Published in: Luth N. (Hrsg.): Inhaltsbezogene Suche von Bildern und Videosequenzen in digitalen multimedialen Archiven, Beiträge eines Workshops der 22. Jahrestagung für Künstliche Intelligenz (KI '98) in Bremen, Bremen, Sept. 1998, pp. 7-25

Architekturen von Multimedia-Datenbanksystemen zur Speicherung von Bildern und Videos

Günther Specht

Technische Universität München
Institut für Informatik
Orleansstr. 34
D-81667 München
email: specht@in.tum.de

Abstract. Dieses Papier stellt, ausgehend von der Architektur konventioneller Datenbanksysteme und den demgegenüber neuen Anforderungen an Multimedia-Datenbanksystemen, vier verschiedene Basisarchitekturen für Multimedia-Datenbanksysteme vor. Im letzten Abschnitt wird als ein Beispiel das System MultiMAP vorgestellt, ein multimediales Datenbanksystem, das an der TU München entwickelt wurde.

1 Einführung und Überblick

Multimedia- und Hypermedia-Systeme haben in den letzten Jahren einen enormen Aufschwung genommen. Sie werden vor allem als Intranet und Internet-Informationssysteme eingesetzt. Immer mehr treten dabei Bilder, Audios, Videos, Graphiken und Animationen einschließlich beliebiger Links zwischen ihnen in den Vordergrund gegenüber rein sequentiellen Texten. Mit zunehmender Größe des Datenbestands und des Benutzerkreises treten jedoch Probleme auf, die eine effektive und effiziente Nutzung, sowohl bei der Informationseingabe und Bereitstellung als auch beim Retrieval erschweren. Inkonsistente Links, fehlendes Transaktionskonzept bei mehreren Autoren, eingeschränkte Suchmöglichkeiten, zu grober Zugriffsschutz und keine Sekundärindizierung sind nur einige der Schwachpunkte heutiger, immer noch größtenteils dateibasierter Systeme.

Der Einsatz von Datenbanksystemen bietet dagegen eine Reihe an Vorteilen wie effizienter, indexunterstützter Zugriff, Verwaltung auch großer Datenmengen, automatische Konsistenzprüfungen, transaktionsgeschützter Mehrbenutzerbetrieb und Recovery.

Jedoch sind klassische Datenbanksysteme für Multimedia-Anwendungen nicht oder nur bedingt ausgelegt. Relationale Datenbanken wurden für große Mengen an gleich strukturierten, alphanumerischen Datensätzen entwickelt, die normalisiert in Tabellenform ablegbar sind, wie beispielsweise Personaldaten oder Kontodaten, wobei nur ein eingeschränkter Datentypumfang zur Verfügung steht. Objektorientierte Datenbanken

kapseln Objekte und ihre Methoden, bieten aber keine Synchronisationsunterstützung. So hat sich aus diesen Anforderungen in den letzten Jahren ein neues Gebiet innerhalb der Datenbanksysteme entwickelt, die Multimedia- oder Hypermedia-Datenbanksysteme.

Die klassischen Datenbanksystem-Fähigkeiten sind Persistenz, Transaktionskonzept (ACID), Mehrbenutzerfähigkeit, Recovery, Ad-hoc Queries, Integritätsbedingungen zur Konsistenzsicherung, Sicherheit und Performanz in der Queryauswertung. Für Multimedia-Datenbanksysteme müssen diese Fähigkeiten jedoch erweitert werden um das Speichern und Wiedergewinnen von Multimedia-Daten, insbesondere von kontinuierlichen Daten. Dazu gehören

a) beim Speichern:

- die Eingabe von Multimedia-Objekten, Links, Synchronisationspunkten etc.
- die Fähigkeit, sehr viele Multimedia-Objekte speichern zu können
- geräte- und formatunabhängiges Archivieren
- die Bereitstellung eines entsprechenden Speichermanagementsystems: performant, hohe Kapazität, kosten-optimiert

b) beim Retrieval:

- die Unterstützung komplexer Suche, insbesondere
  * navigierende Suche,
  * attributbasierter Suche und
  * inhaltsbasierter Suche
- Effizienz (spezielle Indizes)
- die Auswertung auch komplexer Queries (Aggregation, Filtern)
- Konvertierungen zur Unterstützung der Geräte- und Formatunabhängigkeit
- die Unterstützung der Echtzeit-Anforderungen bei der Ausgabe kontinuierlicher Medien

c) beim Update:

- zumindest ein Ersetzen, seltener auch ein Editieren multimedialer Daten.

Bei der Speicherung und Verwaltung multimedialer Objekte hat man es intern jeweils mit drei Typen an Informationen zu jedem Objekt zu tun. Es handelt sich dabei um

- **Rohdaten,** sie werden nicht weiter interpretiert (z.B. BLOBs);
- **Registrierungsdaten** (Steuerdaten), sie beziehen sich auf das Medien-Objekt (z.B. Datenformat, Höhe, Breite, Dauer, Farbtiefe, Kompressionsverfahren, Aufzeichnungsbitrate etc.) und
- **Beschreibungsdaten,** also symbolische Daten in Attributen oder Volltexten, die den Inhalt beschreiben;

Die Rohdaten und ihre Zusatzinformationen können zusammen oder verteilt abgelegt sein. Eine gemeinsame Speicherung wird bei den (beiden) Backend-Architekturen und der integrierten Architektur vorgenommen. Die verteilte Ablage führt zur Medien-

Server-Architektur für Multimedia-Datenbanksysteme. Wir werden alle vier Architekturen diskutieren.

Dieses Papier gliedert sich wie folgt. In Abschnitt 2 wiederholen wir kurz die Schichtenarchitektur konventioneller Datenbanksysteme. Abschnitt 3 stellt die neuen Anforderungen Multimedialer Datenbanksysteme gegenüber konventionellen Datenbanksystemen zusammen. Deren Realisierung führt zu vier möglichen Basisarchitekturen. Sie werden in Abschnitt 4 diskutiert. Schließlich stellen wir in Abschnitt 5 unser eigenes multimediales Datenbanksystem MultiMAP und einige seiner Anwendungen vor.

2 Internarchitektur konventioneller Datenbanksysteme

Zur Beschreibung des internen Aufbaus von Datenbanksystemen - unabhängig vom Datenmodell (relational, objektorientiert, etc.) - kann eine abstraktes Schichtenmodell definiert werden. Das Datenbanksystem wird dabei in zueinander hierarchisch angeordneten Schichten aufgeteilt, die bezüglich der Anfrageverarbeitung jeweils unterschiedliche Abstraktionsniveaus darstellen. Ohne die externe Applikationsschicht erhält man dabei typischerweise vier Schichten. Abbildung 1 zeigt rechts die Schichtennamen mit den wesentlichen Teilen ihrer Funktionalität und links die abstakte Datenstruktur auf der die jeweilige Schicht aufsetzt. Ziel ist es, SQL-Queries, die ja Anfragen auf der Ebene von Relationen darstellen, über die Tupel und Segment-Ebene auf Anforderungen von Seiten und Plattenblöcken abzubilden. Die folgende Beschreibung stützt sich in der Sprechweise auf das relationale Modell. Für objektorientierte DB-Systeme gilt entsprechendes.

Insgesamt unterscheidet man 5 Ebenen. Zuoberst sitzen die **DB-Applikationen**. Sie greifen über *embedded SQL* oder *ad hoc Queries* auf das Datenbanksystem zu.

