# SMOOTH – A Distributed Multimedia Database System

Harald Kosch, Alexander Bachlechner, László Böszörményi, Christian Hanin,
Christian Hofbauer, Margit Lang, Carmen Riedler, Roland Tusch
Institute of Information Technology, University Klagenfurt, Austria
Email Correspondence to : harald.kosch@itec.uni-klu.ac.at

**Extended version of the short paper appeared in the** *International Very Large Database Conference 2001*, **pp. 713-714.**

## 1 Introduction

A multimedia database system deals with the storage, manipulation and retrieval of digitally representable information objects such as text, images, video and audio [1]. Providing mechanisms that allows the user to retrieve desired multimedia information by their semantic content is now an important issue in multimedia database systems. However, current products (e.g., Oracle 9i *inter*Media and Informix Datablade Modules) allow us to index, exclusively, low-level features of multimedia objects, usually not encoded into attributes provided by the database schema. Therefore special techniques are needed for semantic indexing and retrieval of multimedia objects [2].

In this context we present the **SMOOTH** system, a prototype of a distributed multimedia database system. It implements an integrated querying, annotating, and navigating framework relying on our generic video indexing model *VIDEX*. The model has been described in a previous paper [3]. The framework allows the structuring of videos into logical and physical units and the annotation of these units by typed semantic objects. A meta-database stores these structural and semantic information. We provide further a clear concept for capturing and querying the semantic content of multimedia objects, their correlation with low-level objects, as well as their temporal relationships. In particular, it allows users, on the one hand, to navigate through semantic content not apparent at the surface and, on the other hand, to submit complex queries whose results can be used for subsequent decision making.

Next Section 2 states the significance of the contribution. Section 3 describes the demonstrated prototype *SMOOTH*. Finally, Section 4 presents another *SMOOTH* application for orthopedic operations.

## 2 Main Contributions of SMOOTH

**1) Integrated Model for Low- and High-Level Video Indexing**
SMOOTH relies on an integrated model for low- and high-level video indexing. This model takes advantage of related approaches (see the overview in [4] and our previous paper [3]) and extends them by the introduction of a) means for structuring video steams and b) genericity in the indexing process. The core of the index model defines base classes for an indexing system, while application specific classes are added by declaring subclasses (content classes) of the base classes.
We share with the Multimedia Description Schemes (MDS) of MPEG-7 [5] the manner how to extend base classes to a specific application domain, the general purpose is, however, different from that of MPEG-7. The main goal of MPEG-7 is the standardization of descriptions for communications, while we are focusing on modeling of video data supported by a meta-database. For further information on the commonness and differences of MPEG-7 and our data model, please refer to [6].

**2) Domain Transparency**
'Domain Transparency' means that the client software automatically adapts to (correct) extensions made by a new application domain to the base classes. Thus, the GUI of our client implementation is built dynamically from the meta-database. A special alias table allows the display of the menu-points in different languages.

**3) Cache for SQLResults**
'Domain Transparency' could result in high traffic network with the database, as every attribute name, label of the GUI ought to be read out of the database, if no supplemental measures were undertaken. In order to manage this problem, we have introduced a cache for attribute names and aliases (i.e., the labels, menu items of the GUI). This reduces the amount of exchanged SQLResults considerably.

**4) Integrated Querier, Annotator, Navigator**
The client integrates a querier, annotator and navigator interface in a common framework and defines a smooth interaction of the different interfaces in order to allow an effective working with the video material. It realizes an **open architecture** in a way that it enables the integration of existing tools during the video indexing process, e.g., automatic and manual methods for segmenting video streams or methods for extracting and tracking low-level visual objects.

# 3 Brief Overview over the Framework Functionality

The SMOOTH prototype[1] of a distributed multimedia database system integrates a query, navigation, and annotation framework for video media material driven by the content of a meta-database. This meta-database implements the semantic and structural part our multimedia data model *VIDEX*, presented in [3].

