Unterstützung des strukturierten Zugriffs auf MPEG Videos in einem Multimedia-Datenbankmanagementsystem

Matthias Malsy, Silvia Hollfelder, Arnd Steinmetz, Karl Aberer
GMD - Forschungszentrum Informationstechnik
IPSI – Institut für Integrierte Publikations- und Informationssysteme
Dolivostraße 15
D - 64293 Darmstadt, Germany
E-mail: {malsy, hollfeld, arnd.steinmetz, aberer} @darmstadt.gmd.de

Zusammenfassung

Videoanwendungen, die auf einzelne Segmente eines kodierten Mediums zugreifen, benötigen Informationen über die Struktur der Daten. Um diese Strukturinformation zu erhalten, müssen kodierte Videodaten geparst werden. Diese Informationen werden üblicherweise nicht persistent gehalten, was bei einem wiederholten Zugriff ineffizient ist. Dieser Artikel beschreibt die Integration eines MPEG-Parsers in ein erweiterbares Datenbankmanagementsystem. Dadurch werden Funktionen für den Zugriff auf inhaltsbezogene Daten wie auch grundlegende Funktionen zum Extrahieren von Datenströmen oder Dekodieren einzelner Bilder als Datenbankfunktionalität zur Verfügung gestellt. Dieses sogenannte MPEG-DataBlade bildet die Grundlage für zukünftige Erweiterungen eines MM-DBMS.

1. Motivation

Multimediale Videoanwendungen lassen sich grob in streaming-orientierte und interaktive Anwendungen klassifizieren. Bei streaming-orientierten Anwendungen, wie z.B. Video-on-Demand, stehen Lösungen für effiziente Speicherungs-mechanismen und Echtzeitretrieval zur kontinuierlichen Präsentation in verteilten Umgebungen im Vordergrund, die durch kommerzielle Systeme, wie beispielsweise dem Oracle Video Server [ORA98], unterstützt werden. Im Gegensatz dazu erfordern interaktive Anwendungen, wie beispielsweise Video Editing und Produktion [S97], Video Browsing [THE98] und Computer Based-Training (CBT), effiziente Such-, Zugriffs- und Präsentationsmöglichkeiten für einzelne Videosegmente.

Für diese Videoanwendungen haben sich eine Reihe von Formaten durchgesetzt. Bei der Speicherung von Filmen ist MPEG-1 [ISO93], [LG91] und MPEG-2 [ISO96], [R96], deren Spezifizierung durch ein internationales Komitee (Motion Picture Experts Group) durchgeführt wurde, weit verbreitet. Die primäre Zielsetzung des MPEG-Standards war eine effiziente Kodierung und Dekodierung mit einer hohen Datenkompression ohne signifikante Verluste bei der Bild- und Tonqualität.

Videoformate wie MPEG erweisen sich bei interaktiven Anwendungen, die sich mit der Nachbearbeitung, dem Indizieren und dem Retrieval von Videodaten beschäftigen, als problematisch, da die hierfür notwendigen Zugriffsfunktionen auf kodierte Daten ausschließlich für die Präsentation optimiert sind. Anforderungen, wie der wahlfreie Zugriff auf einzelne Videosegmente, können nur durch Parsen, d.h. eine strukturelle Zerlegung der kodierten Videos, ermöglicht werden. Diese Vorgehensweise ist ineffizient, wenn die beim Parsen erzeugten Strukturinformationen nach Beendigung des Videozugriffes nicht persistent gespeichert werden und dadurch für einen wiederholten Zugriff nicht zur Verfügung stehen.

Durch den Einsatz von Multimedialen Datenbankmanagementsystemen (MM-DBMS) können Strukturinformationen als Metadaten verwaltet werden. Somit muß die Extraktion von Strukturinformationen von Parser und Decoder nur einmalig ausgeführt werden. Zusätzlich kommen die Vorteile von DBMS wie beispielsweise Persistenz, Multi-User-Fähigkeit, Transaktionsmanagement, Sicherheit, Skalierbarkeit und Query-Unterstützung zum Einsatz.

