Entwicklung und Einsatz einer Videodatenbank im WWW – ein Erfahrungsbericht

Hermann Körndle
TU Dresden
Institut für Psychologie IV
Weberplatz 5
01062 Dresden
koerndle@rcs.urz.tu-dresden.de

Ulrich Marder
Universität Kaiserslautern
Fachbereich Informatik
Postfach 3049
67653 Kaiserslautern
marder@informatik.uni-kl.de

Günter Robbert
Otto-Friedrich-Universität Bamberg
Fakultät Sozial- und
Wirtschaftswissenschaften
96045 Bamberg
robbert@pi1.sowi.uni-bamberg.de

Zusammenfassung

In einer Kooperation zwischen Anwendern und Entwicklern wurde eine Videodatenbank einschließlich WWW-Benutzerschnittstelle prototypisch implementiert. Einsatzgebiet der Datenbank sind experimentelle, WWW-basierte Teleteaching-Systeme, so daß spezielle Anforderungen aus diesem Umfeld zu erfüllen waren. Dieser Beitrag stellt die von den Autoren gewählte Lösung vor, die auf einem herkömmlichen relationalen Datenbanksystem, verschiedenen Internet-Videoservern und einem Webserver mit Server-Side-Scripting-Unterstützung basiert. Die bei der Entwicklung gemachten Erfahrungen führen zu einigen Verbesserungsvorschlägen, z. B. Einführung von automatischer Lastverteilung und Information-Retrieval-Funktionen, und Realisierungsalternativen, z. B. Einsatz von Java oder objekt-relationalen DBMS, die jedoch beim heutigen Stand der Technik meist noch sehr aufwendig umzusetzen sind.

1 Einführung

Die Anschaulichkeit und das Verständnis dynamischer Sachverhalte wird durch Video- und Filmdokumente erheblich gefördert. Deswegen stellen sie eine wichtige Materialkomponente in Forschung und Lehre verschiedener wissenschaftlicher Disziplinen, wie z. B. der Verhaltensbiologie, der Psychologie, der Medizin, etc. dar. Sie dienen der Speicherung, Analyse und Präsentation zeitlich veränderlicher Sachverhalte. Die Verwendung von Video- und Filmdokumenten in Forschung und Lehre ist allerdings mit hohem Aufwand verbunden, der dazu führt, daß solches Material in der Lehre selten, beim Selbststudium fast gar nicht verwendet wird. Im wesentlichen erschweren folgende Gründe den Einsatz:

Aufwand bei der Beschaffung: Im Vergleich zu Büchern oder Zeitschriften sind Video- und Filmmaterialien schwer zu beschaffen und wenig verfügbar, da kaum gut ausgestattete Mediatheken mit entsprechenden (Online-)Katalogen zur gezielten Suche nach themenspezifischem Material existieren.

Aufwand bei der Auswahl: Videos und Filme bestehen aus einer sequentiellen Anordnung vieler Einzelszenen, auf die ein wahlfreier Zugriff nicht oder nur mit speziellen Hilfsmitteln möglich ist.

Aufwand bei der Präsentation: Film- und Videomaterial läßt sich nur mit spezieller Hardware präsentieren. Außerdem erfordern unterschiedliche Formate entsprechend kompatible Wiedergabegeräte.

Vor allem für eine moderne Lehre und ein effizientes Studium wäre es wünschenswert, Videomaterial ohne allzu großen Aufwand ort- und zeitunabhängig in einem vernetzten Informationssystem speichern und mit geringem Aufwand abrufen zu können. Da ein für solche Einsatzzwecke geeignetes

Informationssystem nicht handelsüblich ist, wurde es in enger Zusammenarbeit von Informatikern und Nutzern aus dem Fach Psychologie spezifiziert, entwickelt und eingesetzt. Das Kernelement dieses Informationssystems ist eine Videodatenbank, auf die über ein Rechnernetzwerk zugegriffen werden kann. Im folgenden werden zunächst die Anforderungen an ein solches Informationssystem und dann einzelne Schritte der dazu erfolgten Entwicklungsarbeit dargestellt.

Der Rest des Papiers gliedert sich wie folgt: Kapitel 2 stellt die Anforderungen der Anwender aus der Psychologie an die Videodatenbank dar. Kapitel 3 beschreibt den Prototypen der Videodatenbank. Kapitel 4 bemüht sich um eine Bewertung des Prototypen, gibt Erfahrungen wieder, die während der Entwicklung gemacht wurden, und beschreibt Möglichkeiten zur Weiterentwicklung. Kapitel 5 diskutiert einige Realisierungsalternativen, und Kapitel 6 beschließt das Papier mit ersten Erfahrungen beim Einsatz der Videodatenbank.

2 Anforderungen der Anwendung an die Videodatenbank

Studieren bedeutet, verschiedene Lehr- und Lernmaterialien zu einem Themenbereich zu beschaffen, zu bearbeiten, zu hinterfragen und miteinander in Beziehung zu setzen. Deshalb ist bei der Entwicklung und Konfiguration eines Videodatenbanksystems für Lehre und Studium zu beachten, Film- und Videomaterial nicht nur komplett zu digitalisieren und unter einem Dateinamen abzulegen, sondern zu indexieren und wohlorganisiert zu speichern. Des weiteren ist eine Dokumentenarchitektur zu wählen, die eine spätere Erweiterung und Verknüpfung mit anderen zusätzlichen Lehr- und Lernmaterialien ermöglicht. Prinzipiell sollte es möglich sein, für ein Dokument folgende das Dokument charakterisierende Metadaten festzulegen:

Logische Struktur: Um welches Material handelt es sich?

Inhaltsstruktur: Aus welchen Inhalten setzt sich das Dokument zusammen?

Layout-Struktur: Wie sollen die Inhalte präsentiert werden?

Semantische Struktur: An welcher Stelle ist das Dokument inhaltlich in ein Themengebiet einzuordnen?

Relationale Struktur: Welche Verknüpfung gibt es zu anderen Dokumenten bzw. Dokumentensystemen?

Dies setzt eine Dokumentenarchitektur voraus, die nach dem Prinzip der Entkopplung von Logik-, Layout- und Inhaltsstruktur aufgebaut ist [Flu96].

Die für eine in Lehre und Forschung einsetzbare Videodatenbank notwendigen strukturellen Komponenten, die für das Lernen und Studieren erforderlichen Bedienungsmöglichkeiten sowie die Hardwareanforderungen lassen sich aus der Beschreibung der Anwendung für die Datenbank feiner spezifizieren. Bei der Entwicklung des Systems ist auch zu beachten, daß der Beschaffungs-, Auswahl- und Präsentationsaufwand für die Dokumente möglichst gering bleibt.

1. Videodatenbanken sollen die *Ablage, Verwaltung und Pflege von Videosequenzen* ermöglichen. Direkt an das Bildmaterial gekoppelt müssen Metadaten, wie z. B. Autor, Inhalt, Dauer, Schlagworte, etc. mit abgelegt und verwaltet werden können. Zu berücksichtigen ist, daß Videosequenzen sich in ihrer Länge, in ihrer Kodierung (z. B. MPEG, AVI, RealVideo), in ihrem Kompressionsgrad und in der Bildgröße unterscheiden. Die Datenbank muß außerdem den einfachen Austausch, die Veränderung und Ergänzung der Videosequenzen und ihrer Metadaten ermöglichen.
2. Die in der Datenbank abgelegten Videosequenzen müssen ähnlich wie in Literaturdatenbanken übersichtlich nach unterschiedlichen Ordnungsmerkmalen angeordnet werden können, um den Studierenden einen effizienten Zugriff auf die Szenen über ihre Inhaltsbeschreibung zu ermöglichen.
3. Der *Abruf von Videosequenzen* sollte über vernetzte Rechner möglich sein, wobei über Clients in einem LAN oder WAN auf die Datenbank zugegriffen wird, die auf einem oder auf mehreren Servern abgelegt sein kann. Nach dem Aufruf der Videodatenbank-Anwendung sollten die Nutzer

über ein Inhaltsverzeichnis die gewünschte Sequenz auswählen oder über eine Schlagwortsuche finden können. Zur Präsentation des Materials müssen die üblichen Playerfunktionen wie Start, Stop, Pause, schneller Vor- und Rücklauf und Zeitlupe zur Verfügung stehen. Eine weitere ergänzende Funktion sollte das Erstellen von Playlisten ermöglichen, mit Hilfe derer Folgen von Videosequenzen abgerufen werden können. Für die Verwendung von Videosequenzen beim Studieren ist es außerdem von großer Bedeutung, die Videosequenzen nicht nur abrufen, sondern über Annotationen und Notizen mit anderen Studiermaterialien, die über das Netz verfügbar sind, verknüpfen zu können.

4. Videosequenzen sollten unter möglichst wenig Hardware-Restriktionen verfügbar sein. Als Clients sollten deshalb übliche Personalcomputer für das Abspielen der Videosequenzen verwendet werden können.