Die Anfrage kommt zuerst an die **DB-Verwaltung**, die die Anfrage auf der Ebene der beteiligten Relationen sieht. Diese Schicht hat Zugriff auf das *Data Dictionary* (DB-Schema, Katalog) und damit auch auf die Rechteverwaltung. Sie enthält weiterhin die Programme zur Übersetzung und Verarbeitung der *Data Definition Language* (DDL) zur Schemaerstellung und Änderung im Katalog, sowie der *Data Manipulation Language* (DML) und der *Data Retrieval Language* (DRL) zur Übersetzung und Optimierung der Anfragen in die relationale Algebra. Auf dieser Zugangsschicht sind auch Teile der Concurrency-Verwaltung angesiedelt. Aufgabe der Datenbankverwaltung ist es also, Objekte und Operationen des jeweiligen Datenmodells zur Verfügung zu stellen, den Mehrbenutzerbetrieb zu realisieren und Anfragen in der Querysprache (SQL, OQL, etc.) auf die nächstniedrigere interne Auswertungsebene, dem Operatorgraph (oder Query-Execution-Plan) optimiert abzubilden und auszuführen. Der Operatorgraph arbeitet auf der Ebene der relationalen Algebra. Hier werden die Joins, Selektionen, Projektionen, Mengenvereinigungen und -differenzen etc. ausgeführt.

Der Operatorgraph greift in seinen Blättern auf Relationen bzw. einzelnen Tupeln daraus und damit auf Indexstrukturen zu. Diese werden von der nächsten Schicht, der

Zugriffsstrukturverwaltung angefordert. Auf dieser Ebene werden die Zugriffs- und Indexstrukturen (z.B. B*-Baum, Hash, etc.) bereitgestellt, einschließlich entsprechender Hilfsprogramme, wie die Sortierung oder die Scan-Operationen. Über den B-Baum-Zugriff kommt man auf einzelne Blätter. Diese entsprechen in ihrer optimierten Größe gerade der Seitengröße des Betriebssystems, so daß ein Knoten immer einer Seite entspricht. Die Sperrenverwaltung kann auf der Ebene der B-Bäume und deren Knoten aufsetzten, oder auf der Seitenebene. Dann wird sie erst vom Segmentverwalter aufgerufen.

Die Anfrage wird also in der nächst niedrigeren Schicht auf der Ebene von Seitenanforderungen bzw. allgemein Segmentanforderungen betrachtet. Dies realisiert die **Segment-Verwaltung** (oder Gebietsverwaltung). Sie bildet die logische Einteilung in Speichersegmente auf Seiten und Blöcke ab, verwaltet den DB-Cache und enthält die zugehörige Pufferstrategie und Freispeicherverwaltung. Ziel des DB-Cache-Einsatzes ist die Optimierung des physischen Datentransports zwischen Platte und Hauptspei-

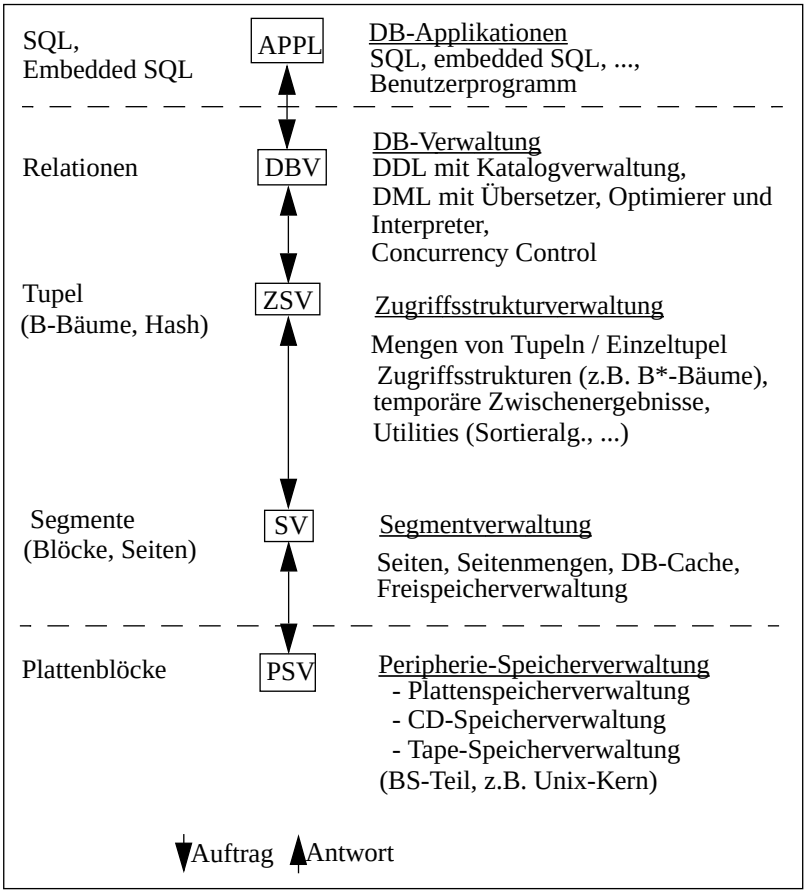

Abb. 1: Hierarchische Schichtung der DBS-Software

cher. Auf dieser Ebene kann also entschieden werden, ob die relevanten Teile bereits im Cache liegen oder erst von der Platte geholt werden müssen. Entsprechend sind auf dieser Ebene auch die Commit-Verarbeitung von Transaktionen und die Recovery-Mechanismen angesiedelt. Sie müssen insbesondere auf die Eigenschaften der DB-Cache-Schnittstelle abgestimmt sein (z.B. synchrones, verzögertes oder gänzlich asynchrones Schreiben, entsprechend dem gewählten Redo/Undo-Verfahren der Recovery). Die Segmentverwaltung ist auch zuständig für das Schreiben der Log-Datei.

Für den Plattenzugriff wird schließlich die **Peripherie-Speicherverwaltung** (des Betriebssystems) aufgerufen. Diese arbeitet auf der physischen Speicherebene des jeweiligen Speichermediums, also im Fall einer Plattenspeicherverwaltung auf Plattenblöcken und Plattenzylindern (Spuren) und optimiert die Plattenarmbewegungen (manchmal zum Leidwesen des DBS, das am Transaktionsende am liebsten ein sofortiges, synchrones Durchschreiben der DB-Cache-Daten auf die Platte hätte); im Fall eines DAT-Bandes oder Magnetbandes auf physischen Magnetbandblöcken und den entsprechenden Spulvorgängen.