Da die Basisdatentypen der konventionellen MM-DBMS eingeschränkt sind, wird die Speicherung von multimedialen Daten, wie Audio, Video und Animation, meist nur als Binary Large Objects (BLOBs) unterstützt. Der Zugriff auf einzelne Strukturelemente der komprimierten Rohdaten ist meist nicht möglich. Inhaltssuche und Editing erweisen sich als problematisch [D98]. Dieses Problem wurde durch die Einführung von erweiterbaren, objektrelationalen DBMS (OR-DBMS) und objektorientierten DBMS (OO-DBMS) teilweise gelöst, die die Integration beliebiger Datentypen und somit auch Videodatentypen ermöglichen. Beispiele für erweiterbare objektrelationale DBMS sind der Informix Dynamic Server mit der „DataBlade“ Technologie, DB2 von IBM mit „Relational Extenders“ und das DBMS Oracle 8, dessen Erweiterungen „Cartridges“ genannt werden [ORA97]. Das objektorientierte DBMS Jasmine [HFPH98] von Computer Associates (CA) unterstützt multimediale Anwendungen durch eine eigene Klassenbibliothek. Diese Klassenbibliothek bietet allgemeine Verwaltung-funktionen für BLOBS an, die MM-Daten repräsentieren. Zudem werden medienspezifische Methoden wie beispielsweise Komprimierung angeboten. Der Datentyp MPEG wird jedoch nicht unterstützt [JAS].

Dieser Artikel beschreibt die Forschungs- und Entwicklungsarbeit zur Integration eines MPEG-Datentyps in ein kommerzielles objektrelationales Datenbankmanagementsystems am GMD-IPSI. Ausschlaggebend für die Wahl eines OR-DBMS als Entwicklungsplattform ist deren starke Etablierung aufgrund der normierten Abfragesprache SQL. Zudem bieten OR-DBMS meist mehr Zuverlässigkeit und Skalierbarkeit. Die notwendigen Zugriffsfunktionen für MPEG-Ströme werden in Form von Bibliotheken zur Verfügung gestellt. Der Ansatz bietet gleichzeitig Erweiterungsmöglichkeiten für komplexere Anwendungsfunktionen von Videodaten wie Watermarking, Schnitterkennung, Videobearbeitung und Videoretrieval.

2. Der MPEG-Standard

MPEG ist ein durch die ISO standardisiertes Datenformat, das in den Versionen MPEG-1 und MPEG-2 definiert wird. Eine der Hauptanwendungen von MPEG-2 ist die digitale Fernsehtechnik [R97], [R97]. Beide Formate werden durch eine Datenstruktur beschrieben, die einen gültigen MPEG-Datenstrom (sog. „stream“) definiert. Diese Datenstruktur besteht aus hierarchisch angeordneten Strukturen, sogenannten „layers“, die ineinander verschachtelt sind.

Ein MPEG-Datenstrom kann aus genau einem Video, genau einem Audio oder einem sogenannten „system stream“ bestehen, der bis zu 32 Audio- und 16 Videoströme ent-halten kann. Die einzelnen Daten-ströme sind dabei in gemultiplexter Form innerhalb von Paketen ge-speichert. Der „System Header“ beinhaltet Meta-informationen des kodierten Stroms. Abbildung 1 zeigt beispielhaft die Struktur eines MPEG-1 System Streams (nach ISO 11172), der zwei Video- und einen Audiostrom enthält.

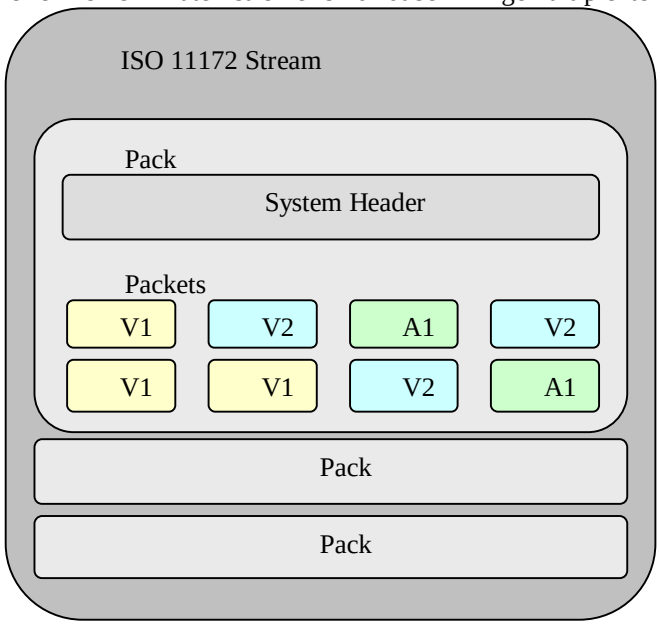

Abbildung 1: Beispiel eines MPEG-1 Stroms

Nimmt man den Inhalt der einzelnen Datenpakete eines Video- bzw. eines Audiostroms und fügt sie wieder zusammen, so ergibt sich ein gültiger Video bzw. Audiostrom, der unabhängig abspielbar ist.