Der Erfolg beim Einsatz einer Videodatenbank in Studium und Lehre wird wesentlich davon abhängen, daß bei der Entwicklung möglichst alle an ein solches System zu richtenden Anforderungen berücksichtigt werden.

3 Realisierung der Videodatenbank

3.1 Rahmenbedingungen

Der Rahmen für die Entwicklung der Videodatenbank ist durch die am Institut für Psychologie vorhandene Ausstattung an Hard- und Software vorgegeben. An Hardware sind dort etwa ein Dutzend PCs zu finden, die fast ausschließlich unter dem Betriebssystem *Windows NT* betrieben werden. Zwei dieser Rechner werden dediziert als Server eingesetzt und konnten im Rahmen dieses Projekts als Datenbankserver, als WWW-Server und als Server für Videofilme genutzt werden.

Zu Beginn des Projekts wurde eine grobe Aufwandsabschätzung durchgeführt. Der Aufwand für die Durchführung des Projekts wurde hierbei mit insgesamt vier Mannmonaten veranschlagt, der sich jeweils zu gleichen Teilen auf die Entwicklung der Videodatenbank und auf die Implementierung der Benutzerschnittstelle verteilt.

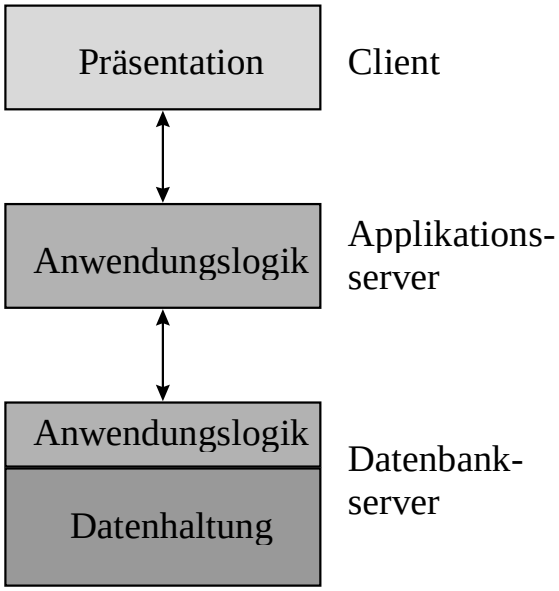

Abbildung 1: 3-tier Client/Server Architektur mit verteilter Anwendungslogik

3.2 Softwarearchitektur

Als Grundprinzip wurde eine dreistufige Client/Server-Architektur gewählt (Abbildung 1). Diese bietet gegenüber zweistufigen Client/Server-Architekturen den Vorteil der besseren Skalierbarkeit. Da Applikationsserver und Datenbankserver unabhängig voneinander replizierbar sind, lassen sie sich optimal im Netz plazieren, um potentiell Hunderten von Clients höchstmögliche Dienstgüte und Verfügbarkeit zu bieten.

Moderne Datenbankverwaltungssysteme (DBVS) bieten die Möglichkeit, Teile der Anwendungslogik in die Datenbank zu integrieren. Die Mittel hierfür sind z. B. *Trigger* und *Stored Procedures*. Diese werden auch in der Videodatenbank zur semantischen Konsistenzerhaltung und zur Vereinfachung der eigentlichen Anwendungsentwicklung eingesetzt (z. B.: Generierung von Default-Eintragungen, kaskadierendes Löschen etc.), wodurch sich die in Abbildung 1 gezeigte Verteilung der Anwendungslogik ergibt.

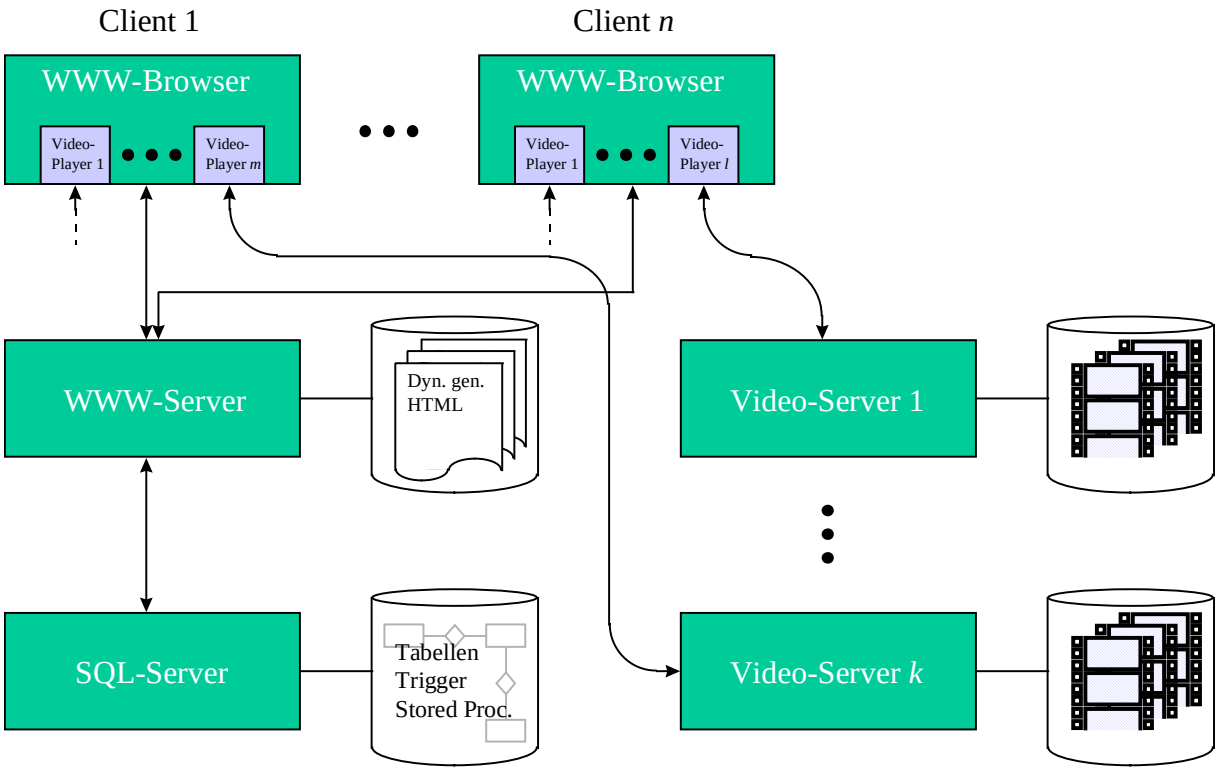

Abbildung 2: Softwarearchitektur der Videodatenbank

Abbildung 2 zeigt eine konkretere Darstellung der Architektur. Hier wird zunächst deutlich, daß es sich tatsächlich nicht um eine reine dreistufige Client/Server-Architektur handelt. Der Grund ist, daß Standard-Videoserver, wie sie hier verwendet werden sollen, für eine solche Architektur nicht geeignet sind. Daher kommunizieren die Clients ohne Umweg über den Applikationsserver direkt mit den Videoservern.

Des weiteren sieht man, daß für die Client-Seite entsprechend der Anforderungen keine speziell zu entwickelnde Software vorgesehen ist. Es genügen Standard-WWW-Browser und Videoplayer. Damit werden Installation, Administration und Wartung der (Standard-) Client-Software vollständig in den Verantwortungsbereich des Endanwenders verlagert.

Als Applikationsserver kommt ein WWW-Server zum Einsatz. Die Anwendungslogik wird dementsprechend größtenteils durch Programme realisiert, die dynamisch HTML-Seiten generieren. Der "klassische" Weg ist hier der Einsatz von CGI-Programmen. Dieser hat jedoch einige gravierende Nachteile, u. a.:

- Die Wartung der Programme wird schon bei geringer Komplexität und Anzahl sehr aufwendig und schwierig.
- Bei jedem Aufruf eines CGI-Programms wird ein neuer Prozeß erzeugt, worunter Performanz und Skalierbarkeit leiden.
- Der Sitzungskontext ist nur sehr umständlich und damit fehleranfällig realisierbar.

Für die Videodatenbank wurde daher ein anderer, wenngleich weniger portabler Weg beschritten (s. u.).

Die Aufgabe des SQL-Servers wird von einem relationalen Datenbanksystem übernommen. Vorteile dieser bewährten Standardtechnologie sind vor allem in der hohen Robustheit und Verfügbarkeit sowie in den kurzen Entwicklungszeiten (dank hervorragender Werkzeugunterstützung) zu sehen, während Nachteile vor allem auf das relativ einfache relationale Datenmodell zurückzuführen sind (vgl. Kap.5.2).