3 Anforderungen an Multimedia-Datenbanksysteme

Aufgrund der Tatsache, daß wir es bei Multimedia-Datenbanksysteme mit kontinuierlichen Daten, sehr großen Einzeldaten und einem stark interaktiven Benutzerverhalten zu tun haben, ergeben sich einige Unterschiede in den Anforderungen an Multimedia-Datenbanksysteme gegenüber den klassischen Datenbankfunktionalitäten:

1. Unterstützung multimedialer, erweiterbarer Datentypen:
   Die wichtigste Anforderung besteht in der Unterstützung erweiterbarer abstrakter Datentypen (ADTs) zur Speicherung und Verarbeitung mediengebundener, insbesondere kontinuierlicher Daten, nicht nur BLOBs. Zunächst wäre bereits eine Anreicherung der festen Datentypen um spezielle Medien-Datentypen wie `MPEG`, `AVI`, `QUICKTIME`, `MIDI`, `JPEG`, `GIF`, `TIFF`, `TEXT`, etc. hilfreich. Man erkennt jedoch bald, daß diese Liste laufend erweitert werden müßte, so daß die Forderung nach benutzerdefinierbaren Datentypen auftritt. Diese Datentypen beinhalten immer auch verschiedene Zugriffsfunktionen und Auslieferungsoperationen (kontinuierlich oder als Ganzes) sowie verschiedene Indexstrukturen (sofern diese überhaupt definierbar sind). Damit könnte man zwar alle benötigten Datenformate einbringen, eine Formatabstraktion wird aber noch nicht unterstützt. Daher wird als Optimum eine Erweiterung um abstrakte Datentypen (ADTs) gefordert, wobei abstrakte Datentypen (wie `IMAGE`, `VIDEO`, `AUDIO`, etc.) von den konkreten Datenformaten abstrahieren. Sie stellen nicht nur den Speicherplatz, sondern auch die Zugriffsfunktionen (einschließlich evtl. notwendiger Formatumwandlungen) und die algebraischen Gesetze darauf (wichtig für den Optimierer) zur Verfügung. Über diesen ADTs sollte eine Klassenhierarchie definierbar sein, einschließlich der Vererbung von Methoden.
   [7] schlägt beispielhaft einen ADT IMAGE für Rasterbilder vor. IMAGE ist nur über anwendbare Funktionen beschreibbar und zugreifbar. Entsprechend bleibt nach außen die interne Speicherstruktur verborgen. Einzelne Komponenten kön-

nen auch nur virtuell vorhanden sein, sie werden also gar nicht explizit gespeichert, sondern aus anderen Komponenten errechnet. Folglich legt die für die Erzeugung eines Datentyps IMAGE angebotene Funktion `create_image` in keiner Weise fest, welches interne Format tatsächlich verwendet wird:

```
create_image
      (height             : integer,
       width              : integer,
       depth              : integer,
       aspect-ratio       : real
       encoding           : code
       colormap_length    : integer,
       colormap_depth     : integer,
       colormap           : list(2,*) of list(1,*) of integer
       pixelmatrix        : list(1,*) of bit
      ):IMAGE
```

In objektorientierten Datenbanksystemen kann IMAGE als Klasse definiert werden, in objekt-relationalen als Typ oder als Data-Blade und in erweiterbaren relationalen z.B. als Domain. IMAGE kann dann wie ein normales Attribut verwendet werden.

```
CREATE TABLE employee (emp-no: integer, ..., portrait: IMAGE).
```

2. Geräte- und formatunabhängiges Retrieval:
   Während man sich bei der Speicherung einer Integer-Zahl beim Retrieval nicht um das intern verwendete Format (z.B. little endian oder big endian) kümmern muß, sind Bilder und Videos heute i.a. nur im Eingabeformat wieder herausholbar. Automatische Konverter zum jeweils günstigsten Anzeigeformat gehören heute nicht zum Standardsatz eines Multimedia-Datenbanksystems, sondern noch in die Applikationsschicht. Gegebenenfalls muß aus Performanzgründen auch redundant in verschiedenen Formaten gespeichert werden. Dies sollte durch die ADTs nach außen gekapselt sein. Das Multimedia-Objekt sollte jeweils in der gewünschten Form ausgeliefert werden. Es kann dabei aus Performance-Gründen intern in dem am häufigsten gewünschten Format abgelegt sein. Ändert sich der häufigste Formatwunsch, könnte das Multimediasystem auch intern zur Optimierung eine Umcodierung vornehmen, ohne daß die externen Aufruffunktionen geändert werden müßten.

3. Unterstützung verschiedener Kompressionstechniken:
   Mit der formatunabhängigen Speicherung hängt eng auch die Frage des Einsatzes von Kompressionstechniken zusammen. Die große Menge an Daten sowohl zur Speicherung als auch zur Übertragung kann den Einsatz von Kompressionstechniken erfordern. Zusätzliche Angaben, ob ein Objekt verlustbehaftet gespeichert oder ausgeliefert werden darf, kann dem Datenbanksystem Hinweise zum erlaubten Einsatz von Kompressionstechniken geben.

4. Zugriff auf Teildaten:
   Während man einen BLOB nur als Ganzes an die Applikationsschicht ausgeben kann, wird aber in der Query oft nur ein Teilstück daraus angefordert, z.B. eine

Szene aus einem Film. Der direkte Ansprung kann nur bei der DB-Unterstützung im Zugriff auf Teildaten effizient unterstützt werden.

5. Indexstrukturen für Multimedia-Objekte:
   Klassische Datenbanksysteme unterstützen an Indexstrukturen vor allem Varianten an B-Bäumen und Hash-Indexe. Diese Indexstrukturen sind auf sehr viele, aber relativ kurze Tupel mit alphanumerischen Daten ausgelegt. Genügen diese auch für Bilder und Videos und die Objektidentifikationen innerhalb dieser? Je nach Anwendungsgebiet werden auch mehrdimensionale, räumliche oder zeitliche Indexe erforderlich (z.B. UB-Trees, R-Trees, k-d-Trees, Oct-Trees, Grid-Files, etc.)

6. Unterstützung von Multimedia-Clustering:
   Dieser Punkt hängt eng mit obigem zusammen. Standardmäßig wird relationenweise bzw. klassenweise und innerhalb dieser nach dem Primärindex geclustert. Medien-Objekte wie Videos sind wesentlich größer als eine Seite. Ihre Speicherung benötigt selbst auch wieder Clusterinformation. Während der Videoauslieferung ist der Plattenarm beschäftigt. Gleichzeitig benötigte Anker oder durch Synchronisationsbögen parallel dazu anzuzeigende Texte, Bilder oder Audios, dürfen kein Plattenarmflattern verursachen. Eine räumlich nahe oder verschachtelte Speicherung ist dann wünschenswert. Entsprechende Clustering-Informationen, auch über Relationen hinweg, sollten im Schema angegeben werden können. Diese könnten z.B. analog einer View definiert werden.

7. Skalierbarkeit und Partitionierbarkeit der Daten:
   Aufgrund der Speichergröße der Multimedia-Daten, kann nicht davon ausgegangen werden, daß eine Multimedia-Datenbank auf eine Platte paßt. Es kommen Plattenfarmen oder hierarchische Sekundär- und Tertiärspeicher zum Einsatz. Darüber müssen die Daten geeignet partitioniert, gegebenfalls auch gesplittet werden. Die optimale Verteilung - unter den beiden Randbedingungen: effizienter Zugriff und minimaler Verschnitt - ist aufgrund der unterschiedlichen Größe der Daten wesentlich schwieriger als bei herkömmlichen Tabellen.