MPEG kodiert Bilder in Blöcke, die mit der DCT-Transformation [IEEE90], [L84] gewandelt und zusätzlich quantisiert werden. Das Verfahren erreicht seine hohe Datenkompressionssrate im Wesentlichen durch Referenzen zu zeitlich vorangegangen und nachfolgenden Bildern, die mit Hilfe von Bewegungsvektoren angegeben werden. Dies spiegelt sich unter anderem in den verschiedenen Bildtypen des MPEG-Datenformates wieder. Die wichtigsten sind die Intra-kodierten Bilder (I-Frames), die unabhängig dekodiert werden können, da sie keine Referenzen zu anderen Bildern beinhalten, die „Predictet“ kodierten Bilder, deren Daten teilweise aus Rückwärtsreferenzen zu I-Frames bestehen und den „Bidirectional Predicted“ Bildern (B-Frames), die sowohl Vorwärts- als auch Rückwärtsreferenzen zu P-Frames bzw. I-Frames haben können. Hierin liegt die besondere Komplexität bei der Verwaltung und Bearbeitung von MPEG-Videos verborgen, da einzelne P- und B-Frames wegen bestehender Referenzen nur in Verbindung mit anderen Frames dekodierbar sind.

3. Integration eines MPEG-Datentypen in ein DBMS

3. 1 Anforderungen an die Systemarchitektur

Im Anschluß wird die Systemarchitektur beschrieben, die folgenden Anforderungen genügen soll:

- Es soll eine einfache, erweiterbare Schnittstelle zur Verfügung stehen, die sowohl für Video- als auch für Metadaten einsetzbar ist, um die Konsistenz zwischen den Daten zu gewährleisten. Diese unabhängige Schnittstelle soll für unterschiedliche Plattformen und für unterschiedliche Programmiersprachen einsetzbar sein.
- Die physikalische Speicherung von Mediendaten und Metadaten soll getrennt erfolgen, um eine bessere Performanz durch spezialisierte Medienserver zu erreichen. Sowohl für die Rohdaten als auch für die Metadaten müssen Anfragen durch effiziente Datenstrukturen und Query-Mechanismen möglich sein.

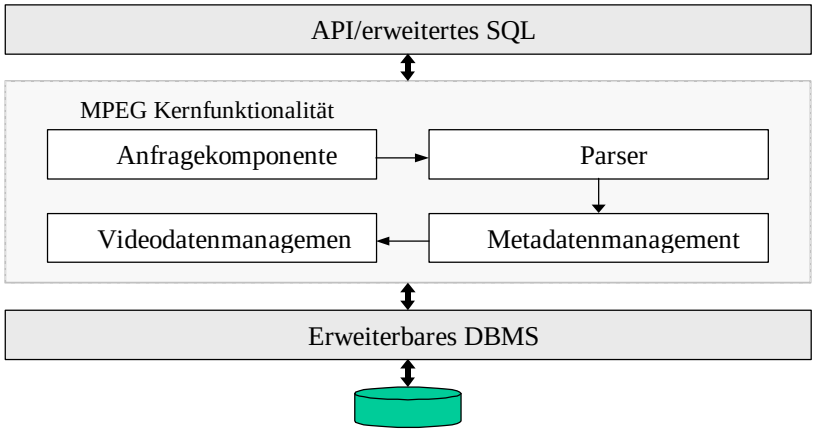

Abbildung 2: Systemarchitektur

Die Systemarchitektur (Abbildung 2) besteht aus den folgenden Komponenten: der Kern ist ein OR-DBMS, das um Zugriffsfunktionen auf Videodaten und deren Metadatenverwaltung erweitert wird. Die Anfragekomponente führt Benutzer-anfragen aus, indem sie auf die Ergebnisdaten eines Parserdurchlaufes zugreift.