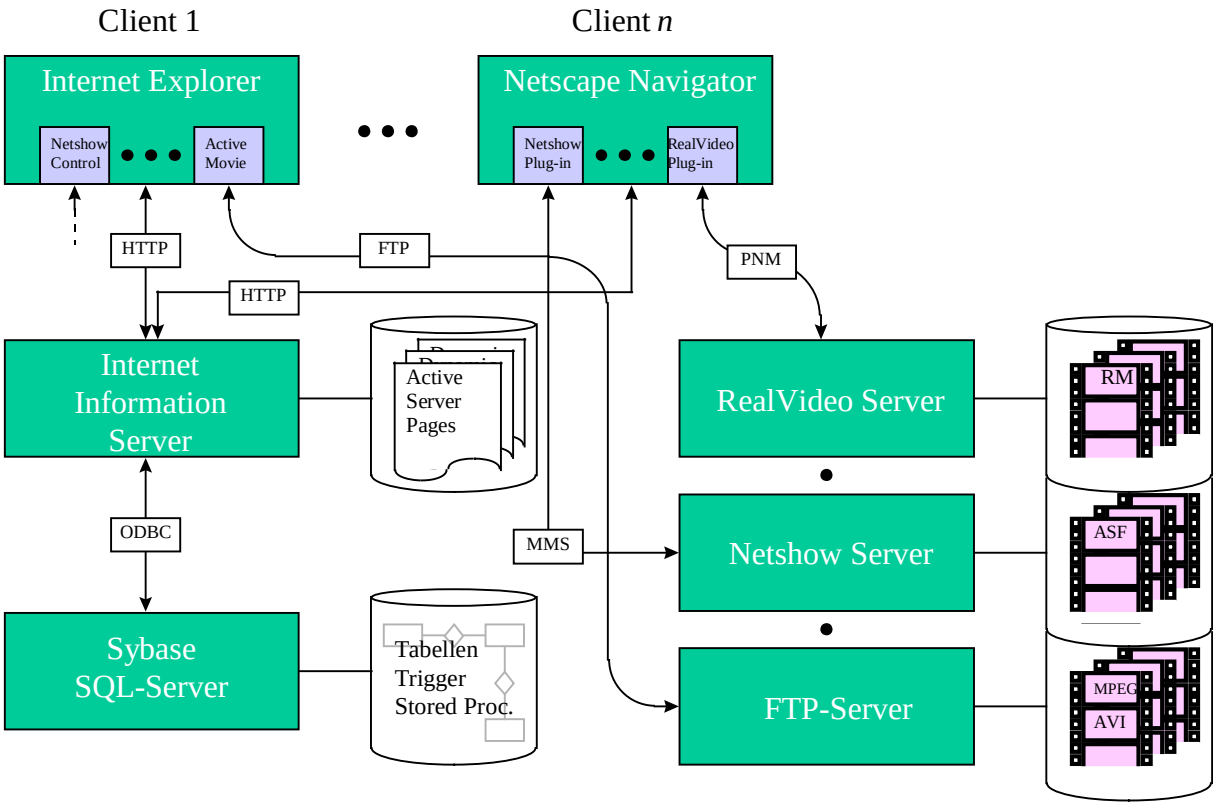

Abbildung 3: Instanziierung der Videodatenbank-Softwarearchitektur mit aktuellen kommerziellen Produkten (MMS und PNM sind proprietäre Protokolle für *Continuous Media Streaming*)

Abbildung 3 zeigt, wie die Architektur mit aktuellen kommerziellen Produkten instanziiert werden konnte. Mit Ausnahme des Sybase SQL-Servers wurden nur lizenzgebührenfreie Produkte ausgewählt (teilweise sind leistungsfähigere gebührenpflichtige Versionen verfügbar, z. B. von RealVideo [Rea97]).

Aus den weiter oben genannten Gründen wurde statt CGI auf die *Server-Side-Scripting*- Technik des Internet Information Servers gesetzt [MS97b]. Diese von Microsoft als *Active Server Pages* (ASP) bezeichnete Technik ist (genau wie vergleichbare Konkurrenzprodukte) proprietär, bietet dafür aber folgende Vorteile:

- HTML und Skriptsprache können gemischt verwendet werden. Dadurch können z. B. Benutzereingaben (Formulare) leicht verarbeitet und Datenbankanfrageergebnisse leicht für die Präsentation aufbereitet werden.
- Es können alle bereits vorhandenen, ebenso wie selbst zu entwickelnde *ActiveX Automation Server* [Ses98, S. 45ff.] eingebunden werden (z. B. *Active Data Objects* [MS97c], [Ses98, S. 355-360] für den Zugriff auf Datenbankserver). Damit ist man für die Programmierung der Anwendungslogik nicht auf die relativ geringe Mächtigkeit und Performanz einer Skriptsprache beschränkt.
- Die Verwaltung des Sitzungskontexts ist weitgehend automatisiert, z. B. können Datenbankverbindungen leicht für die gesamte Anwendersitzung offengehalten und Anfrageergebnisse für wiederholte Verwendung zwischengespeichert werden.
- Alle Skripts einer Anwendung werden im selben Prozeß ausgeführt.
- Es sind gute Programmierwerkzeuge, z. B. Codegeneratoren, verfügbar.

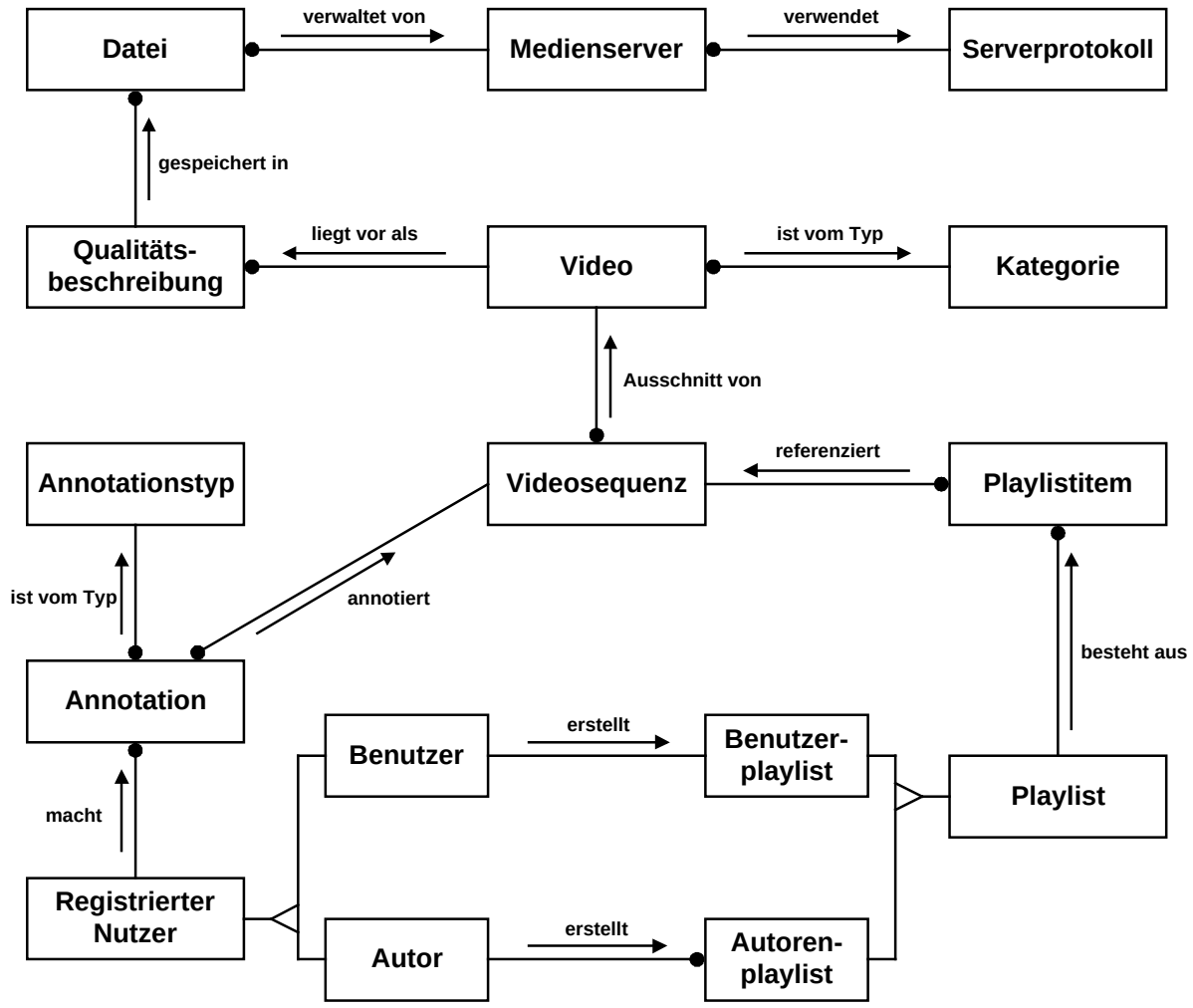

Abbildung 4: Konzeptionelles Schema der Referenzdatenbank

3.3 Design der Datenbank und der Benutzerschnittstelle

Schon zu Beginn der Entwurfsphase der Videodatenbank wurde festgelegt, daß die relationale Datenbank ausschließlich als Referenz- bzw. Meta-Datenbank für Videofilme dient. Dies bedeutet, in der relationalen Datenbank werden nur Informationen *über* Videofilme gespeichert. Die Videofilme selbst befinden sich nicht in der Referenzdatenbank, sondern werden von speziellen Videoservern verwaltet (vgl. Abbildung 2).