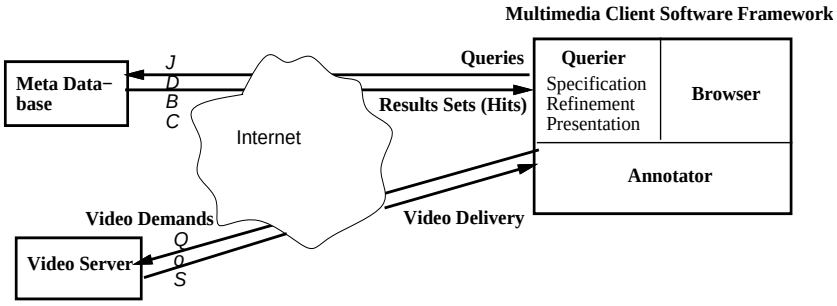


Figure 1: *Distributed multimedia system scenario.*

Fig. 1 displays the general systems' architecture of *SMOOTH*. The video server provides selective access to the physical video streams. We currently use the Oracle Video Server with the supported protocol types UDP and RTP.

The core of the meta-database are the base classes: *events*, *objects*, *persons* and *locations*. These classes are subclasses of a general *ContentObject* which may refer to low-level motion objects. Application specific classes are added by declaring subclasses (content classes) of the base classes. Furthermore, the meta-database contains classes for a detailed structuring of video streams and for relating the instances of the semantic classes to the media segments.

Fig. 2 shows the rich set of content classes we defined for a soccer application. For instance, we distinguish events like goals, penalty, etc., and persons like players and referees.

**Querier, Presentation Manager and Annotator** The JAVA-based client provides means to annotate, query and navigate through video material using the graphical library Swing. The core components of the client include the annotator, querier, presentation manager and navigator. The *annotator* allows the structuring of the video into segments and the annotation of high-level content objects. The *querier* follows a text-based, structured query specification technique. It enables the definition of video queries by specifying conditions on content objects and the specification of semantic and temporal relations to other content objects. The results of a query are presented in a compact form, i.e., server address, length, video description etc. In the *presentation* manager, the user can compose presentations, with temporal constraints, from the query results. The *navigator* allows the navigation through the contents of the meta-database.

One of the main characteristics of the client software is the 'Domain Transparency', i.e., the client builds its interfaces dynamically from the specific application part of the meta-database with the help

[1] A previous version of *SMOOTH* was presented in the demonstration session at the ACM Multimedia Conference 2000 [7].

