Multimedia-Datenbanken

Kapitel 13: Implementierung

Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Institut für Informatik
Lehrstuhl für Informatik 6 (Datenbanksysteme)

Prof. Dr. Klaus Meyer-Wegener

Wintersemester 2002 / 2003

Technische Universität Kaiserslautern
Fachbereich Informatik
AG Datenbanken und Informationssysteme

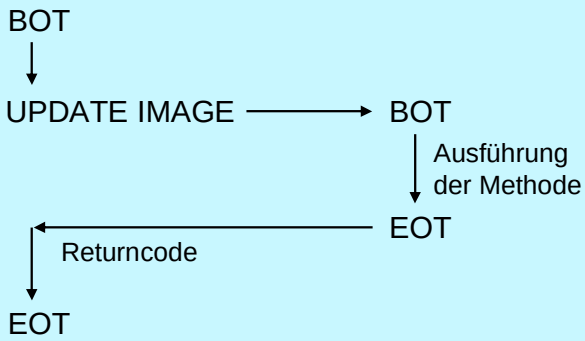
Dr. Ulrich Marder

Wintersemester 2003 / 2004

13.1 Transaktionskonzept

- **ändernde Operationen auf Abstrakten Datentypen**
  - müssen atomar sein (Einkapselung!)

  → **geschachtelte Transaktionen**



  - sonst bei Scheitern der Methode rufende Transaktion mit zurücksetzen
    - Programmierer wird über Returncode informiert
  - „Alles oder nichts“ auch nicht immer erwünscht (z. B. Videoaufnahme)

Transaktionskonzept (2)

- **Mehrbenutzerbetrieb**
  - ganze Medienobjekte als Einheit sperren
    - gemeinsames Ändern von Texten, Bildern etc. in der DB nicht zu erwarten
    - oder durch Aufteilung in mehrere Objekte (Kapitel, Szenen bei Video) und Aggregation unterstützt
- **Implementierung**
  - mit wenigen Operationen große Datenmengen änderbar
    - einzelne Attributwerte belegen u. U. etliche Seiten
  - kein Update in Place!
    - neue Blöcke für neue Werte und Umschaltung durch Pointer
    - vgl. ORION oder Schattenspeicherkonzept
  - Seitenprotokollierung
    - hier evtl. günstiger als Eintragsprotokollierung

13.2 Speicherungsstrukturen

- **Abbildung von Sätzen (Tupeln) auf Seiten**
  - können jetzt sehr lang werden (mehrere GB)
  - auf jeden Fall > 1 Seite
- **Änderung von Abschnitten in einem Attributwert**
  - z. B.: Einfügen einer Textzeile, Ersetzen eines Bildausschnitts durch ein anderes Bild („Einblendung“)
  - ohne Kopieren des ganzen Werts realisieren
- **ein Vorschlag: EXODUS [Care86]**
  - "EXtensible Object-oriented Database System"
  - University of Wisconsin, Madison
- **Baustein: Speicherobjektverwalter**
  - Speicherobjekte als Behälter für beliebig lange Sätze