Da die Nutzer *persönliche* Daten (i. e. Annotationen und Playlisten) in der Videodatenbank ablegen können sollen, ist in der Referenzdatenbank eine Benutzerverwaltung erforderlich. Entsprechend der beim Lernen üblichen Unterscheidung zwischen *Lehrern* und *Lernenden* wird auch hier zwischen zwei verschiedenen Arten von Nutzern unterschieden, die im folgenden (etwas weniger anwendungsspezifisch) als *Autoren* respektive *normale Benutzer* (kurz: Benutzer) bezeichnet werden.

Das konzeptionelle Schema der Referenzdatenbank ist in Abbildung 4 dargestellt (Notation: OMT [RB90]). Hier wird die Unterscheidung zwischen Autoren und normalen Benutzern explizit modelliert. Somit können beide Nutzergruppen über öffentliche WWW-Schnittstellen mit der Datenbank arbeiten, wobei der Applikationsserver die Berechtigung zur Nutzung der wesentlich mächtigeren Autorenschnittstelle überprüfen kann, ohne daß hierfür spezielle Rechner- (WWW-Server) oder Datenbank-Accounts (SQL-Server) für die Nutzer erforderlich wären.

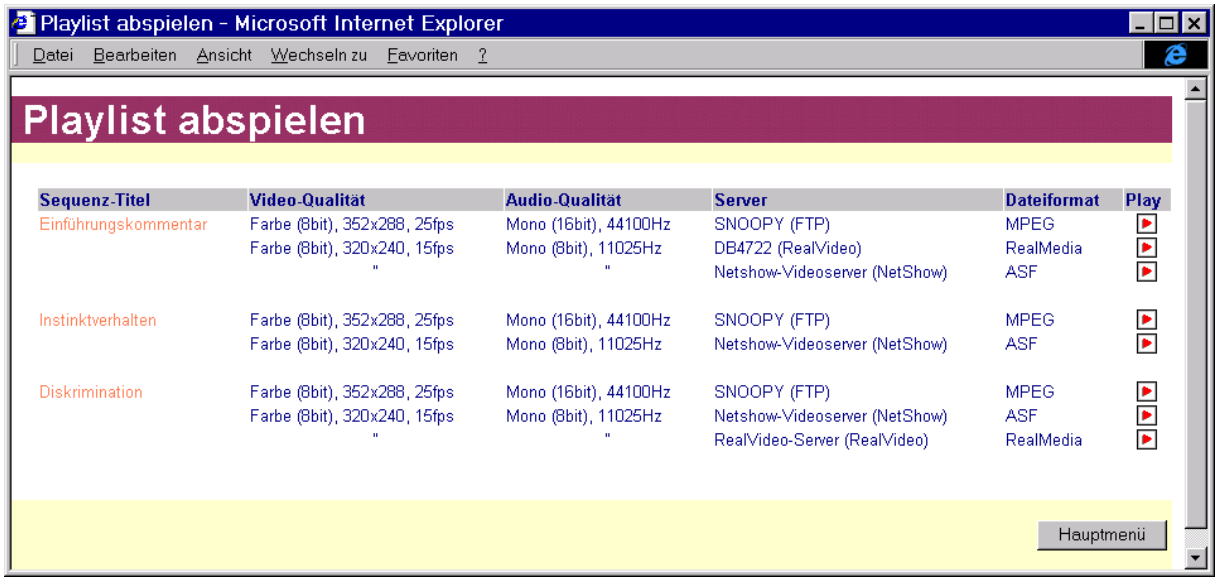

Abbildung 5: Darstellung einer Playliste an der Benutzerschnittstelle

Für *Administratoren* der Videodatenbank ist keine spezielle WWW-Schnittstelle vorgesehen. Das Schema in Abbildung 4 sieht daher keine explizite Modellierung dieser Benutzergruppe vor.

In den folgenden Abschnitten wird das Datenbankschema und die Benutzerschnittstelle aus Sicht der verschiedenen Benutzergruppen dargestellt und erläutert. Hierbei werden im Text die jeweils angesprochenen Klassen des Schemas durch Verweise der Form (→ `Klasse`) referenziert. Insbesondere von der Benutzerschnittstelle können leider nur die wichtigsten Teile vorgestellt werden. Dem interessierten Leser (mit hinreichender technischer Ausstattung) steht jedoch die Möglichkeit offen, die WWW-Schnittstelle unter der Adresse “http://db4727.inf.tu-dresden.de/” selbst zu testen.

3.3.1 Benutzersicht

Dem normalen Benutzer (Lernenden) präsentiert sich die Videodatenbank in erster Linie in Form von Playlisten (→ `Playlist`), die mit Ausnahme der persönlichen Playliste (→ `Benutzerplaylist`) des Benutzers von Autoren (Lehrern) nach thematischen Gesichtspunkten zusammengestellt werden. Er kann diese sog. Autorenplaylisten (→ `Autorenplaylist`) abspielen, aber nicht verändern. Playlisten werden zum Abspielen in Form einer Tabelle dargestellt, siehe Abbildung 5. Hierbei werden zu jedem Titel (→ `Playlistitem, Videosequenz`) alle verfügbaren Alternativen aufgelistet, die sich z. B. in der Qualität (Bildgröße, Tonqualität, ...) oder dem Format (MPEG, ASF, ...) unterscheiden. Bei der Entscheidung für oder gegen eine Alternative sind vor allem folgende Kriterien zu beachten:

- Ist ein für das Format geeigneter Videoplayer installiert?
- Sind lokal genügend Ressourcen verfügbar (Temporärspeicher, Prozessor- und Grafikleistung, ...)?
- Reicht die Netzwerkleistung aus?

Wird eines der Videos zum Abspielen ausgewählt, dann wird eine HTML-Seite wie die in Abbildung 6 angezeigt. Diese ist in zwei Ausgabe- und einen Eingabebereich unterteilt. Im ersten Ausgabebereich wird das Video angezeigt und im zweiten die Annotationen (→ `Annotation`) zum Video – sekundengenau während des Abspielens. Der Eingabebereich dient zum Anlegen neuer Annotationen. Durch Betätigen eines speziellen Schalters “Annotieren” wird das Video angehalten und die aktuelle Position in das Annotationsformular übertragen. Danach kann die Annotation eingegeben und an die Videodatenbank gesendet werden.