3.2 Die erweiterbare DBMS Plattform

Als Plattform wurde der Informix Dynamic Server (IDS) von Informix ausgewählt, der durch die „DataBlade-Technologie“ eine Reihe von Erweiterungs-mechanismen zur Verfügung stellt, wie beispielsweise die Anbindung von sogenannten „shared libraries“ an den Datenbankkern. Die in den DataBlades definierten Funktionen werden der Datenbank zur Laufzeit bekannt gegeben und ermöglichen dadurch, daß sie den gleichen Adressraum wie der Datenbankkern benutzen, um effizient innerhalb des DBMS zu arbeiten. DataBlades werden durch eine Reihe von Datentypen und Funktionen unterstützt, die in der Informix-eigenen Skriptsprache „Stored Procedure Language“ (SPL) definiert werden.

Der Informix Dynamic Server bietet Erweiterungsmöglichkeiten durch folgende Definitionen bzw. Komponenten:

- Programmierung von Prozeduren und Funktionen in SPL (Stored Procedure Language).
- Definition zusammengesetzter Datentypen via SQL.
- Anbindung von Prozeduren und Funktionen über eine C-Schnittstelle als „Shared Library“.
- Definition und Implementierung eigener Datentypen und Operatoren, deren Repräsentation und Speicherung in einer C-Struktur durchgeführt wird.

- Definition von Aggregatfunktionen analog zu den SQL Aggregatfunktionen MIN oder SUM.
- Implementierung eigener Indexstrukturen (Sekundärzugriff).
- Transparente Integration externer Quellen über die Implementierung eigener Primärzugriffe (virtuelle Tabellen).

3.3 Die Kernfunktionalität zur Unterstützung von MPEG

Zur Unterstützung von MPEG-Videos werden folgende vier Software-komponenten benötigt.

Videodaten Management - Rohdatenverwaltung

Die Basisfunktionalität zur Verwaltung von Mediendaten wird durch das Video Foundation DataBlade (VF-DB) von Informix [VFDB97] abgedeckt, das Einfüge-, Lösch- und Zugriffsfunktionalitäten unterstützt. Es basiert auf einer Softwarearchitektur, die eine Schnittstelle zur Verwaltung von Mediendaten auf externen Speichermedien (z.B. Filesystem, Videoserver) anbietet.

Die Verwaltung von Videodaten erfolgt nach dem Paradigma von Archivsystemen. Die Modifikation eines Videos löst die Generierung eines neuen modifizierten Videos aus. Videodaten und Metadaten bleiben somit immer konsistent, da ein modifizierter Videostrom eine neue Identifikation (ID) erhält. Weiterhin wird dem OR-DBMS so die Optimierung und das Cachen von Zugriffsfunktionen erlaubt, indem alle Funktionen als „not variant“ deklariert werden. D.h. das mehrfache Aufrufe einer Funktion mit gleichen Parametern immer das gleiche Resultat ergeben.

Management von Metadaten

Der Hauptnutzen der Integration eines MPEG-Datentypen in ein OR-DBMS besteht in der Speicherung von Strukturinformationen als Metadaten und der Bereitstellung von Zugriffsfunktionen auf diese Informationen. Das VF-DB stellt ein Datenschema für generische Metadaten (z.B. Erstellungsdatum, Medientyp, Autor, Länge) zur Verfügung. Sowohl das Management von Metadaten als auch von Videodaten ist unabhängig vom Videoformat. „Stratas“ erlauben die Modellierung von Zeitdimension in Videos und die Assoziation von Metadaten mit Videosegementen [SD92].

Metadaten können automatisch generiert sein, beispielsweise durch automatische Schnitterkennung oder können in einem Anwendungskontext manuell hinzugefügt worden sein, z.B. durch Textannotationen einzelner Bilder oder eines Kurzvideos.

Kern dieser Arbeit ist die Entwicklung eines MPEG-DataBlades, das auf dem Video Foundation DataBlade aufbaut. Es liefert Funktionen zum strukturierten Zugriff auf komprimierte MPEG-Videoströme, wie beispielsweise das Extrahieren von Datenströmen oder Dekodieren einzelner Bilder.

Das MPEG-DataBlade unterstützt folgende Funktionen:

- Identifikation der vorhandenen Video- und Audioströme sowie deren Bildwiederholrate, Bildgröße, und weitere formatbezogene Metadaten.
- Zugriff auf Einzelbilder wie die Identifikation des Bildtyps (I-, P-, B- Frames) sowie das Dekodieren einzelner Bilder.
- Zugriff auf Teile von Einzelbildern, wie Blöcke und die zugehörigen Bewegungsvektoren.
- Extrahieren (Demultiplexen) einzelner Datenströme.