8. Erweiterung des Transaktionskonzepts:
   Während das klassische ACID-Transaktionskonzept ein “Alles oder Nichts”-Prinzip realisiert, wird dieses bei der Aufnahme und der synchronen Wiedergaben von Video- oder Audioströmen unpraktikabel. Man will ja gerade nicht, daß nach einem Fehler in der zehnten Minute das ganze Video nochmals neu aufgezeichnet werden muß und kann auch andererseits bei einem Wiedergabefehler in der zehnten Minute das Gezeigte nicht mehr ungesehen machen. Das führt zur den Forderungen nach *geschachtelten Transaktionen* (*nested transactions* [8]), genauer *offen geschachtelten Transaktionen* und zusätzlichen Wiederaufsetzpunkten in *langen Transaktionen*. Offen geschachtelte Transaktionen erlauben im Gegensatz zu geschlossen geschachtelten Transaktionen die Freigabe von Subtransaktionen nach außen, ohne daß die Vater-Transaktion bereits commited ist. Sind alle Subtransaktionen auf derselben Ebene, lassen sich darüber Transaktionsketten bilden (bei Beendigung einer TA wird die nächste angestoßen) [16]. Zusätzlich lassen sich in geschachtelten Transaktionen Alternativen für fehlgeschlagene Subtransak-

tionen definieren. Das ist besonders hilfreich, da eine nie fehlschlagende Ersatztransaktion die leere Transaktion ist. So läßt sich z.B. definieren: Gib das Video aus, falls das nicht geht, gib ein Bild dafür aus. Falls auch das fehlschlägt, dann skip.
Dieser Punkt betrifft nicht nur das Transaktionskonzept, sondern auch das Recovery-Verfahren. Auch hier möchte man bei einem Systemabsturz nicht alle bereits aufgezeichneten Szenen wieder verlieren.

9. Einfluß auf die Sperrgranularitäten:
Multimedia-Daten sind i.a. große Objekte, d.h. sie sind wesentlich größer als eine Speicherseite. Das ist aber, neben der ganzen Relation bzw. Klasse, die klassische Sperrebene in Datenbank- und Betriebssystemen. Für Multimediadaten stellt sich also die Frage der Sperrgranularität neu. Insbesondere ist dabei auch auf die Architektur des darunterliegenden Speichersystems Rücksicht zu nehmen, sonst kann durch das Sperren ganzer Objekte der Vorteil der RAID-Architektur (sektorenweise Verteilung sehr großer Multimedia-Objekte (z.B. eines Videos) auf mehrere Platten wie bei RAID-Level 2, 3, 4 und 5, um bei mehrfachem, minimal zeitversetzten Zugriff auf dasselbe Objekt Plattenarmflattern zu vermeiden) versehentlich wieder aufgehoben werden.

10. Garantieangaben zur Auslieferungsrate von kontinuierlichen Strömen:
Diese Anforderung legt den Quality-of-Service bezüglich des Jitters in der Ausgabe fest. Insbesondere sollten beliebig geforderte Bitraten unterstützt werden. Weitergehende Anforderungen fordern sogar eine dynamische Anpaßbarkeit der Auslieferungsrate an die jeweils aktuell verfügbare Netzkapazität.

11. Unterstützung fortschrittlicher Retrievalarten:
Dazu gehört nicht nur eine attributbasierte Suche auf den Beschreibungsattributen, sondern insbesondere eine inhaltsbasierte Suche, zumindest in der Form von Volltextsuche auf Texten, sowie eine navigierende Suche, sofern das System eine Verlinkung unterstützt.

Diese Liste stellt eine Maximalforderung dar. Heutige Systeme unterstützen -wenn überhaupt - nur Teile davon.

4 Architekturen für Multimedia-Datenbanksysteme

Multimedia-Datenbanksysteme können heute in vier verschiedenen Architekturvarianten unterteilt werden: Multimedia-Systeme mit einem relationalem Datenbanksystem als Backend, Multimedia-Systeme mit einem objektorientiertem Datenbanksystem als Backend, Multimedia-Systeme als Kopplung von Medien-Server und Datenbanksystem sowie integrierte Multimedia-Datenbanksysteme. Wir stellen diese vier Techniken, einschließlich ihrer Vor- und Nachteile, im folgenden kurz vor.

4.1 Multimedia-DBS mit relationalem DBS als Backend

Diese Architektur zeichnet sich aus durch eine Multimedia/Hypermedia-Schicht als

Frontend zu einem relationalem DBS. Alle Hypermedia-Aktionen werden intern in E-SQL umgesetzt und an das relationale Datenbanksystem durchgereicht.

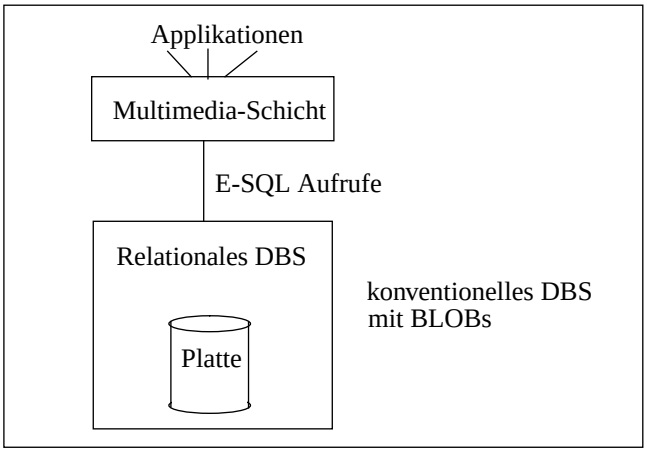

Abb. 2: Multimedia-DBS mit relationalem DBS als Backend

Im einfachsten Fall handelt sich dabei um ein Datenbanksystem, das lediglich über *binary large objects* (BLOBs) als Datentyp verfügt. Dabei sollten die BLOBs in der Datenbank gespeichert sein, nicht nur vom Datenbanksystem verwaltet werden (keine Pointer ins Filesystem!). In dieser Minimalvariante ist aber keine feinere Strukturierung der Daten in den BLOBs im Datenbanksystem möglich. Weder gibt es im Datenbanksystem Operationen auf BLOBs, dadurch sind keine Vorselektionen, keine Inhaltssuche und keine Indexe auf BLOBs möglich, noch kann eine Unterscheidung zwischen kontinuierlichen und diskreten Daten und ihrer unterschiedlichen Auslieferungsart getroffen werden. BLOBs können nur als Ganzes gelesen und verarbeitet werden, ein Teilzugriff oder ein kontinuierliches Auslesen (Streaming-Verfahren) für Videos ist nicht möglich.

Daher werden oft erweiterbare relationale Datenbanksysteme als Backends vorgeschlagen. Diese erlauben die Definition zusätzlicher, anwendungsbezogener Datentypen als Domänen für Attribute. Diese Domänen müssen zusammengesetzte Datentypen unterstützen, nicht nur Synonyme für BLOBs mit Constraints, damit z.B. obiger Datentyp `IMAGE` und analog `VIDEO`, `AUDIO`, `GRAPHIC`, `TEXT`, etc. eingesetzt werden können. Auf ihnen sollten spezielle Zugriffsfunktionen definierbar sein. Diese benutzerdefinierten Datentypen sollten wie jeder anderen build-in Typ benutzt werden können. Objekt-relationale Datenbanksysteme [14] beiten dazu oft Datablades an, mittels derer neue Datentypen über eine erweiterte DDL definiert werden können (allerdings keine real-time-Operationen).

4.2 Multimedia-DBS mit objektorientiertem DBS als Backend

Diese Architekturvariante benutzt ein objektorientiertes Datenbanksystem als Bakkend. Der Vorteil dieses Ansatzes liegt in der leichteren Integrierbarkeit neuer Medien-Datentypen, da diese als persistente Klassen innerhalb einer Klassenhierarchie definiert und in das Datenbankschema integriert werden können. Klassen haben inhärent auch eigene vererbbare Zugriffs-methoden, so daß darüber auch eine reichere Semantik definierbar ist. Sofern das Datenbanksystem aber keine kontinuierliche Auslieferung und keine interne Synchronisation unterstützt, bietet diese Architekturvariante zwar eine leichtere Implementierbarkeit, aber keine größere konzeptionelle Mächtigkeit als die relationale Variante. Das Auslesen der Daten bleibt dann BLOB-orientiert.