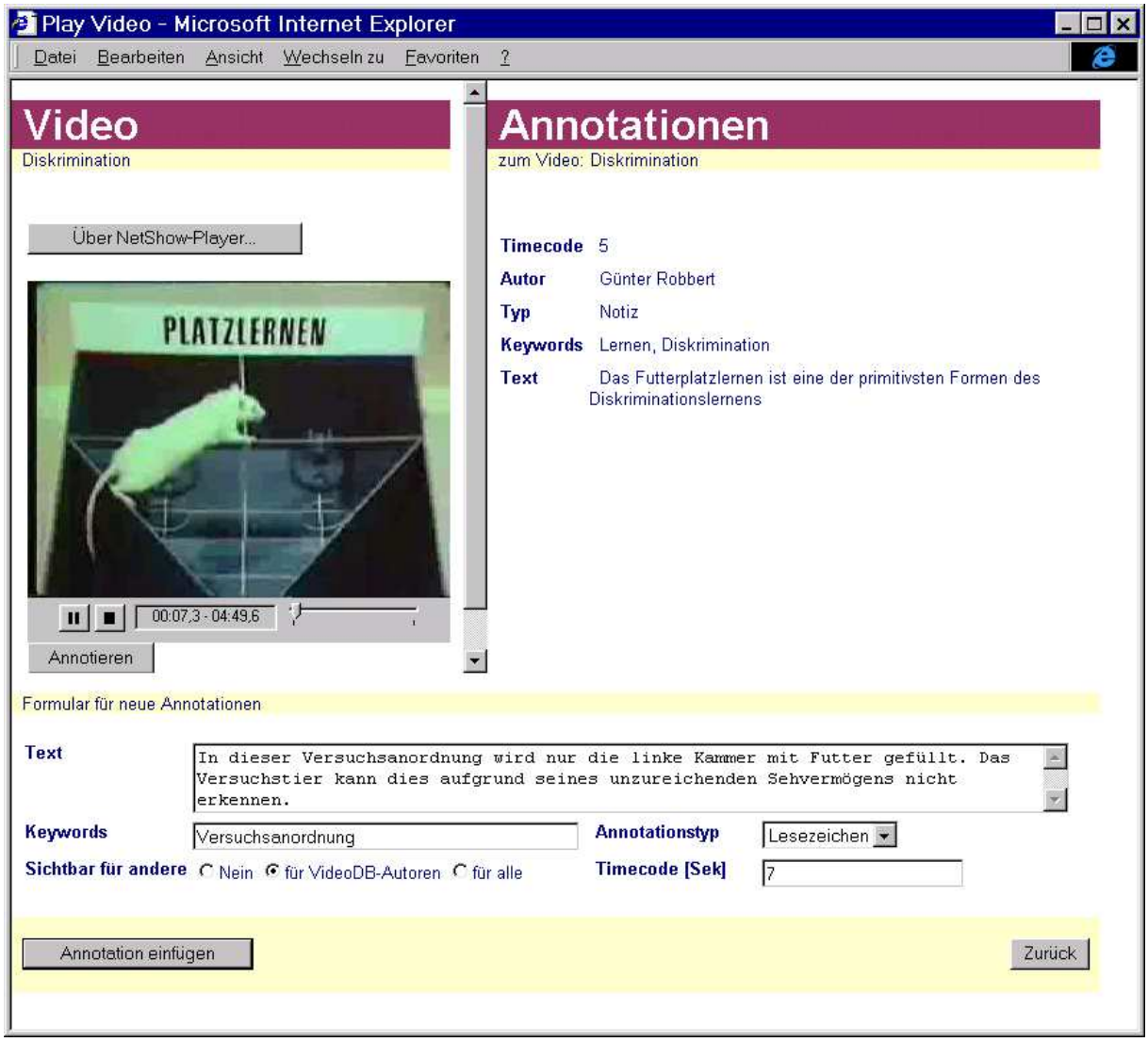

Abbildung 6: Abspielen und Annotieren eines Videos

Eine Annotation gehört automatisch dem Benutzer, der sie angelegt hat, und wird dauerhaft in der Datenbank gespeichert, bis sie von ihm selbst oder einem Administrator wieder gelöscht wird. Beim Anlegen einer Annotation kann er zwischen drei verschiedenen Zugriffsschutzebenen für den Lesezugriff auswählen:

- Eine *private* Annotation ist nur für den Besitzer sichtbar,
- eine *interne* Annotation ist für den Besitzer und alle Autoren lesbar, und
- eine *öffentliche* Annotation kann von allen Benutzern (inkl. Autoren) gelesen werden.

Des weiteren können Annotationen verschiedenen Klassen (→ `Annotationstyp`) zugeordnet werden, z. B. Notiz, Mitteilung, Lesezeichen usw. Durch die Angabe von Schlüsselwörtern ist es möglich, Annotationen zusätzlich zu klassifizieren, um eine spätere Suche nach Annotationen zu erleichtern.

Die Stichwortsuche ist die zweite Möglichkeit für normale Benutzer, mit der Videodatenbank zu arbeiten. Hierbei stehen zwei Suchräume zur Verfügung:

- Eigene und (soweit Leserechte bestehen) fremde Annotationen (→ `Annotation, Annotationstyp`) sowie

- die inhaltsbezogenen Videobeschreibungen (→ `Video, Videosequenz, Kategorie`).

Die gefundenen Videofilme können entweder direkt abgespielt werden oder in die persönliche Playliste des Benutzers (→ `Benutzerplaylist`) eingefügt werden.

3.3.2 Autorensicht

Autoren haben im wesentlichen zwei Aufgaben zu bewältigen: zum einen das Hinzufügen neuer Videos zur Videodatenbank und die Pflege der dazugehörigen Metadaten (→ `Video, Videosequenz, Kategorie, Qualitätsbeschreibung, Datei`) und zum anderen die Erstellung und Pflege der Autorenplaylisten (→ `Autorenplaylist, Playlistitem`).

Bevor ein Videofilm in der Referenzdatenbank beschrieben wird, ist dieser vom Autor auf mindestens einem geeigneten Videoserver bereitzustellen. Da dies weder der Datenbankserver noch der Applikationsserver kontrollieren kann, obliegt es der Verantwortung der Autoren, daß die in der Datenbank gespeicherten Referenzen auf Videodateien (→ `Datei`) auch wirklich korrekt sind.

Mit den Metadaten werden physische, inhaltliche sowie strukturelle Eigenschaften der Videofilme beschrieben:

Physische Eigenschaften sind z. B. Bild- und Tonqualität (→ `Qualitätsbeschreibung`). Der Autor kann zu jedem Video mehrere Qualitätsbeschreibungen angelegen. Auf diese Weise ist es möglich, den unterschiedlichen Qualitätsanforderungen der Benutzer gerechtzuwerden. Beispielsweise könnten so für Benutzer, welche nur über eine Netzwerkanbindung mit niedriger Übertragungsrate an die Datenbank verfügen, die Videos zusätzlich in einer anderen Qualität bereitgestellt werden, die eine entsprechend geringere Bandbreite für die Netzübertragung erfordert. Natürlich muß für jede Qualität auch eine passende Datei (→ `Datei`) angelegt werden. Es dürfen aber auch mehrere sein, d. h. es ist möglich, Videodateien auf verschiedene Videoserver zu replizieren, um so beispielsweise die Verfügbarkeit der Videos zu erhöhen oder Videos in verschiedenen Dateiformaten anbieten zu können.

Inhaltliche Eigenschaften sind z. B. Filmtitel, Inhaltszusammenfassung, Namen von Produzent, Regisseur, Mitwirkenden, Entstehungsjahr usw. (→ `Video`). Zudem kann der Autor die Filme in bestimmte Kategorien (→ `Kategorie`), beispielsweise „Lehrfilm“ oder „Dokumentation“, einordnen.

Strukturell kann ein Video aus mehreren in sich abgeschlossenen Teilsequenzen bestehen (→ `Videosequenz`), welche als (logisch) eigenständige Videosequenzen in Playlisten aufgenommen und abgespielt werden können. Im einfachsten Fall, daß das Video nur aus einer Sequenz besteht, braucht der Autor gar nichts zu tun, da die erforderliche Videosequenzbeschreibung automatisch generierbar ist. Anderenfalls können beliebige (auch überlappende) Teilsequenzen manuell erfaßt und mit eigenen Inhaltszusammenfassungen versehen werden.

Die Erstellung von Autorenplaylisten (→ `Autorenplaylist`) wird durch eine Stichwortsuche (ähnlich wie bei der normalen Benutzerschnittstelle) unterstützt. Jeder Autor kann beliebig viele Autorenplaylisten erstellen, wobei jede Playliste einen eindeutigen Namen zugewiesen bekommt. Mit dessen Hilfe kann ein Playlisten-URL gebildet werden, so daß Hyperlinks auf Autorenplaylisten in beliebigen HTML-Seiten stehen können.

3.3.3 Administratorensicht

Administratoren haben sich im wesentlichen um die Verwaltung der Datenbank- bzw. Videoserver sowie der registrierten Videodatenbanknutzer (→ `RegistrierterNutzer`) zu kümmern.

Es können jederzeit neue Videoserver installiert und in die Videodatenbank eingebunden werden. Hierzu muß der Name und die Internetadresse des Servers (→ `Medienserver`) sowie das vom Server verwendete Protokoll (→ `Serverprotokoll`) in die Referenzdatenbank eingetragen werden. Mit Hilfe dieser Angaben bildet der Applikationsserver später den korrekten vollständigen URL für eine abzuspielende Videodatei.

4 Lessons learned

4.1 Architektur

Die Softwarearchitektur wurde vor allem auf Flexibilität und Skalierbarkeit hin ausgelegt. Das System ist in der Lage, durch Vervielfachung der Applikations- und Videoserver, gegebenenfalls auch durch Replikation des SQL-Servers, mit den Benutzerzahlen zu wachsen. Auch wurde auf die Unterstützung möglichst vieler verschiedener Videoserver- und Client-Software Wert gelegt.

Ein Schwachpunkt, der jedoch aufgrund fehlender Schnittstellen (auf SQL-Server- ebenso wie auf Videoserver-Seite) derzeit hingenommen werden muß, ist zweifellos die fehlende direkte Kopplung zwischen SQL-Server und Videoservern. Durch eine solche Kopplung könnte z. B. die Konsistenz (referentielle Integrität) der Daten wesentlich besser gewährleistet werden. Auch ein Monitoring der Videoserver durch den SQL-Server wäre wünschenswert, insbesondere, wenn viele Videodateien repliziert werden. Anstatt einem Benutzer alle Replikate eines Videos anzubieten, könnte man sich dann auf den Server mit der aktuell geringsten Last beschränken. Dazu mehr in Abschnitt 4.4.

4.2 Benutzerschnittstelle

Die Entscheidung, die Benutzerschnittstelle mittels dynamisch generiertem HTML (bzw. ASP) zu realisieren und nicht (wie zunächst geplant) mit Java, war in erster Linie von der Hoffnung auf einen möglichst geringen Entwicklungsaufwand getragen. Diese Hoffnung wurde nicht enttäuscht, obwohl die verwendeten Programmierwerkzeuge noch nicht ausgereift sind (z. B. hinsichtlich der Debug-Unterstützung). Parallel durchgeführte Arbeiten an einem Java-Client zeigten aber, daß hier vor allem die GUI-Programmierung für einen erheblichen Mehraufwand sorgt (etwa das Zwei- bis Dreifache).

