IEC IS 13522-5.* ISO/IEC, 1995.

[KA97] W. Klas and K. Aberer. Multimedia and its Impact on Database System Architectures. In P.M.G. Apers, H.M. Blanken, and M.A.W. Houtsma, editors, *Multimedia Databases in Perspective.* Springer, London, 1997.

[KL97] G. B. Koch K. Loney. *Oracle 8: The Complete Reference (Oracle Series).* Oracle Press, 1997.

[Kru92] C. W. Krueger. Software Reuse. *ACM Computing Surveys*, 24(2), June 1992.

[MKK95] F. Moser, A. Kraiß, and W. Klas. L/MRP: A Buffer Management Strategy for Interactive Continuous Data Flows in a Multimedia DBMS. In *Proceedings VLDB 1995.* Morgan Kaufmann, USA, 1995.

[NBT96] Kingsley C. Nwozu, P. Bruce Berra, and B. Thuraisingham, editors. *Design and Implementation of Multimedia Database Management Systems.* Kluwer Academic Publishers, 1996.

[Ora98] Oracle. *Oracle8*, 1998. URL http://www.oracle.com/st/products/uds/oracle8/-html/oracle8.html.

[Paz97] P. Pazandak. Evaluating ODBMSs for Multimedia. *IEEE Multimedia Journal*, 4(3), 1997.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, et al. *Object-Oriented Modeling and Design.* Prentice-Hall, Englewood Cliffs, 1991.

[RKN96] T. C. Rakow, W. Klas, and E. J. Neuhold. Research on Multimedia Database Systems at GMD-IPSI. *IEEE Multimedia Newsletter*, 4(1), April 1996.

[RNL95] T. C. Rakow, E. J. Neuhold, and M. Löhr. Multimedia Database Systems – The Notions and the Issues. In G. Lausen, editor, *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW)*, Reihe Informatik Aktuell, pages 1–29. Springer-Verlag, Berlin, 1995.

[SJ96] V. S. Subrahmanian and S. Jajodia, editors. *Multimedia Database Systems-Issues and Research Directions.* Springer, 1996.

[SK98] A. Sheth and W. Klas. *Multimedia Data Management - Using Metadata to Integrate and Apply Digital Media.* McGraw-Hill, New York, 1998.