Parser

Der MPEG-Parser extrahiert Metadaten. Er arbeitet direkt auf den Binärdaten der Videos, die auf dem externen Speichermedium abgelegt sind, und liefert der Datenbank diese Daten zur Speicherung.

Anfragekomponente

Die SQL Funktionen, die das MPEG-DataBlade zur Verfügung stellt, sind primär Funktionen, die interne Strukturen von MPEG sichtbar machen. Dies erfolgt möglichst vollständig für alle Layer von MPEG-1 und MPEG-2 bis zum Picture Layer. Weiterhin werden Funktionen zur Verfügung gestellt, die Daten des MPEG-Stroms interpretieren und weiterverarbeiten. Dies sind beispielsweise die Berechnung der Bildnummer (eine Information, die nicht direkt im MPEG kodiert ist), das Dekodieren von einzelnen Bildern oder das Extrahieren von Teilströmen. Als Basis für alle Funktionen wird ein neuer Datentyp MPEG eingeführt.

Die nachfolgende Liste gibt einen kleinen Überblick über die zur Verfügung gestellten Funktionen auf einen MPEG-System Stream:

- Einlesen eines Stroms (Funktion des VF-DB) mit automatischem Parsen:
  ```
  insert into myMPEGtable
  values('Vacancy97', FileToVSI('myVideo.mpg')::MPEG);
  ```
- Auslesen der Datenstruktur „System Header", die u.a. Anzahl und Typen der verfügbaren Teilströme enthält:
  ```
  select systemHeader(mpeg) from myMPEGtable;
  ```
- Auslesen der Anzahl der im Datenstrom verfügbaren Teilströme:
  ```
  select numStreams(mpeg) from myMPEGtable;
  ```
- Einfügen vorhandener Teilstrom-Typen in die Tabelle „videoTypes":
  ```
  insert into videoTypes execute function
  listVideoTypes (select mpeg from …);
  ```
- Demultiplexen einzelner Streams:
  ```
  insert into myMPEGtable execute function
  demux(1, (select mpeg from …));
  ```

Eine Reihe von weiteren Funktionen können auf die unterschiedlichen Video- und Audioströme sowie deren einzelnen Layer angewendet werden.

3.4 Anwendungsentwicklungs-Schnittstelle (API)

Wie stark sich eine Multimediaerweiterung in ein DBMS integrieren läßt ist jeweils systemspezifisch und kann somit nicht generell beantwortet werden. Der Standard SQL 92 als kleinster gemeinsamer Nenner aller relationalen DBMS erlaubt keine Erweiterung. Erst im Standard SQL3 [ISO96b] sind Typerweiterungen möglich.

Basierend auf den im Kapitel 3.2 vorgestellten Erweiterungsmöglichkeiten werden für den IDS zwei Konzepte zur Integration von Video diskutiert, die sogenannte „vollständige Integration" und die „benutzerdefinierte Integration".

Vollständige Integration aller MPEG-Strukturen

Dieser Ansatz bildet alle Struktur- und Metadaten der MPEG-Ströme in virtuellen Tabellen mit dem jeweiligen Layer-Namen ab. Auf die MPEG-1 Struktur angewandt ergibt dies z.B. die Tabellen „Pack" und „Packet", die jeweils alle Daten ihres Layers beinhalten und über Fremdschlüssel miteinander verbunden sind. Realisiert werden virtuelle Tabellen durch eine Gruppe von Funktionen, die eine Abbildung von SQL

Zugriffen auf MPEG-Strömen über die Definition eines Primärzugriffes realisieren. Der Zugriff auf häufig benötigte Daten kann durch Anlegen von Indizes optimiert werden.

Der Vorteil dieses Integrationskonzeptes ist die vollständige transparente Einbindung der Daten aller Ströme. Durch die Verknüpfung über Fremdschlüssel werden alle Kombination von Abfragen ermöglicht. Ein Nachteil des Konzeptes ist die immense Datenmenge. Zugehörige Indizes können nur auf alle Videos gleichzeitig angewendet werden, obwohl im Anwendungskontext zu erwarten ist, daß nur spezielle Videos und Strukturen für eine jeweilige Person bzw. Applikation relevant sind. Die aktuell verfügbaren Indexstrukturen sind somit nicht geeignet.