Eine HTML-Lösung setzt jedoch einer benutzerfreundlichen Gestaltung der Schnittstelle wesentlich engere Grenzen als etwa Java. Dies liegt zum einen daran, daß eine verhältnismäßig starre dialogorientierte Struktur einzuhalten ist, und zum anderen an der geringen Zahl und Funktionalität der von HTML angebotenen Formularelemente.

Eine weitere Einschränkung ist, daß längere Transaktionen, die sich über mehr als eine Dialogseite erstrecken, sehr problematisch sind. Zwar ist es prinzipiell möglich, eine Transaktion in einem ASP-Skript zu beginnen und erst in einem zweiten zu beenden, jedoch gibt es keine Möglichkeit, den Benutzer dazu zu zwingen, das Skript zum Beenden einer Transaktion auch zur Ausführung zu bringen. Dadurch könnten leicht große Teile der Datenbank unabsichtlich oder mutwillig für längere Zeit blockiert werden, was natürlich bei Systemen mit vielen konkurrierenden Zugriffen nicht hinnehmbar ist. Bei der Videodatenbank-Benutzerschnittstelle wurde daher auf skriptübergreifende Transaktionen verzichtet, was in seltenen Fällen zu verlorengegangenen Updates führen kann.

Ein grundsätzlicher Vorteil der HTML-Lösung ist, daß sie sehr geringe Anforderungen an die Client-Software stellt. Es wird lediglich ein WWW-Browser benötigt, der Tabellen und Formulare darstellen kann. Dieser Vorteil wird allerdings im vorliegenden Fall durch die Notwendigkeit, einen Videoplayer in den Browser zu integrieren, weitgehend zunichte gemacht. Nun gibt es zwar inzwischen eine ganze Reihe von Videoplayern, die in HTML einbettbar sind (überwiegend Netscape-Plug-ins), und viele moderne Browser unterstützen das Einbetten auch. Jedoch funktioniert die Annotationskomponente der Videodatenbank nur dann korrekt, wenn der Videoplayer über einige bestimmte API-Funktionen verfügt.

Das Videoplayer-API muß nämlich eine zeitliche Synchronisation des Videos mit den Annotationen unterstützen. Dazu muß es u. a. eine Funktion zum Ermitteln der Abspielposition bereitstellen. Diese wird in Skripts (oder Java-Applets), die im Browser ausgeführt werden, genutzt, um z. B. das Anzeigen der Annotationen zu steuern.

Unter allen getesteten Videoplayern besitzen nur der *ActiveMovie Player* [MS98] und das *Netshow ActiveX Control* [MS97a] ein den Anforderungen genügendes API (der *RealVideo Player* erfüllt die Anforderungen zwar nicht ganz, es konnte jedoch eine Behelfslösung gefunden werden, die sowohl bei der Plug-in- als auch bei der ActiveX-Variante leidlich funktioniert). Eine Einschränkung der Clients auf diese drei Player wäre allerdings sehr restriktiv, da sie alle (bis auf das RealVideo-Plug-in) derzeit nur in Microsofts Internet Explorer funktionieren. Als Ausweg kam hier indes nur ein Kompromiß in Frage: die Benutzerschnittstelle unterstützt auch andere Browser und Videoplayer (auch nicht-eingebettete), jedoch auf Kosten eines nicht einheitlichen, von der Client-Software abhängigen Funktionsumfangs (worauf die Benutzer gegebenenfalls auch hingewiesen werden).

4.3 Konzeptionelles Schema

Das Design der Datenbank erfüllt den größten Teil der ursprünglichen Anforderungen. Das konzeptionelle Schema der Datenbank ist aber durchaus noch ausbaufähig. Insbesondere wurde bislang darauf verzichtet, mögliche Beziehungen *zwischen* Videosequenzen zu modellieren. Dies geschah zum einen aus Zeitmangel und zum anderen, weil zum Entwicklungszeitpunkt diesbezüglich noch zu wenig Konkretes über Anwenderverhalten und -wünsche bekannt war. Im folgenden werden noch einige weitere Punkte genannt, bei denen nach Vorliegen hinreichender Erfahrungen aus dem praktischen Einsatz der Videodatenbank über Schemaerweiterungen nachzudenken sein wird.

Die Benutzerverwaltung der Videodatenbank ist bewußt relativ einfach gehalten, da geplant ist, die Videodatenbank in das JaTeK-Projekt [FNR97] zu integrieren und die dort schon vorhandene Benutzerverwaltung zu nutzen. Für den Einsatz der Videodatenbank im Rahmen anderer Projekte (bzw. bei Änderung der ursprünglichen Planung) wäre die Benutzerverwaltung dann entsprechend auszubauen. Denkbar wäre beispielsweise die Vergabe verschiedener Rollen für Benutzer oder die Bereitstellung eines Zugriffsschutzes mit Hilfe von Paßwörtern.

Von der Videodatenbank wird nur der Datentyp Video unterstützt, doch für die Erstellung von Lernmaterialien für das World Wide Web ist Video nur einer von vielen multimedialen Bausteinen. Eine Erweiterung des Schemas um die Möglichkeit, verschiedene Medientypen verwalten zu können (beispielsweise durch das Hinzufügen von Klassen für Bilder und Audiodateien), ist noch leicht vorstellbar. Von einem Datenbanksystem zur Speicherung und Verwaltung komplexer multimedialer Dokumente (mit einer externen Repräsentation im HTML-Format) wäre man damit allerdings immer noch weit entfernt.

Der derzeitige Entwurf sieht zur Unterstützung der inhaltsorientierten Suche nur eine einfache Kategorisierung sowie spezielle Attribute (z. B. Schlüsselwortlisten) für einzelne Klassen vor. Um bessere Suchfunktionen zu ermöglichen, könnten noch spezielle Klassen für eine systematische Verschlagwortung der Daten hinzugefügt werden [Mac91]. Diese Methode gilt jedoch – besonders für multimediale Daten – als wenig effektiv. Hier wäre besser einem Einsatz der im folgenden Abschnitt kurz angerissenen *Information-Retrieval*-Verfahren der Vorzug zu geben.

4.4 Funktionalität

Der für die Realisierung der Videodatenbank verwendeten Standardsoftware fehlen noch einige wünschenswerte Eigenschaften, was die (mögliche) Funktionalität des Systems in mehreren wichtigen Punkten einschränkt. Hiervon sind vor allem die Aspekte *Lastverteilung*, *Information Retrieval* und *Dienstgüte (Quality of Service)* betroffen. Im folgenden soll dies noch etwas genauer erörtert werden.

4.4.1 Lastverteilung

Ein Videoserver kann nur eine verhältnismäßig geringe Zahl von Benutzern gleichzeitig bedienen (hier fehlen noch exakte Erfahrungswerte, man muß aber bei der im Projekt eingesetzten Hardware von einer maximalen Zahl von etwa 25 Benutzern pro Server ausgehen). Diese Beschränkung wurde zwar durch die Skalierbarkeit des Systems prinzipiell entschärft, viele Videoserver garantieren jedoch noch lange keine gleichmäßige Lastverteilung. Ohne eine regulierende Instanz kann es immer hoch belastete Server geben, während andere kaum beschäftigt sind. Automatische Lastverteilung zu realisieren, bedeutet nun im wesentlichen zweierlei:

- *Automatische Replikation der Videofilme*. Die Videodatenbank unterstützt zwar bereits Replikation, jedoch nur "manuell". Für eine automatische Replikation müßten zusätzlich Zugriffsstatistiken geführt werden, um häufig nachgefragte Videos sowie stark bzw. schwach belastete Server zu ermitteln ("Hotspots"). Während diese Anforderung noch relativ leicht zu erfüllen ist (z. B. mittels spezieller Log-Tabellen und Trigger), ist die automatische Durchführung der Replikation wegen des erforderlichen Kopierens der Videodaten von einem Server zum anderen wesentlich aufwendiger zu realisieren.
- *Ständige Beobachtung (Monitoring) der Videoserver*. Die Vorteile der Replikation kommen nur zum Tragen, wenn man bei der Beantwortung einer Anfrage genau das (Videodatei-)Replikat liefert, welches auf dem Server mit der (aktuell) geringsten Last liegt. Dies erfordert ein Monitoring der Videoserver. Einige der von uns verwendeten Videoserver-Typen (z. B. RealVideo) verfügen über eine Monitoring-Schnittstelle. Diese sind jedoch generell so unterschiedlich, daß ihre Nutzung viel Programmieraufwand erfordert, wenn man auf der Unterstützung vieler Server-Typen beharrt. Für den RealVideo-Server müßte z. B. ein Monitor-Plug-in entwickelt werden [Rea98], während man sich bei Servern ohne spezielle Monitoring-Schnittstelle z. B. auf eine Komponente zur (permanenten) Analyse der Logdatei verlegen könnte.