Eine Multimedia/Hypermedia-Schicht bildet das Frontend zum OO-DBS. Wie stark diese als eigene Schicht ausgeprägt ist, oder ob sie in Form persistenter (instanzloser) Klassen im OO-DBS gehalten wird, ist Implementationsdetail.

Die Zugriffsschnittstellen sind OQL (= objektorientierte Erweiterung von SQL) oder persistentes C++. Da sich im letzteren Fall Anwendungssystemprogrammiersprache und Datenbank-Querysprache decken, hat die Multimedia-Schicht erheblich weniger Code als im relationalen Fall. Ebenso kann die Linkverfolgung über Pointerverfolgung statt über Joins definiert werden. Auch bei 1:n- und n:m-Links, da *list-of* und *set-of* Konstrukte möglich sind (keine 1. Normalform). Dabei ist jedoch Vorsicht vor einem Trugschluß geboten: Auch persistente Pointerverfolgung benötigt aufgrund der indirekten Adressierung i.a. mehrere Plattenzugriffe (mindestens zwei) und aufgrund des Pointerswizzlings zur Konvertierung in den DB-Cache einen entsprechenden Rechenaufwand. Objektorientierte Pointerverfolgung ist also nicht per se schneller als relationale Indexjoin-Berechnung.

4.3 Multimedia-DBS als Kopplung von Medien-Server und DBS

Beide obige Backend-Architekturen können die kontinuierliche Auslieferung von Videos und Audios im Streaming-Verfahren (= kontinuierliche Auslieferung, nicht *on block*) und die Synchronisation zwischen ihnen nur unzureichend unterstützten. Videos müssen in BLOBs abgelegt werden und können nur als ganzes in die Multimedia-Schicht geholt werden, die dann erst ein Streaming-Protokoll aufbauen kann, dazu aber den ganzen BLOB zwischenspeichern muß. Auch die, zur Überbrückung der großen Ladezeit und des u.U. immensen Pufferbereichs gelegentlich eingesetzte Zerstückelung der Videos in Teilsequenzen, die dann in je einem BLOB abgelegt werden, ist, aufgrund der nichtgarantierbaren nahtlosen Anschlußauslieferung, nur ein Work-around (allerdings der beste, der unter dieser Architektur möglich ist).

Oft wird daher, falls die Multimedia-Datenbank ihren Schwerpunkt auf der Speicherung und Auslieferung kontinuierlicher Daten hat, eine Kopplung mit einem speziellen Medien-Server vorgeschlagen. Der Medien-Server, z.B. ein Video-Server oder Audio-Server, speichert die Videos bzw. Audios und kann sie als kontinuierlichen Strom ausliefern. Manchmal werden auch Bilder wegen ihrer Größe auf spezielle Bild-

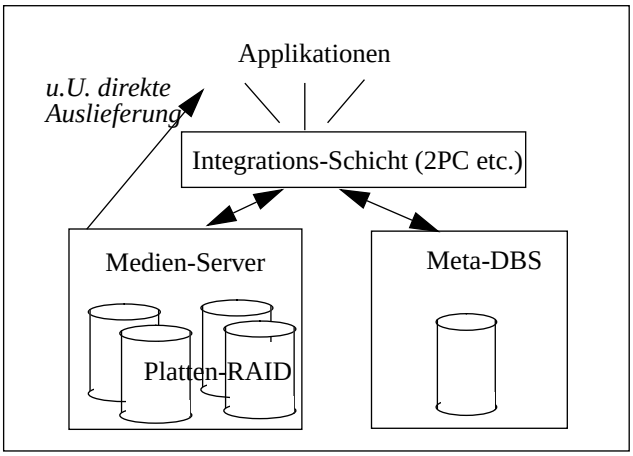

Abb. 3: Multimedia-DBS als Kopplung von Medien-Server und DBS

Datenbanken ausgelagert. Diese fungieren dann ebenfalls als Medien-Server. Medien-Server sind spezialisierte Speichersysteme für einen oder wenige Medientypen. Sie realisieren im wesentlichen die untersten drei Schichten eines Datenbanksystems (Zugriffsstrukturverwalter, Segmentverwalter und Peripheriespeicherverwalter). Manchmal sind sie auch gänzlich Datei-basiert. Die Registrierungsdaten und Beschreibungsdaten zu diesen Medienobjekten werden dagegen im gekoppelten Datenbanksystem (meist relational, seltener objektorientiert) gehalten und verarbeitet. Die Suche geht also immer zuerst in die Beschreibungs- oder Meta-Datenbank. Die multimedialen Objekte werden dort nur durch ihren Identifikator (UID) repräsentiert. Für ihre Auslieferung und synchrone Anzeige sind dann die speziellen Medien-Server zuständig. Diese Architektur erfordert ein 2-Phasen-Commit-Protokoll (2PC) zwischen Medien-Server und Meta-Datenbank. Die Medien-Server verwenden aufgrund der großen Speichermengen, einer besseren Verfügbarkeit oder einer höheren Parallelität Platten-RAIDs (Redundant Arrays of Inexpensive Discs) oder eine hierarchisches Speichersystem mit Platten, optischen Speichern, Jukeboxen und evtl. Bandarchiven. Wir erhalten eine Architektur wie sie in Abbildung 3 dargestellt ist. Der Übergang von dieser Architektur zu verteilten Multimedia-Datenbanksystemen ist fließend.

Beispiele für Medien-Server mit ADT- und Datenbank-Funktionalität stellen die Systeme *MOSS* [3] und dessen Nachfolgeprojekt *Kangaroo* [6] dar. Sogenannte *Single-medium data objects* (SMOs) werden in *medium-specific abstract datatypes* (MADTs) wie `TEXT`, `IMAGE`, `GRAPHIC`, `SOUND` und `VIDEO` formatunabhängig und applikationsneutral gespeichert.

4.4 Architektur von integrierten Multimedia-Datenbanksystemen

I.a. unterstützen Medien-Server aber nur selten eine volle Datenbankfunktionalität, oft sind sie sogar nur Datei-basiert. Will man alle Vorteile und Anforderungen an Mul-

timedia-Datenbanksysteme in einem System zusammenfassen und zur Verfügung stellen, kommt man zu einer integrierten Architektur. Dabei treten alle Teile und Ebenen konventioneller Datenbanksysteme wieder auf, wenn auch mit teils deutlich erweiterter und an die Anforderungen kontinuierlicher Daten angepaßter Funktionalität.