Benutzerdefinierte Integration der MPEG-Strukturen

Das Konzept der benutzerdefinierten Integration legt eine Reihe von Funktionen fest, die Daten und Metadaten aus einem MPEG-Strom generieren und sie über eine Einfügeoperation in eine benutzerdefinierte Tabelle speichern. Die MPEG-Daten und Strukturen stehen somit dem Benutzer nach Einfügen eines Videos nicht automatisch zur Verfügung, sondern werden von ihm explizit erzeugt. Der Ansatz sieht vor, daß vom Benutzer bzw. applikationsspezifisch festgelegt wird, welche Daten in welcher Granularität persistent gespeichert werden. Die Vorteile dieses Konzeptes sind, daß die Datenmenge gering gehalten wird und über übliche Indexmechanismen optimiert werden kann. Nachteilig ist die Beschränkung der Anfragemöglichkeiten auf die festgelegte Datenstruktur.

Der Ansatz der benutzerdefinierten Integration, dessen Funktionalität für unsere Zwecke hinreichend ist, wurde ausgewählt und realisiert. Das Fehlen geeigneter Indexstrukturen erwies sich bei der vollständigen Integration als Ausschluß-kriterium. Es sind Varianten dieses Konzeptes für spezielle Anwendungen denkbar, die jedoch hier nicht weiter verfolgt werden.

4. Realisierung

Ausgangspunkt der Realisierung ist ein objektorientierter MPEG-Parser, der über die Anfragekomponente angesteuert wird. Dieser erzeugt während des Parserdurchlaufs, der direkt auf den Rohdaten des Videos im Videoserver arbeitet, eine Datenstruktur, die die wichtigsten Informationen für den direkten Zugriff auf die MPEG-Daten enthält. Diese Struktur, die für jedes Video spezifisch erstellt wird, besteht im wesentlichen aus den Startpositionen der einzelnen Header und aus zusätzlichen, nicht im MPEG-Strom enthaltenen Information (z.B. Bildnummer). Aufträge der Anfragekomponente werden in wahlfreie Zugriffe umgesetzt und das Ergebnis zurückgegeben. Es gibt zur persistenten Speicherung zwei Funktionen, die die Umwandlung der Speicherstruktur, dem sogenannten Parserzustand („Parser State“), in einen Datenstrom bzw. dessen Rekonstruktion ermöglichen.

Bei der Realisierung als MPEG-DataBlade wurde der Parser auf die speziellen Programmieranforderungen eines DataBlades angepaßt. Diese beinhalten unter anderem Vorgaben zum Speichermanagement sowie eine eingeschränkte Benutzung von Systemaufrufen.

Der Zugriff auf MPEG-Daten erfolgt in zwei Schritten, dem Parsen und der Abfrage von Daten bzw. dem Erzeugen von Metadaten.

4.1 Parsen

Das Parsen (Abbildung 3) eines MPEG-Stromes wird entweder durch einen expliziten Aufruf der Parser-Funktion oder durch sogenanntes „Casten“ des Datentypes „VSIFile“ zum MPEG-Datentyp, der eine Videoreferenz im Video Foundation DataBlade beschreibt, ausgelöst (1). Im zweiten Schritt (2) greift diese Funktion über Lese- und Positionierfunktionen, die durch das VF-DB zur Verfügung gestellt werden, auf den Binärdaten der Videos im Videoserver zu. Dann wird der Parserzustand erstellt, der in der Datenbank in einer Tabelle persistent gespeichert wird (3).