4.4.2 Information Retrieval

Die Videodatenbank unterstützt bislang nur die Volltextsuche auf Beschreibungsdaten und Annotationen, die von den Benutzern zu den Videos angelegt werden. Die Volltextsuche ist hinreichend effektiv und effizient, solange in der Videodatenbank nur eine überschaubare Anzahl von Videos verwaltet wird (Größenordnung: <1000). Aber bei sehr großen Datenmengen liefert die Volltextsuche u. U. eine riesige Treffermenge zurück, mit der ein Benutzer dann nichts mehr anzufangen weiß. Zudem werden von der Volltextsuche vage Anfragen oder Anfragen mit räumlichen oder zeitlichen Bezug zu den Videodaten nur unzureichend unterstützt.

Wesentlich bessere Ergebnisse erzielt man durch den Einsatz von Indexierungs- und Suchverfahren, die neben den vorhandenen Metadaten auch die Videodaten selbst nutzen. Mit Hilfe geeigneter Indexierungsverfahren werden hierbei die relevanten Informationen aus den Videodaten gewonnen, die später bei der Auswertung von Anfragen d. h. für eine inhaltsorientierte Suche notwendig sind [Sch97]. Mittlerweile gibt es eine Vielzahl solcher Indexierungsverfahren, die meist ihren Ursprung in der Mustererkennung haben, z. B. [KNB95].

Durch Verwendung inhaltsorientierter Suchverfahren bieten sich neue Möglichkeiten, die der Benutzer bei seinen Anfragen einsetzen kann. Als Beispiel sei die Suche nach Aspekten wie räumliche Gegebenheiten oder zeitliche Abläufe angeführt [CL96], z. B. "ermittle alle Videos, in denen ein Auto von links nach rechts durch das Bild fährt". Um nun die neuen Suchmöglichkeiten auch auf Benutzerseite anbieten zu können, ist von der Datenbank eine entsprechend erweiterte bzw. angepaßte Anfragesprache bereitzustellen [ÖS+97].

4.4.3 Dienstgüte

Dienstgüte (Quality of Service, QoS) ist ein seit Jahren intensiv bearbeitetes Forschungsthema, überwiegend im Bereich der Rechnernetze und verteilten Multimediasysteme [ACH98, HS97], aber auch z. B. im Bereich der Multimedia-DBMS [TK96, MR97]. In die Entwicklung heutiger Videoserver wie z. B. die von uns verwendeten haben diese Arbeiten jedoch noch wenig Eingang gefunden. Eine Aushandlung von Dienstgüteparametern (z. B. der Bildgröße) ist i. d. R. nicht möglich, da all diese Parameter schon bei der Erstellung der Videodateien festgelegt werden. Will man dem Benutzer dennoch zumindest die Auswahl zwischen verschiedenen vorgegebenen Videoqualitäten bieten, muß man verschiedene Versionen der Videodateien mit unterschiedlichen Dienstgüteparametern (z. B. unterschiedlicher Bildgröße) bereitstellen. Die Videodatenbank unterstützt diese Lösung, die jedoch mit mehreren Nachteilen behaftet ist:

- Die Erstellung der verschiedenen Versionen muß manuell, d. h. ohne Unterstützung durch die Videodatenbank erfolgen.
- Jede Version belegt permanent Speicherplatz auf dem Videoserver. Wird eine Version kaum oder gar nicht genutzt, ist der Platz verschwendet.
- Die Festlegung der Dienstgüteparameter ist weiterhin sehr statisch und kann nicht an spezielle Benutzerwünsche angepaßt werden.

5 Diskussion: alternative Realisierungsmöglichkeiten

5.1 Softwarearchitektur

Die Anforderung, daß Benutzer über ihren Web-Browser auf die Videodatenbank zugreifen sollen, läßt sich alternativ zu der von uns gewählten Architektur auch durch eine einfachere zweistufige Client/Server-Architektur umsetzen. Das heißt in diesem Fall: Verlagerung des größten Teils der Anwendungslogik in den Client und Verzicht auf den Applikationsserver. Generelle Vor- und Nachteile dieses Ansatzes sind z. B.:

+ Weniger Komponenten, daher geringerer Administrationsaufwand, + evtl. benutzerfreundlichere Clients (mehr oder weniger erzwungen),	– "fette" Clients, – weniger gut skalierbar, – Clients greifen direkt auf den DB-Server zu (d. h., sie benötigen Zugangsberechtigungen, und der Server kann schnell durch viele offene Verbindungen überlastet werden), – höherer Entwicklungsaufwand, da HTML und Skriptsprachen bei den meisten Ansätzen kaum nutzbar.

Es gibt mehrere mögliche Wege, um diese Architektur umzusetzen, z. B. *Java-Applets*, *ActiveX-Controls* oder integrierte WWW- und DB-Server [NTT95]. Übersichten zu verschiedenen Techniken der WWW-Anbindung von Datenbanken findet man ferner in [BG98] und [Loe97]. Vor- und Nachteile einer Java-basierten Lösung sind u. a.:

+ Unterstützung auf vielen Plattformen, + keine Softwareinstallation auf Client-Seite, + gutes Sicherheitskonzept,	– Inkompatibilität zwischen verschiedenen Java-Releases, – Applets werden immer über Netz geladen, daher lange Ladezeiten bei fettem Client, – Entwicklungswerkzeuge, Bibliotheken (z. B. JDBC) usw. noch zu unausgereift.

Bei einer ActiveX-basierten Lösung verkehren sich die genannten Punkte weitestgehend ins Gegenteil, d. h., keine von beiden trägt einen klaren Sieg davon.

Wollte man die Vorteile von Java (oder ActiveX) gegenüber HTML für die Client-Entwicklung nutzen, so ist eine zweistufige Client/Server-Architektur sicherlich einfacher und schneller zu realisieren, zumal dies durch moderne Java-Entwicklungsumgebungen (z. B. PowerJ von Powersoft) bereits recht gut unterstützt wird. Die oben genannten Nachteile (fette Clients, direkte Client-DB-Server-Kopplung, …) sprechen jedoch deutlich gegen diese Architektur. Ein weiterer Nachteil ist, daß sich wegen des Java-Sicherheitskonzepts der DB-Server und der WWW-Server auf dem gleichen Rechner befinden müssen. Ein vielversprechender, wenngleich wesentlich aufwendiger umzusetzender (und daher in diesem Projekt verworfener) Ansatz wäre daher eine *drei*stufige Architektur mit Java-basiertem Client *und* Applikationsserver.

5.2 DBMS: relational, objektorientiert oder objekt-relational?

Es kann hier nicht im einzelnen auf die Unterschiede bzw. Vor- und Nachteile von relationalen (RDBMS), objektorientierten (ODBMS) und objekt-relationalen (ORDBMS) Datenbankverwaltungssystemen eingegangen werden. Hierfür sei auf die zahlreich vorhandene Literatur verwiesen, z. B. [AB+89], [SST97] und [Sto96].

ODBMS und ORDBMS bieten mächtigere Datenmodelle als RDBMS. Angesichts der eher geringen Komplexität des Videodatenbank-Schemas wären hier aus dem Einsatz solcher Modelle aber wohl nur wenige Vorteile zu ziehen. Doch wie sieht es mit zwei anderen Schwachpunkten aus, die wir in Bezug auf den SQL-Server genannt hatten (vgl. Kap. 4.1 und 4.4)? Das ist zum einen die sehr schwache Kopplung zwischen dem Datenbank- und den Videoservern und zum anderen die geringe Information-Retrieval-Funktionalität.

Zum ersten Punkt ist festzustellen, daß auch ODBMS und ORDBMS keine Videoserver-Funktionen (*Continuous Media Streaming*) anbieten. Dafür werden also weiterhin spezielle Server wie z. B. der von uns eingesetzte RealVideo-Server benötigt. Das Problem der schwachen Kopplung besteht demnach zunächst auch hier. In vielen aktuellen Produkten, z. B. [Inf97a], [NCR97] und [CA97], wird eine Kopplung zwischen DBMS und Videoserver prinzipiell nicht anders realisiert als in unserer Lösung: mit externen Referenzen, deren Integrität vom System nicht kontrolliert wird. IBM's *DataLinks*-Konzept [NMB96] hingegen bietet die gewünschte referentielle Integrität (sowie Backup und Recovery) für externe Dateien. Eine weitere interessante Lösung mit sehr enger ORDBMS-Videoserver-Kopplung ist in [HS+98] beschrieben.

Bezüglich des zweiten Punkts geben derzeit ORDBMS Anlaß zur Hoffnung. Spezielle IR-Erweiterungen wie sie z. B. für den *Informix Universal Server (IUS)* entwickelt wurden [Inf97a], demonstrieren, daß komplexe IR-Funktionen für Text-, Audio- und Videodaten in ein solches Datenbanksystem integrierbar sind. Auch Entwicklungen wie das *Virtual-Index Interface* [Inf97b] des IUS, das die Integration neuer sekundärer Zugriffspfadstrukturen erlaubt, bieten hier interessante Perspektiven, die freilich noch eingehender zu erforschen sind.