Ein integriertes Multimedia-Datenbanksystem kann in folgende vier Ebenen unterteilt werden (vgl. Abbildung 4): Zuunterst liegt das hierarchische Speichersystem mit Plattenspeichern, optischen Speichern, evtl. Jukeboxen und Bandarchiven. Darüber sitzt die Multimedia-Engine. Sie entspricht in ihrer Basisfunktionalität den klassischen Datenbankmanagementsystemen, im Idealfall erweitert um folgende Anforderungen:

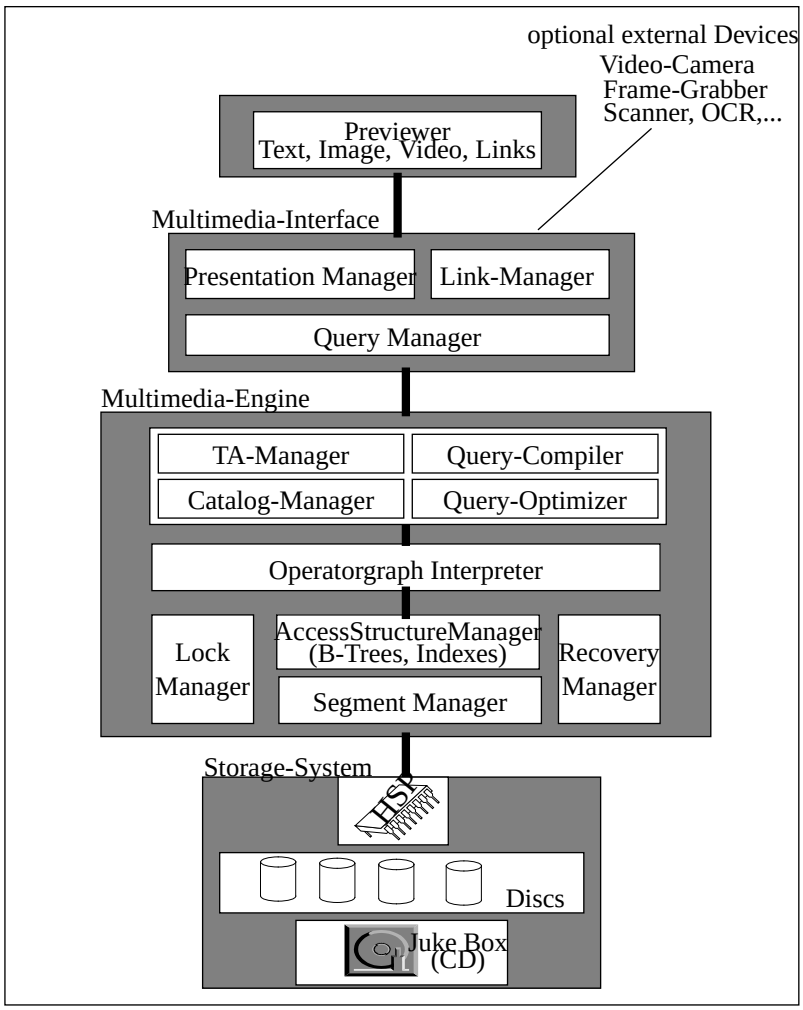

Abb. 4: Architektur eines integrierten Multimedia-Datenbanksystems.

- Geschachtelte Transaktionen, lange Transaktionen, anpaßbare Sperrgranulate,
- erweiterbare abstrakte Datentypen (ADTs),
- Zugriff auf Teildaten,
- kontinuierliche Ausgabeströme,
- geeignete Indexstrukturen für Multimedia-Objekte, insbesondere eine Indizierung über der Zeitachse,
- geräte- und formatunabhängiges Retrieval, sowie
- Multimedia-Clustering auf der Speicherebene.

Darüber liegt das Multimedia-Interface. Seine Hauptaufgabe ist die Bereitstellung der Multimedia- und Hypermedia-Applikationslogik nach oben und deren Übersetzung in die E-SQL-, OQL- oder Operatorgraph-Schnittstelle der Multimedia-Engine nach unten. In WWW-Einbettungen wird diese Schicht auch häufig als anwendungsunabhängiger Teil des Applikationservers bezeichnet. Dieser kann in einer Client/Server-Umgebung bereits auf einem anderen Rechner liegen. Zuoberst sitzen die Browser und Previewer der Benutzer. Die anwendungsabhängige Applikationslogik kann entweder mittels *Foreign Functions* in das Multimedia-Interface oder die Multimedia-Engine integriert sein, als eigene Zwischenschicht zwischen Multimedia-Interface und Previewer liegen oder wie z.B. bei der Verwendung von Java-Browsern auch in diese ausgelagert sein.

5 Das Multimedia-Datenbanksystem MultiMAP und seine Anwendungen

MultiMAP ist ein interaktives, erweiterbares Hypermedia-Datenbanksystem, in dem Texte, Bilder, beliebige Objekte in den Bildern, Audios und Videos gespeichert und durch Links miteinander verknüpft werden können. Ein Schwerpunkt ist dabei die Unterstützung einer schnellen und einfachen (mausunterstützten) Erstellung von Anwendungen, denn die Personalkosten zur Erstellung und Pflege einer Anwendung erweisen sich heute zunehmend als der wesentlichste Faktor an den Gesamtkosten eines Informationssystems.

Die Verwendung der MultiMAP-Datenbank für ein Multimedia-Informationssystem bietet gegenüber rein file-basierten Techniken (wie z.B. WWW) eine Reihe von Vorteilen wie z.B. Konsistenz, insbesondere inhärent konsistente Links, effiziente Zugriffspfade und Indexstrukturen für komplexe Suchanfragen, transaktionsgeschützter Mehrbenutzerbetrieb, Recovery und effiziente und einfache Verwaltung auch sehr großer Datenmengen. Alle Daten werden komplett im Datenbanksystem gespeichert. Es gibt also keine Auslagerungen und Verweise in das Filesystem. Das erhöht den Zugriffsschutz und die Sicherheit. Neben diesen Datenbank-basierten Vorteilen bietet MultiMAP aber auch noch eine Reihe an Multimedia- und Hypermedia-Erweiterungen:

Das Linkkonzept von MultiMAP geht weit über die üblichen WWW-Links hinaus:

1. Links sind eigene Entitäten, d.h. sie haben Attribute und können typisiert werden.
2. Unterstützung beliebiger n:m-Links:
   Es werden nicht nur 1:1 sondern beliebige n:m-Links unterstützt. Das stellt eine Erweiterung klassischer Hypertextstrukturen dar, die nur 1:1-Links verwalten können. Bei disjunktiven 1:n-Links ist zur Linkverfolgung eine zusätzliche Auswahlkomponente nötig.
3. Unterstützung von uni- und bidirektionalen Links.
4. Erweiterung des Hypertextkonzept auf beliebige graphische Objekte:
   Linkausgänge und Linkziele können beliebig umrandete Objekte auf einem Bild oder einem Video sein (beispielsweise ein Flußverlauf oder ein beliebiges Grundstück auf einer Landkarte). Insbesondere können sich die Linkquellenanker beliebig überschneiden und inkludieren. Alle bei einem Klick getroffen Anker werden verfolgt.
5. Auch Textlinks können sich überschneiden und inkludieren. Dies ist insbesondere in einer Anwendung zur Sprachanalyse von Texten von Bedeutung, wenn man mehrdeutige grammatische Analysen korrekt repräsentieren will.