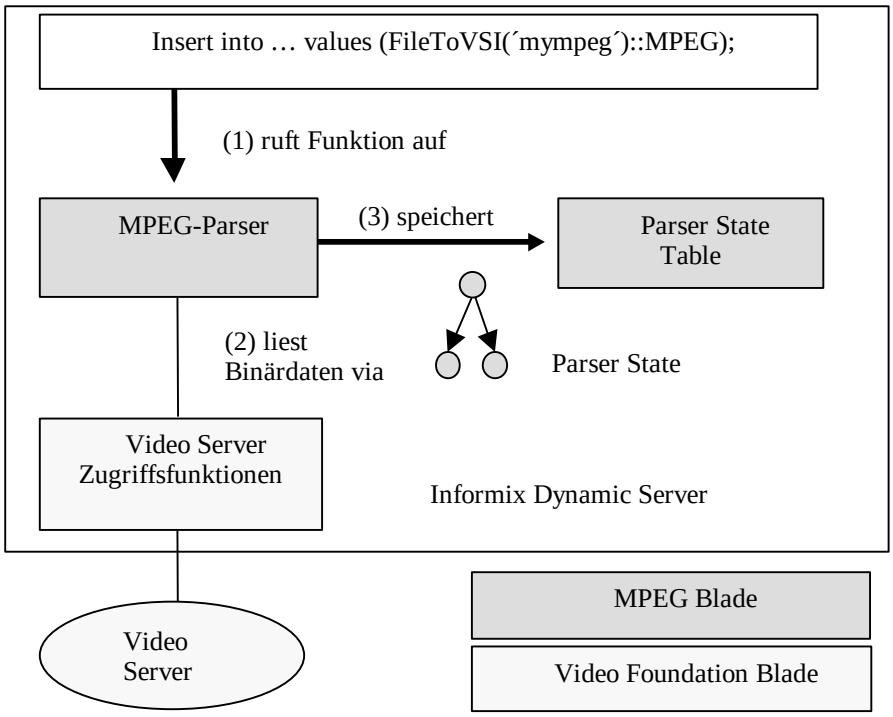

Abbildung 3: Parsen, ausgelöst durch das Einfügen eines Videos

4.2 Anfrage

Die Generierung und Speicherung von Metadaten wird durch entsprechende SQL-Funktionen des MPEG-DataBlades ausgelöst (Abbildung 4). Im ersten Schritt wird der Parserzustand wiederhergestellt. Hierzu wird die persistente Repräsentation wieder in das interne Format überführt. Danach werden die MPEG-Zugriffsfunktionen ausgeführt, die ihre Informationen entweder direkt aus dem Parserzustand erhalten oder sich die Information aus dem Datenstrom holen. Um zu vermeiden, daß die recht teure Operation der Rekonstruktion des Parserzustandes bei jedem Aufruf erneut durchgeführt wird, wird eine konfigurierbare Anzahl von Parserzuständen im transienten Speicher

des IDS vorrätig gehalten. Der Austausch der Parserzustände erfolgt über den Least Recently Used (LRU) Mechnismus.

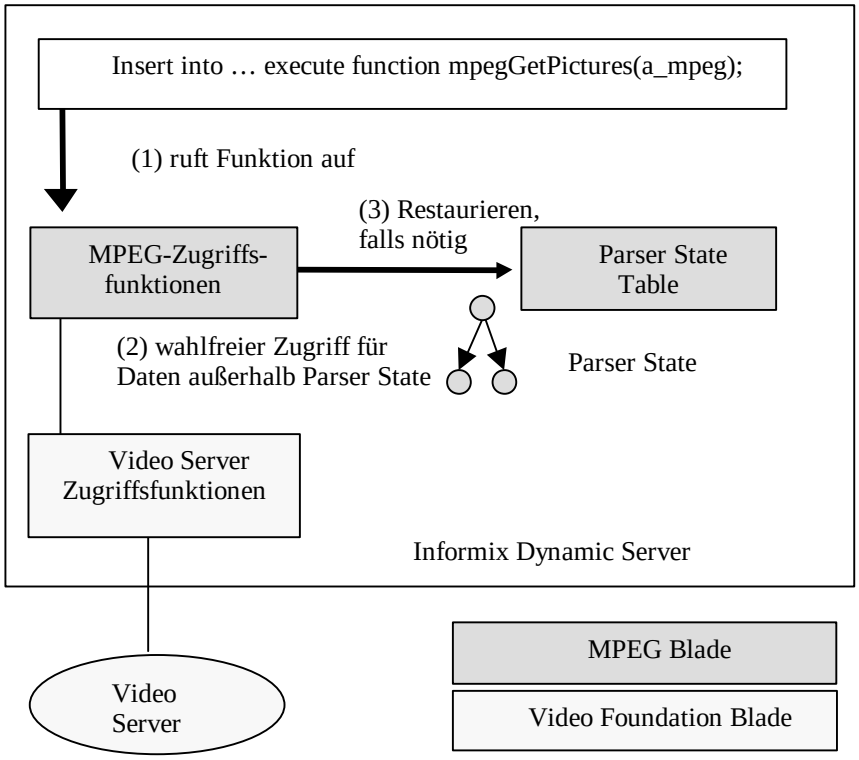

Abbildung 4: Extraktion von Daten und Metadaten

5. Schlußbemerkungen

Die Vorteile dieses Systems ergeben sich aus der persistenten Speicherung der Resultate des Parservorganges. Dadurch können häufig benötigte Struktur- und Metadaten der Videos mit der SQL Abfragesprache abgerufen werden. Die Funktionen des MPEG-DataBlades werden einfach und selektiv auf große Mengen von Videos ausgeführt. Weiterhin besteht die Möglichkeit, Funktionen des MPEG-DataBlades mit Funktionen anderer DataBlades, wie beispielsweise zur Inhaltsanalyse (Image DataBlade), innerhalb der Anfragesprache zu verknüpfen.