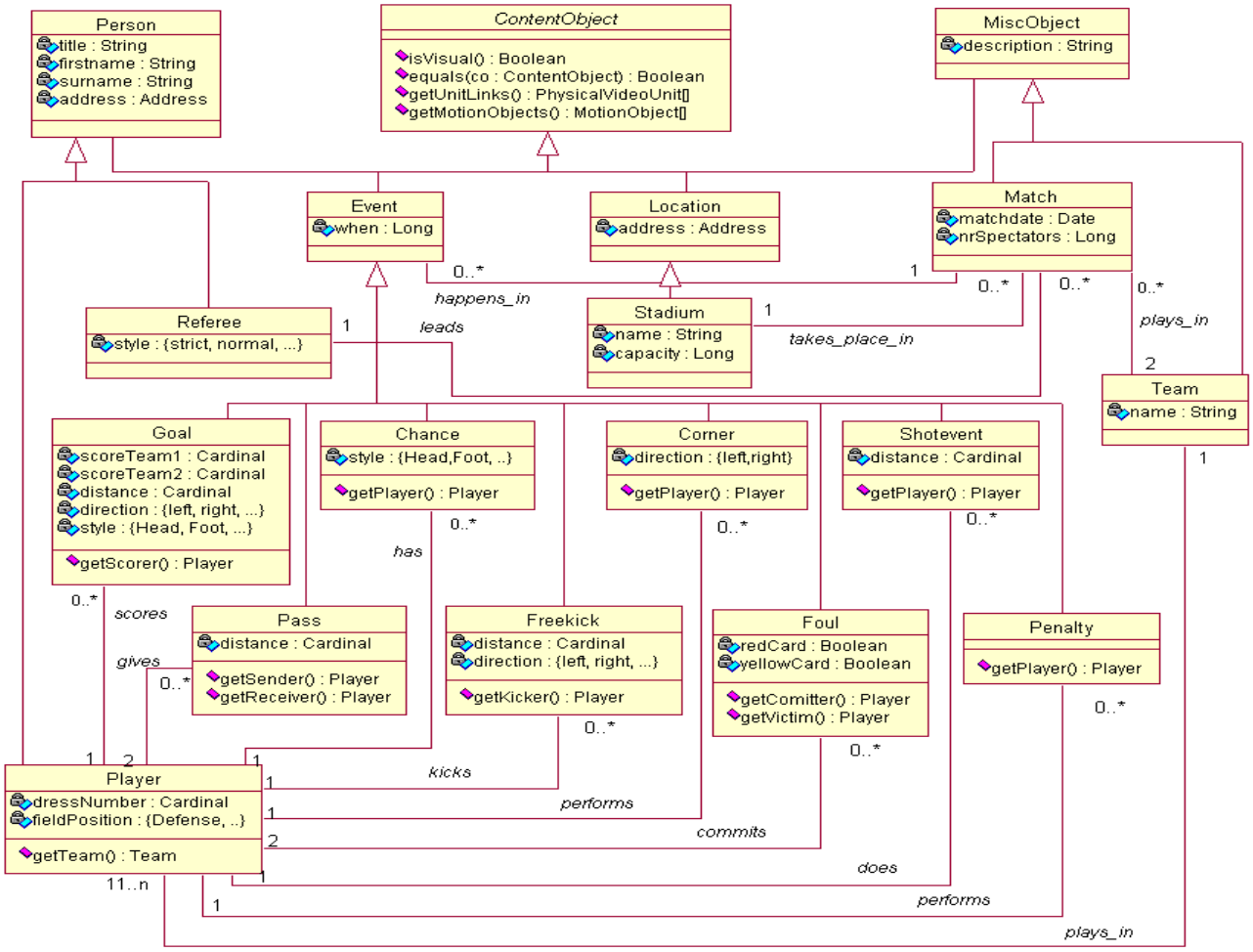


Figure 2: *Data model in UML : Soccer-specific content classes.*

of the Meta-ResultSet primitives of the JDBC API 2.0. For instance, a user likes to retrieve all video shots showing a certain player. The application presents first all subclasses of 'Person' by following the inheritance hierarchy in the database. Once a subclass, let's say 'Player' is selected, the user has to enter (or to reuse) at least the values of all non-NULL attributes of the *Player* table from the meta-database. Based on the database type of the attributes, a rigorous type check is engaged which helps the user in the data input (e.g., for the specification of a date).

**Querier** The user is able to submit query conditions through a combination of fill-in forms, menu selections and text fields. Furthermore query refinement, query storage and replay mechanism are provided.

For instance, in our soccer application, a query like : 'Find all video shots, where the player Carsten Jancker scores a goal after having received a pass over 30 meters', can be specified. Fig. 3 shows an extract of the *SMOOTH* query interface which allows the specification of this query. We need to declare two condition sets, one for the subclass 'Player' of 'Person' and one for the subclasses of 'Goal' and 'Pass' of 'Event'. Especially, the latter conditions require the specification of a temporal constraint, i.e., the goal should strictly occur after the pass. This temporal constraint between the two events 'Goal' and 'Pass' is entered in the 'Event' dialog, which is central in the query interface of Fig. 3.

**Annotator** Implementing the genericity and expressiveness of *VIDEX* has, of course, its price. The price to pay is the complex manual semantic annotation. We, therefore, provide in *SMOOTH* an annotation tool to help the user to annotate the videos and to refer to already annotated units for reuse of information. For instance, Fig. 4 shows a typical annotation situation. In the left window, the already annotated segments are displayed allowing the definition of larger segments, e.g., a 'Scene' can be built from two 'Shots'. Besides, text fields and menu selections are provided to enter the content information for the events.