Abschließend kann man also mutmaßen, daß die Videodatenbank eher von einem Wechsel zu einem ORDBMS profitieren würde, vor allem, wenn die IR-Funktionalität ausgebaut werden soll. Zudem sollte hierbei die Schemamigration leichter fallen. Es ist jedoch zu bedenken, daß man sich mit einem solchen Schritt wesentlich enger als bisher an einen bestimmten DBMS-Hersteller bindet, da der Standardisierungsgrad bei ORDBMS geringer ist als bei RDBMS.

6 Zusammenfassung und Ausblick

Es wurde eine Videodatenbank beschrieben, die nach den Anforderungen der Anwender für eine experimentelle Teleteaching-Umgebung innerhalb von zwei bis drei Monaten entworfen, implementiert und in Betrieb genommen werden konnte. Es zeigt sich hierbei, daß es möglich ist, den grundlegenden

Anforderungen unter Verwendung heute verfügbarer Standardsoftware gerecht zu werden. Viele weitergehende Anforderungen wie z. B. adaptives Lastverteilungs- und Dienstgütemanagement oder effektive Information-Retrieval-Funktionen scheitern jedoch noch am beschränkten Funktionsumfang dieser Software. Teilweise fehlt es sogar an geeigneten Schnittstellen, um solche Funktionen in eigener (aufwendiger) Entwicklungsarbeit hinzuzufügen. Neueste Entwicklungen wie beispielsweise ORDBMS befinden sich zumindest in letztgenannter Hinsicht auf einem vielversprechenden Weg.

Die beschriebene Videodatenbank ist seit einigen Monaten für die Lehre und das Studium im Fach Psychologie einsetzbar. Schon in diesem kurzen Zeitraum hat sich der Nutzen dieser Entwicklung gezeigt. Für die Lehre wichtige Videosequenzen sind sehr viel leichter und schneller in der Datenbank zu finden und aufzurufen als in der bestehenden Sammlung von Videobändern mit ihren Inhaltsangaben. Die Präsentation der Sequenzen kann präziser und mit weniger Aufwand vorbereitet sowie in den Veranstaltungen besser von den Lehrenden auf die Bedürfnisse und Nachfragen der Studierenden abgestimmt werden als das mit herkömmlichem Videomaterial der Fall ist. Ein weiterer wesentlicher Vorteil dieses Videoinformationssystems besteht darin, daß die Studierenden das vorhandene Videomaterial ohne nennenswerten Aufwand auch außerhalb der Lehrveranstaltungen zum Selbststudium nutzen können.

Die im praktischen Einsatz gesammelten Erfahrungen werden weiter aufschlußreiche Informationen liefern – sowohl über die Leistungsfähigkeit (und Schwächen) heutiger Software als auch über notwendige Erweiterungen oder Änderungen an der Konzeption der Videodatenbank. Zudem erhoffen wir uns dadurch wichtige Impulse für die Fortsetzung unserer Forschungsarbeiten an einem Multimedia-DBMS [MR97].

Danksagungen

Die Autoren möchten Herrn Prof. Klaus Meyer-Wegener für die Unterstützung danken, ohne die dieses Videodatenbank-Projekt nicht zustande gekommen wäre. Weiterer Dank gilt den anonymen Gutachtern für ihre hilfreichen Hinweise zur Verbesserung einer früheren Fassung dieses Beitrags.

Literatur

[AB+89] Atkinson, M. P., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D., Zdonik, S.: The Object-Oriented Database System Manifesto. In: Kim, W., et al. (ed.): Deductive and Object-Oriented Databases, Proc. 1st Int. Conf. DOOD'89, Kyoto, Japan, Dec. 1989.

[ACH98] Aurrecoechea, C., Campbell, A. T., Hauw, L.: A Survey of QoS Architectures. In: Multimedia Systems (1998) 6, Springer-Verlag 1998, S. 138-151.

[BG98] Benn, W., Gringer, I.: Zugriff auf Datenbanken über das World Wide Web. In: Informatik-Spektrum Band 21 Heft 1, Springer-Verlag, 1998.

[CA97] Jasmine™ Technical Overview. Computer Associates International Inc., 1997.

[CL96] Chang, W. C., Lee S. Y.: Spatio-temporal Contents Acquisition and Retrieval in a Video Information System. In: Proc. 3rd Pacific Workshop on Distributed Multimedia Systems (DMS'96), HKUST, Hong Kong, 1996.

[Flu96] Fluckiger, F.: Multimedia im Netz. München, London: Prentice Hall, 1996.

[FNR97] Franze, K., Neumann, O., Rennecke, S.: DFN -Projekt "Teleteaching Dresden-Freiberg". 1. Zw.-Ber. "Vorstellung der Konzeption von JaTeK, Javal, JaWoS", TU Dresden, 1997.

[HS+98] Hollfelder, S., Schmidt, F., Hemmje, M., Aberer, K., Steinmetz, A.: Transparent Integration of Continuous Media Support into a Multimedia DBMS. In: Proc. Int. Workshop on Issues and Applications of Database Technology (Berlin, Germany, July 6–9), 1998.

[HS97] Hutschenreuther, T., Schill, A.: Quality of Service Abstraktionen für verteilte Multimedia-Systeme. In: it+ti - Informationstechnik und Technische Informatik Heft 4, R. Oldenbourg Verlag, 1997.

[Inf97a] Informix Digital Media Solutions: The Emerging Industry Standard for Information Management. Informix White Paper, Informix Software, Inc., 1997.

[Inf97b] Informix Universal Server. Virtual-Index Interface Programmer's Manual. Informix Press, 1997.

[KNB95] Kumar, P. S., Narasimhalu, A. D., Babu G. P.: Create-Time Indexing for Digital Video. Technical Report, Institute of Systems Science, National University of Singapore, 1995.

[Loe97] Loeser, H.: Datenbankanbindung an das WWW. In: Datenbanksysteme in Büro, Technik und Wissenschaft, Proc. BTW'97 (Ulm, März 1997), Springer-Verlag, 1997, S. 83–99.

[Mac91] Macleod, I. A.: Text Retrieval and the Relational Model. In: Journal of the American Society for Information Science, 42(3), 1991, S. 155–165.

[MR97] Marder, U., Robbert, G.: The KANGAROO Project. In: Proc. 3rd Int. Workshop on Multimedia Information Systems (Como, Italy, Sept. 25–27), 1997, S. 154–158.

[MS97a] Microsoft Netshow 2.0 Technical Overview. http://www.microsoft.com/netshow/about/whtepprs/techover.htm, Microsoft Corporation, 1997.

[MS97b] Web Server Scripting. White Paper, http://www.microsoft.com/iis/guide/scripting.asp, Microsoft Corporation, 1997.

[MS97c] Database Access for Web Applications. White Paper, http://www.microsoft.com/iis/guide/access.asp, Microsoft Corporation, 1997.

[MS98] Microsoft ActiveMovie. http://www.microsoft.com/directx/pavilion/amovie/, Microsoft Corporation, 1998.

[NCR97] Introduction to Teradata® Multimedia Services. Doc. No. B035-1002-097A, NCR Corp., 1997.

[NMB96] Narang, I., Mohan, C., Brannon, K.: Coordinated Backup and Recovery between DBMS and File Systems. IBM Research Report, IBM Almaden Research Center, Oct. 1996.

[NTT95] Interserv: A WWW Server and DBMS in One. http://www.nttdata.jp/products_services/network/interserver_e.html, NTT Data Corp., 1995.

[ÖS+97] Özsu, M. T., Szafron, D., Oria, V, Li, J. Z.: MOQL: A Multimedia Object Query Language. In: Proc. 3rd Int. Workshop on Multimedia Information Systems (Como, Italy, Sept. 25–27), 1997, S. 19–28.

[Rea97] RealVideo Technical White Paper. http://www.real.com/devzone/library/overview.html, RealNetworks, Inc., 1997.

[Rea98] RealMedia SDK Developer's Guide. http://www.real.com/devzone/sdks/rmsdk/guide/index.html, RealNetworks, Inc., 1998.

[RB90] Rumbaugh, J., Blaha, M.: Object-oriented Modeling and Design. Prentice Hall, 1990.

[Sch97] Schäuble, P.: Multimedia Information Retrieval. Kluwer Academic Publishers, 1997.

[Ses98] Sessions, R.: COM and DCOM: Microsoft's Vision for Distributed Objects. John Wiley & Sons, 1998.

[SST97] Saake, G., Schmitt, I., Türker, C.: Objektdatenbanken: Konzepte, Sprachen, Architekturen. Int. Thomson Publishing, 1997.

[Sto96] Stonebraker, M.: Object-Relational DBMSs - The Next Great Wave. Morgan Kaufman, 1996.

[TK96] Thimm, H., Klas, W.: Playout Management in Multimedia Database Systems. In: Nwosu, K. C., Berra, P. B., Thuraisingham, B., (eds.): Design and Implementation of Multmedia Database Management Systems. Kluwer Academic Publishers, 1996.