Derzeit stehen die Implementierungsarbeiten zur Entwicklung des MPEG-DataBlades kurz vor dem Abschluß. Die Funktionalität des MPEG-DataBlades wird zukünftig am GMD-IPSI im Rahmen verschiedener Projekte als Basisfunktionalität zur Videoanalyse, Videobearbeitung und zur Anpassung der Präsentationsqualität [HSHA98], [HKR97] eingesetzt.

Literaturverzeichnis

[D98] J. Diercks: Am Anfang war das BLOB, Magazin für professionelle Informationstechnik (iX), August 1998.

[HFPH98] A. Heuer, G. Flach, K. Post, O. Hein: Jasmine: OO-Datenbank für multimediale Anwendungen, Magazin für professionelle Informationstechnik (iX), August 1998.

[HKR97] S. Hollfelder, Achim Kraiss, Thomas C. Rakow: A Client-Controlled Adaptation Framework for Multimedia Database Systems. In Proc. of European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'97), Darmstadt, Germany, September 1997.

[HSHA98] S. Hollfelder, F. Schmidt, M. Hemmje, K. Aberer: Transparent Integration of Continuous Media Support into a Multimedia DBMS, In Proceedings of IADT '98, Berlin 1998.

[IEEE90] Specification for the implementation of 8x8 inverse cosine transform, IEEE, 1990.

[ISO93] Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5 Mbits/s, ISO/IEC 11172, 1993.

[ISO96] Information Technology - Generic Coding of Moving Pictures and Associated Audio Information, ISO/IEC 13818, 1996.

[ISO96b] ISO/IEC JTC1/SC21 N10489, ISO/IEC 9075, Committee Draft (CD), Database Language SQL, July 1996.

[LG91] D. LeGall: MPEG: A Video Compression Standard for Multimedia Applications. In Communications of the ACM, 34(4), 1991.

[JAS] http://www.cai.com/offices/germany/jas_konz.htm

[L84] A. Leger: Implementation of fast discrete cosine transform for full color video services and terminals. In Proc. of IEEE Global Telecommunications Conference, 1984.

[ORA97] Oracle8 Multimedia Management Solutions. An Oracle White Paper, June 1997.

[ORA98] Oracle Video Server System - Technical Overview. An Oracle White Paper, March 1998.

[R97] U. Reimers (Ed.), Digitale Fernsehtechnik: Datenkompression und Übertragung für DVB. Berlin/Heidelberg/New York, Springer Verlag, 1997.

[R98] U. Reimers: Ein Führer durch die Welt der DVB-Spezifikationen und -Normen. In Fernseh- und Kino-Technik 52 (1+2): S. 82-86, 1998.

[RS93] L. A. Rowe, B. C. Smith: A New Family of Algorithms for Manipulating Compressed Images. In IEEE Computer Graphics and Applications 13(5): S. 34-42, 1993.

[R96] D. Ruiu: An Overview of MPEG-2. Digital Video Test Symposium, Bad Homburg, Hewlett Packard Inc., 1996.

[SD92] T. G. A. Smith, G. Davenport: The Stratification System: A Design Environment for Random Access Video. In Proc. of Third Int. Workshop on Network and Operating Systems Support for Digital Audio and Video, La Jolla, California USA, November 1992.

[S97] A. Steinmetz: DiVidEd - A Distributed Video Production System. In Proc. of VISUAL'96 Information Systems, VISUAL'96, 1996.

[THE98] U. Thiel, S. Hollfelder, A. Everts: Multimedia Management and Query processing Issues in Distributed Digital Libraries: A HERMES Perspective. Wird publiziert in Proc. of the Sec. Int. Workshop on Query Processing in Multimedia Information Systems QPMIDS'98, in Verbindung mit DEXA'98, August 1998.

[VFDB97] Video Foundation DataBlade Module User's Guide, Version 1.1, INFORMIX Press, Menlo Park, Juni 1997.