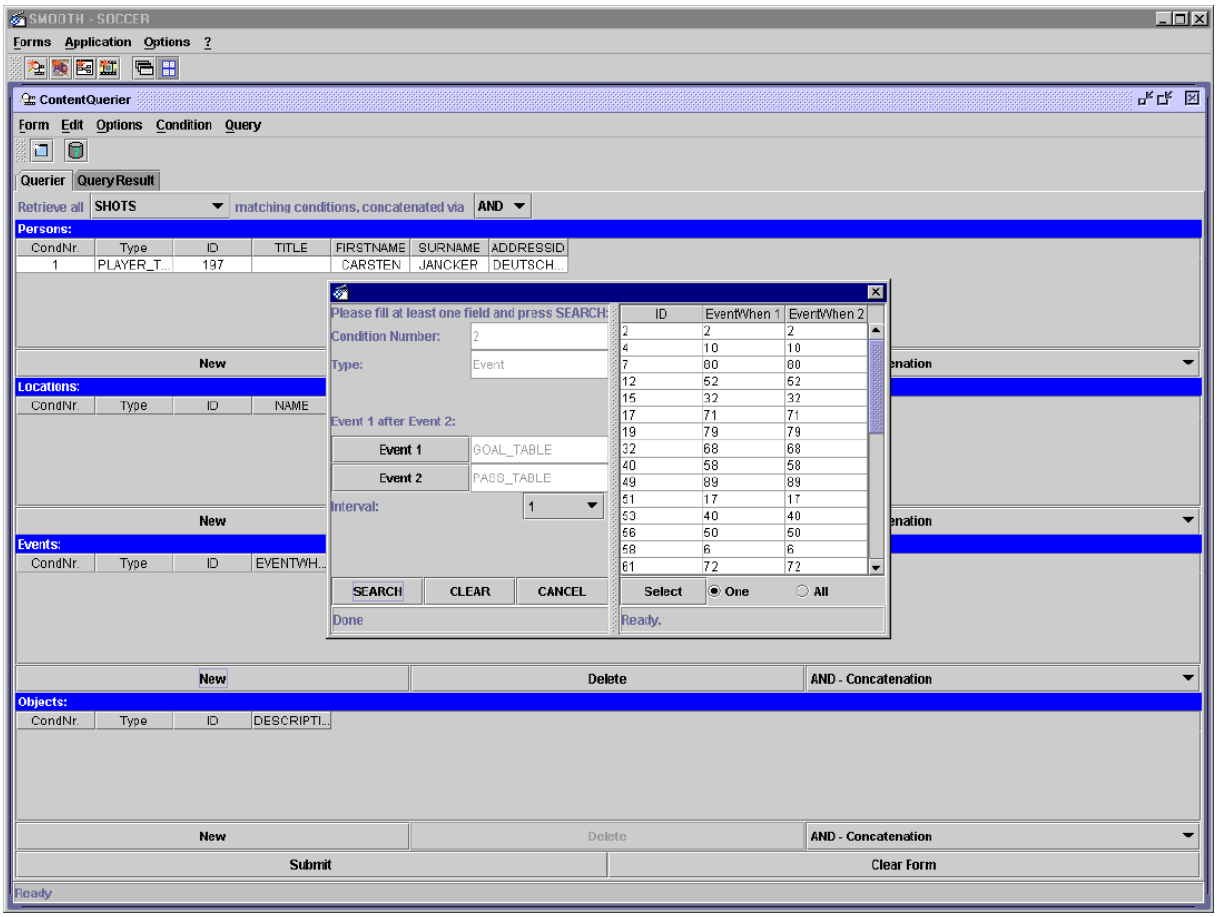


Figure 3: *Query Dialog for the specification of the temporal constraint.*

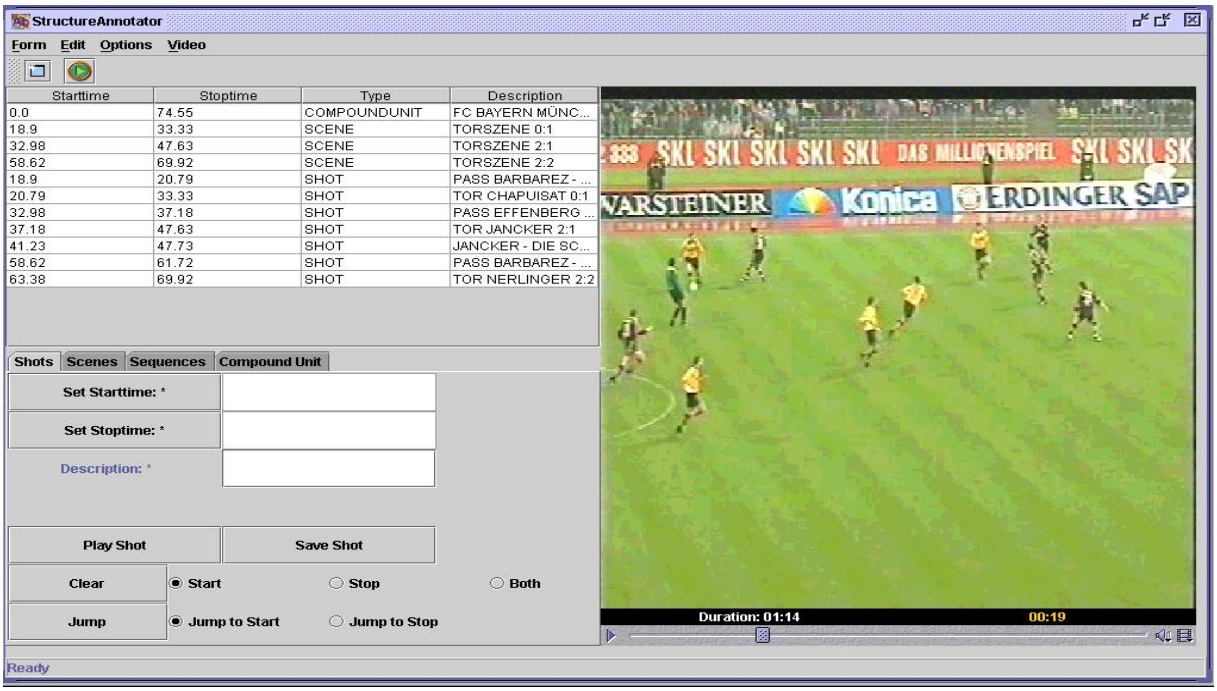


Figure 4: *Extract of the Annotator frame in SMOOTH.*

We also provide a framework for the inclusion of low-level shot/scene detection algorithms in order to allow a pre-selection of interesting video segments for annotating. In Fig. 5 the shot detection frame of the annotator is shown. The shot detection is based on a simple comparison of the pixel color values of two subsequent frames of a modified input video. The modification of the input video is done by a shape analysis, i.e., the input to the shot detection is a video reduced to its shape information. With this modification we enhanced the precision of the video shot detection. The detection process is implemented using the Java Media Framework (JMF) and its Plug-In mechanism. Similar methods have successfully been employed in the *ADVENT* project, for more information consider please [8].

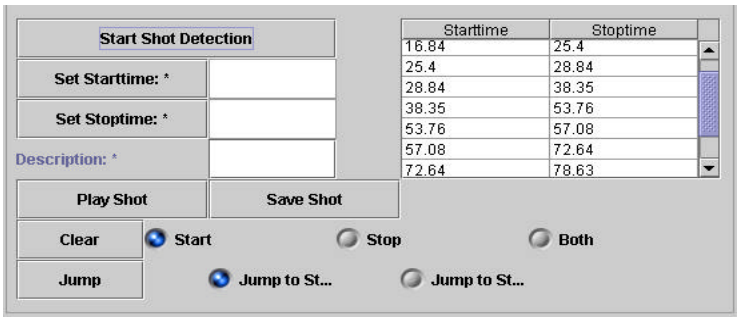


Figure 5: *Video Shot Detection frame of the Annotator.*

The workflow of the annotation is as follows: first, the user starts a video detection. After its completion, shots are processed from the list of shot propositions (a sample list is shown in the right window of Fig 5). Since the shot detection cannot discover the exact shot boundaries of a goal, or a pass, the boundaries have to be readjusted manually by using the time line of the player and the buttons 'Set Starttime' or 'Set Stoptime'. Upon the fixing of the boundaries, an event type should be provided. This may be a 'Goal', 'Pass', 'Penalty', etc. in the soccer application. As the contents of the *SMOOTH* interfaces are schema-driven, respective possibilities are read out from the meta-database and are proposed to the user. This step could be automated to some extent, for example, a proposition of an event could be made based on some event extraction mechanism. Then all relevant fields of the event have to be filled in. This could be simple attributes or links to other content classes. In the latter case, already annotated instances can be reused. Here, once again, support from automatic extraction may be integrated, for example, one might identify similar objects or persons occurring in the shot actual under consideration from an already annotated video scene and propose pre-annotations to the user. However, the effort of automatic indexing has to be balanced with its benefit. For instance, if all players of the soccer's league have been annotated beforehand, the user is surely faster in picking out the right player (if the player is well known) from the pre-annotations than any recognition software.