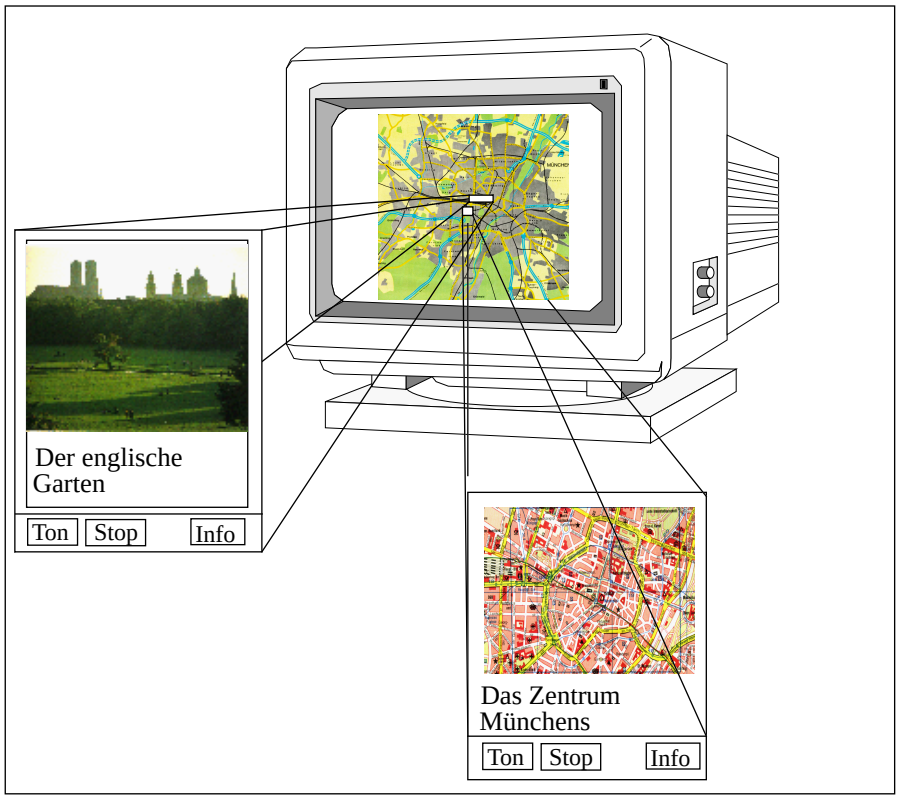

Abb. 5: Anwendung MultiMAP mit interaktiven Landkarten, beliebig langen Texten, Bildern, Audios und Videos (Englischer Garten).

6. Inter- und Intra-Dokumentlinks werden unterstützt, insbesondere Intra-Bilder und Intra-Video-Links, d.h. Linkausgang und Linkziel sind Objekte auf demselben Bild. Der Zielanker wird dann blinkend dargestellt.
7. Zusätzlich zu den Links kann auch durch Volltextrecherche auf allen Texten, Bild- und Objektnamen in der Datenbank gesucht werden. Die Volltextsuche ist in die Objektrecherche integriert und verhält sich wie zusätzliche dynamische Links.
8. Alle Objekte und Links werden innerhalb der Datenbank verwaltet und genügen daher der referentiellen Integrität. Sie sind also auch nach Updates und Löschungen von Zielknoten stets konsistent.

Konzeptionell basiert MultiMAP auf dem Dexter Modell [1] und dessen Erweiterung für kontinuierliche Daten, dem Amsterdam Modell [2]. Der Storage-Layer dieser Modelle wurde komplett in Relationen (bzw. persistente Klassen) überführt und optimiert und stellt so eine theoretisch fundierte Basis des Systems dar.

MultiMAP stützt sich für die Datenhaltung auf kommerzielle Datenbanksysteme als Backend ab. Dabei können alle für Multimedia-Anwendungen relevanten Datenbankmodelle (relationale, objektorientierte und objekt-relationale DBS) eingesetzt werden. In der Grundvariante kommt das relationale Datenbanksystem TransBase™ zum Einsatz. Es wurden aber zu Vergleichszwecke auch eine objektorientierte Variante, basierend auf dem objektorientierten Datenbanksystem O_2, und eine DB/2-Version entwikkelt. In letzterer kommen objekt-relationale Techniken zum Einsatz.

Von jedem Multimedia-Datenbanksystem wird heute eine WWW-Schnittstelle gefordert. Mit einem WWW-Anschluß erhält man die großen Vorteile der clientseitigen Plattformunabhängigkeit der Anwendung, der Ausnutzung schneller WAN-Protokolle und der weiten Verbreitung. Aufgrund der wesentlich erweiterten Linkfunktionalität von MultiMAP kann die Oberfläche nicht einfach auf HTML abgebildet werden. Große Teile des Clients wurden daher in JAVA geschrieben. Die Entwicklung der WWW-Varianten spiegelt die Entwicklung der unterschiedlichen Techniken der letzten 2 bis 3 Jahre zu Datenbankanbindungen ans WWW wieder, von einfachen CGI-Anbindungen bis zu JAVA- und JDBC-Kopplungen. Während die erste MultiMAP-Version noch stand-alone war, arbeitete die zweite Version mit einer zweistufigen, kombinierten CGI- und JAVA-Kopplung. Die dritte benutzt eine dreistufige Architektur, wobei ein Gateway zum Einsatz kommt und der Datenbankserver nicht mehr auf derselben Maschine zu liegen braucht, wie der WWW-Server und der Applikationsserver, der die eigentliche Anwendungslogik enthält. In einem Prototyp werden auch JDBC-Kopplung getestet.[1]

Zahlreiche weitere Erweiterungen wurden in MultiMAP integriert:

- So wird eine dynamische *Personalisierung* der Multimediadaten entsprechend

1. Die Entwicklung der WWW-Varianten und der Netzbetrieb unter Einsatz bei externen Anwendern wird z.Zt. vom DFN (Deutsches Forschungsnetz) im Entwicklungsprogramm “Fortgeschrittene Anwendungen” im Bereich “Multimedia-Teledienste” gefördert.

des jeweiligen Benutzerverhaltens angeboten. Die Personalisierung verhält sich adaptiv, so daß sie auch auf Verschiebungen von Benutzerinteressen entsprechend reagieren kann.

- Jeder Benutzer kann beliebige *Annotationen* an Multimedia-Objekte heften und diese auch für eine Gruppe an Personen freigeben. Dadurch wird jeder Benutzer auch zum potentiellen Schreiber in der Datenbank. Das unterscheidet MultiMAP von reinen Multimedia-Retrieval-Systemen.
- Eine erweiterte *Volltextsuche* ist integriert, die sowohl lokal, bezüglich des aktuell dargestellten Informationsblocks, als auch global über den gesamten Informationsraum angestoßen werden kann.

Eine Reihe an verschiedenartigen Anwendungen setzen auf diesem Hypermedia-Datenbanksystem bereits auf:

1. **MultiMAP:** (Namensgleichheit, da dies die erste Anwendung auf dem System war.) MultiMAP ist eine multimediale Aufbereitung von Landkarten und Stadtplänen mit Fotos, Skizzen, Texten, Ton- und Sprachausgabe und rekursiv weiteren Detailkarten. Anwendungen liegen in Stadtinformationssystemen, Biotopkartierungen oder auch im administrativen Bereich für Raumordnungsverfahren. Dabei kann nicht nur stichwortunterstützt, also über textuelle Links, gesucht werden, sondern insbesondere auch durch Anklicken beliebiger Objekte (Straßen, Gebäude, Grundstücke, Biotope, etc.) auf den Landkarten. Die Landkarten sind als Pixelbilder abgelegt und z. B. bezüglich der Straßenführung mit Vektordaten hinterlegt. Ein Stadtführer von München ist weitgehend fertig. Eine Anfrage zur Unterstützung von Raumordnungsverfahren vom Lehrstuhl für Raumforschung, Raumordnung und Landesplanung der TU München liegt vor.
2. **MultiMED**: Ein weiteres Anwendungsgebiet erstreckt sich im medizinischen Bereich auf multimedial aufbereitete Röntgenbilder und CT-Aufnahmen, einschließlich Detailbilder und verbaler oder textueller Befundung. Ein Anschluß an die Stationsdatenbank mit der Patientenstammdatenverwaltung ist integriert. Ein Prototyp wurde in Zusammenarbeit mit Orthopädischen Abteilung des St. Bernward Krankenhauses in Hildesheim erstellt [9].
3. **MultiBHT**: Ein drittes Anwendungsgebiet liegt in der multimedialen Aufbereitung von Ergebnissen der Sprachanalyse bei Linguisten mit dem Ziel der Erstellung korrekter Grammatiken und der Erstellung textkritischer Editionen. Eine Anwendung für Althebräisch läuft im Institut für Assyriologie und Hethitologie der LMU München [10].
4. **MultiLIB**: Ein multimedialer Führer durch die Universitätsbibliothek und ihrer Außenstellen und Zweigbibliotheken der LMU München. Ziel ist, den Studenten ein schnelles Auffinden der Standorte der Bücher, der Zugangsberechtigungen und Öffnungszeiten und eine Unterstützung bei den Katalogabfragen zu ermöglichen. Hierzu muß das System von in ganz München verteilten Standorten aufrufbar sein. Das System wird gerade in der Zusammenarbeit mit der Universitätsbibliothek aufgebaut.