**Presentation Manager** The query results can be composed to a presentation in the dialog presentation manager shown in Fig. 6. In the upper left corner, the query results are displayed as beams with respective length, in a pool ready to be used for a presentation composition. These query results are then dragged and dropped from the pool to be arranged along a time chart.

Users may specify complex presentation scenarios by imposing temporal constraints on the video delivery. Temporal constraints comprise *precedence constraints* related to the ordering of the videos and *delay constraints* related to the waiting time between the videos. For instance, in Fig. 6, we specified that the second video has to be presented 8s after the start of the first video. Both videos shall run then in parallel until the first video stops.

The server addresses, ports, stream identifiers and time intervals of the requested video units are kept by the result sets of the content-based video queries. The temporal constraints are resolved by respective synchronized calls to the JMF-primitives. The videos are shown in players of the presentation manager dialog. Fig. 6 snapshots a scene where both videos are running.

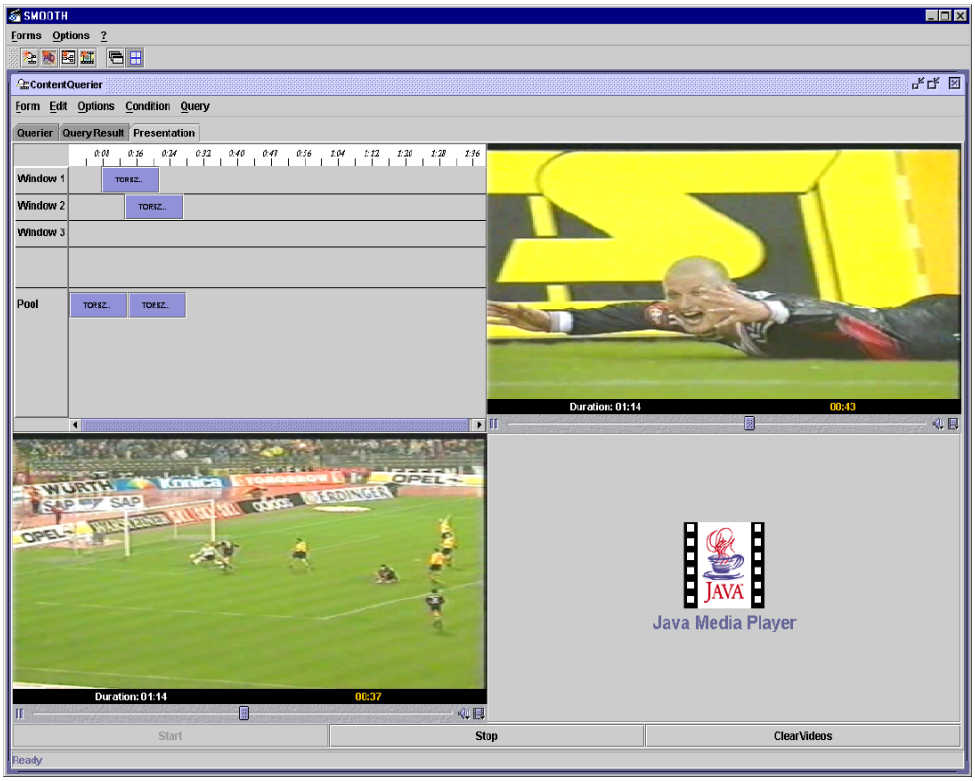


Figure 6: *Presentation Manager in SMOOTH.*

# 4 Indexing Orthopedic Operations

Other *SMOOTH* applications are orthopedic operations: artificial hip joint replacement, lumbar interbody fusion, and screwing of the titanium block. The annotations have been developed in collaboration with orthopedic surgeons from Poland [9].

The *object* class comprises an *operation* and a *patient*. The patient is a passive subject of the operation. For the *patient* we need her/his case history. Additional information like age are also useful. The *operation* attributes are more informative: a reference to the patient, the date and the method of operation and some information about possible complications. Name tags of medical staff participating in the operation are also included.

| Event | Description |
|---|---|
| 1 | Anesthesia |
| 2 | Placement of the patient |
| 3 | Ischaemia |
| 4 | Preparation of operating field |
| 5 | Covering of the field |
| 6 | Incision |
| 7 | Joint dislocation |
| 8 | Preparation of bony placenta |
| 8.1 | The hip joint acetabulum |
| 8.1.1 | Removing the femoral head |
| 8.1.2 | Reaming the acetabulum |
| 8.2 | The femoral shaft |
| 8.2.1 | Preparing the femoral canal |
| 9 | Implantation |
| 9.1 | Inserting the acetabular component |
| 9.2 | Inserting the femoral stem |
| 10 | Attaching the femoral head |
| 11 | Implant repositioning |
| 12 | Insertion the drain |
| 13 | Suturing |

Table 1: *Stages of the operation concerning the artificial hip joint replacement*

The *event* specifies stages of the operation. They are shown in Table 1. For example, incision is a part of each operation. In this case the following information are required: the name tag of the physician who made the cut, the type of approach, the course of incision, the incision area, the layer, the names of the tools that have been used, and (optional) the complications. Another example can be the preparation of the bony placenta, for which similar information are required. At

present, about thirty types of events are implemented. The complete schema can be found at `http://www.itec.uni-klu.ac.at/~smooth/images/DBS.pdf`.

## Final Remarks

Finally, we would like to remark that the meta-database of *SMOOTH* is realized with the *Oracle 9i* DBMS. However, since we used standard SQL-statements, migration to other DBMSs supporting the JDBC API 2.0 is possible.

Further information on *SMOOTH* may be obtained from the SMOOTH homepage at `http://www-itec.uni-klu.ac.at/~smooth`.

## References


[1] V. S. Subrahmanian. *Principles of Multimedia Database Systems*. Morgan Kaufman Press, January 1998.

[2] S.-C. Chen, R.L. Kashyap, and A. Ghafoor. *Semantic Models for Multimedia Database Searching and Browsing*. Kluwer Press, 2000.

[3] R. Tusch, H. Kosch, and L. Böszörményi. VideX: An integrated generic video indexing approach. In *Proceedings of the ACM Multimedia Conference*, pages 448–451, Los Angeles, USA, Oct.-Nov. 2000.

[4] H. Jiang, D. Montesi, and A. K. Elmagarmid. Integrated video and text for content-based access to video databases. *Multimedia Tools and Applications*, 9(3):227 – 249, 1999.

[5] José M. Martínez. *Overview of the MPEG-7 Standard (v5.0)*. ISO/IEC JTC1/SC29/WG11 N4031, March (Singapure Meeting) 2001. `http://www.cselt.it/mpeg/standards/mpeg-7/mpeg-7.html`.

[6] Harald Kosch. MPEG-7 and Multimedia Databases. *SIGMOD Records*, 2001. Accepted for Publication.

[7] H. Kosch, R. Słota, J. Kitowski, D. Nikolow, S. Podlipnig, and K. Breidler. MMSRS - multimedia storage and retrieval system for a distributed medical information system. In *HPCN 2000 Conference*, pages 517–524, Amsterdam, Netherlands, May 2000. Springer Verlag, LNCS 1823.

[8] D. Zhong and S.-F. Chang. Structure analysis of sports video using domain models. In *IEEE Conference on Multimedia and Exhibition*, Tokyo, Japan, August 2001.

[9] H.Kosch, R. Slota, L. Böszörményi, J. Kitowski, and J. Otfinowski. An information system for long-distance cooperation in medicine. In *Proceedings of the 5th International Workshop on Applied Parallel Computing, New Paradigms for HPC in Industry and Academia (PARA)*, pages 233–241, Bergen, Norway, June 2000. Springer Verlag, LNCS 1947.