Die beiden Hauptanwendungen konzentrieren sich derzeit auf MultiLIB und MultiBHT. Während sich die Anwendung MultiLIB mehr an ein breites Publikum, insbesondere an Studenten, wendet, ist MultiBHT eine wissenschaftliche Anwendung. Daraus ergibt sich ein anderes Nutzerprofil: Bei MultiBHT haben wir wenige Nutzer mit intensiveren Sitzungen und sehr unterschiedlich großen, zu übertragenden Datenmengen, wobei jeder gleichzeitig Recherchierer und Ersteller ist. Also mit Lastspitzen in beiden Richtungen. Bei MultiLIB haben wir dagegen sehr viele Nutzer, alle Leser, keine Schreiber, mit sehr ähnlichen Anfragen. Während man es bei MultiLIB eher mit Gelegenheitsnutzer zu tun hat, sitzen bei MultiBHT Experten an der Anwendung.

Zusammenfassend läßt sich die Funktionalität von MultiMAP damit folgendermaßen umreißen:

- Vielfältige Suchmöglichkeiten: Suche über Linkverfolgung (navigierend), über kontextabhängige Mausklicks in Bildern und Videos, über eine eigene Objektrecherche und über globale und lokale Volltextsuche;
- Eine extrem einfache und schnelle Erstellung von Anwendungen durch das Autorensystem;
- Integrierte und optimierte Ablage und Verwaltung aller Daten, auch der Bilder, Töne und Links in der Datenbank, dadurch hohe Effizienz auch bei großen Datenmengen;
- Transaktionsorientierter Multiuser-Betrieb auf der Datenbank auch bei Anfragen und Updates über das (zustandslose) WWW;
- Volle WWW-Oberfläche unter Verwendung von JAVA in einer mehrstufigen Architektur;
- Referentielle Integrität der Links bei einem wesentlich erweiterten Linkkonzept;
- CD-Fähigkeit und volle Recovery-Fähigkeit.

6 Zusammenfassung

Ausgehend von der Architektur konventioneller Datenbanksysteme stellten wir die für Multimedia-Datenbanksysteme notwendigen Erweiterungen vor. Diese Liste stellt eine Maximalforderung dar und wird gegenwärtig in keinem System voll unterstützt. Dennoch zeigt sie die Richtung der Entwicklung innerhalb der Multimedia-Datenbanksysteme auf. Gegenwärtig können Multimedia-Datenbanksysteme in vier verschiedene Architekturen klassifiziert werden. Wir diskutierten ihre Vor- und Nachteile im einzelnen. Schließlich stellten wir als ein Beispiel unser eigenes System MultiMAP vor. MultiMAP ist eine Verknüpfung der Architekturkonzepte Backendarchitektur und Kopplung von Medien-Server und DBS. Einige Einsatzgebiete von MultiMAP wurden ebenfalls vorgestellt.

7 Literatur

[1] Halasz F., Schwartz M.: *The Dexter Hypertext Reference Model*, Communications of the ACM, Vol. 37, No. 2, Feb. 1994, pp. 30-39 (Erstveröffentl. 1990)

[2] Hardman L., Bulterman D, van Rossum G.: *The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model*, Communications of the ACM, Vol. 37, No. 2, Feb. 1994, pp. 50-62

[3] Käckenhoff R., Merten D., Meyer-Wegener K.: *MOSS as a Multimedia-Object Server - Extended Summary*, Proc 2nd. Int. Workshop Multimedia: Advanced Teleservices and High Speed Communication Architectures (IWACA 93), Heidelberg 26.-28.9.94, Springer, LNCS 869, pp. 413-425

[4] Khoshafian S., Baker A.: *Multimedia and Imaging Databases*, Morgan Kaufmann Publishers, 1996

[5] Koegel-Buford J. (Ed.): *Multimedia Systems*, Addison-Wesley, 1994

[6] Marder U., Robbert G.: The Kangaroo Project - Enhaincing a Multimediaa Server with Data Independence, in: Wissenschaftliche Beiträge zur Informatik, TU Dresden, Band 10, Heft 1, 1997 pp. 82-85

[7] Meyer-Wegener K.: *Multimedia-Datenbanksysteme*, B.G.Teubner, 1991

[8] Moss, J. E.: *Nested transactions : an approach to reliable distributed computing*, MIT Press, Cambridge, Mass., 1985

[9] Specht G., Bauer M.: *MultiMED - ein Multimedia-Datenbanksystem zur Aus- und Weiterbildung in der Röntgendiagnostik in der Orthopädie*, in Schoop, Witt, Glowalla (Ed.): Hypermedia in der Aus- und Weiterbildung, UVK-Verlag, Konstanz, 1995, pp. 209-210

[10] Specht G.: *MultiBHT - ein multimediales Datenbanksystem zur Sprachanalyse*, Fakultät für Alterstumskunde und Kulturwissenschaften, LMU München, 1995, 9 S.

[11] Specht G.: *Complexity Analysis of Link Navigation inDexter Based Hypermedia Database System*, International Journal INFORMATICA, Vilnius, Vol. 8 No. 1, 1997, pp. 23-42

[12] Specht G., Zimmermann S., Clausnitzer A.: *Introducing Parallelism in Multimedia Database Systems*, Proc of the 2nd. Int. Symposium on Parallel Algorithms/ Architectures Synthesis (PAS 97), 17.-21.3.1997 in Aizu-Wakamatsu, Japan, IEEE Computer Society Press, 1997, pp. 348 - 355

[13] Specht G.: *Multimedia-Datenbanksysteme: Modelle - Architekturen - Retrieval*, Habilitationsschrift, Technische Universität München 1998

[14] Stonebraker M.: *Object-relational DBMSs: the next great wave*, Morgan Kaufmann Publishers, 1996

[15] Subrahmanian V.S.: *Principles of Multimedia Database Systems*, Morgan Kaufmann Publishers, 1998

[16] Weikum G.: *Transaktionen in Datenbanksystemen*, Addison-Wesley, 1988

[17] Zaniolo C., Ceri S., Faloutsos Ch., Snodgrass R.T., Subrahmanian V.S., Zicari R.: *Advanced Database Systems*, Morgan Kaufmann Publishers, 1997