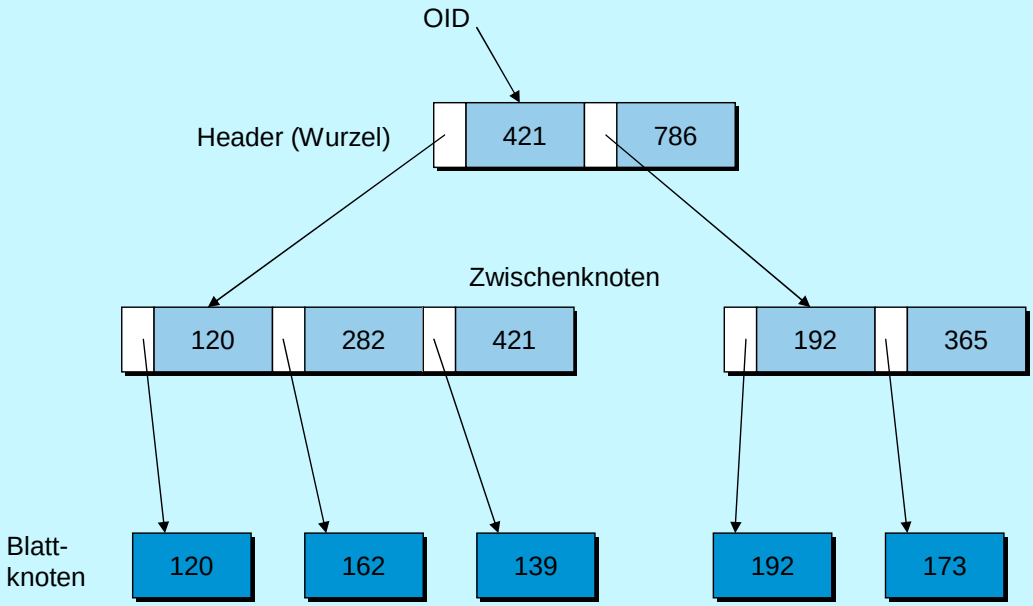
EXODUS-Speicherobjektverwalter

- **Speicherobjekt (SO):**
  - ungetypte, nicht interpretierte Byte-Folge variabler Länge
- **Datei:**
  - Sammlung von Speicherobjekten
- **Operationen:**
  - **Datei-Scan**: gib Objekt-Id des nächsten SO in der Datei
  - **Erzeugen** oder **Löschen** eines SO in einer Datei
  - **Lesen**: gib Zeiger auf Byte-Intervall in SO (implizites Einlesen in Hauptspeicher)
  - **Freigeben** von gelesenen Bytes („unpin“)
  - **Schreiben**: melden, dass gelesene Bytes geändert wurden
  - **Einfügen** und **Löschen** von Bytes an einer Position
  - **Anhängen** von Bytes an das Ende
  - **begin**, **commit**, **abort transaction**

EXODUS-Speicherobjektverwalter (2)

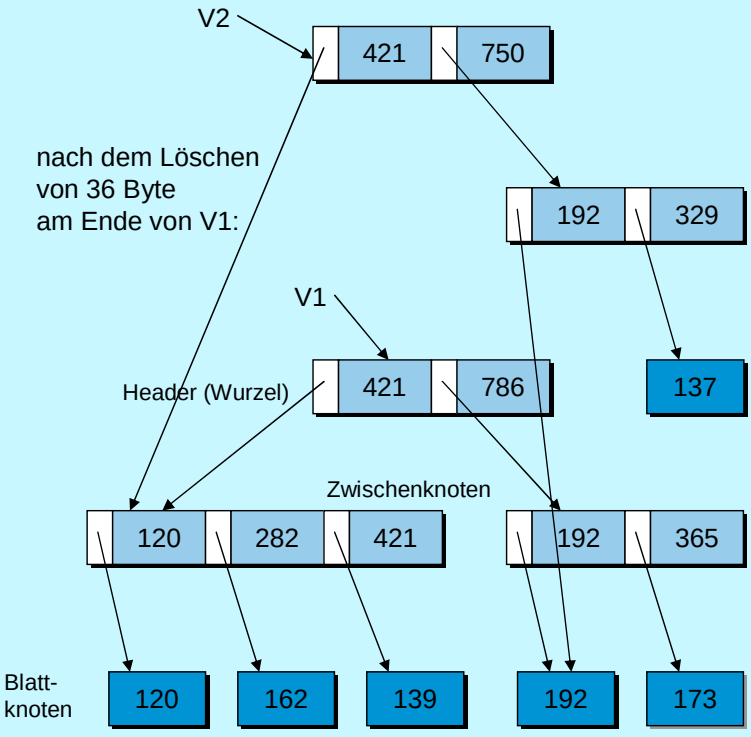
- **außerdem Hinweise zur Leistungssteigerung:**
  - wo ein SO abzulegen ist („in der Nähe von SO x“)
  - wie lang ein SO ungefähr wird
  - ob es allein in einer Seite liegen soll usw.
- **Abbildung auf Seiten:**
  - Objekt-Id (OID) = Seitennr. + Slotnr. (wie TID)
  - Unterscheidung kurze – lange SO (für die aufrufenden Schichten nicht sichtbar)
- **kurze SO**
  - direkt in der Seite der OID abgelegt
- **lange SO**
  - durch **Header** repräsentiert:
    - Liste von Zähler-Seitennr.-Paaren,
      Zähler = lfd. Nummer des letzten SO-Bytes in der Seite
    - Blattknoten exklusiv von diesem SO belegt
    - evtl. Zwischenknoten, die wie Header aufgebaut sind → B^+-Baum über der Byte-Position

EXODUS-Speicherobjektverwalter (3)



EXODUS-Speicherobjektverwalter (4)

- **Versionen**
  - Markierung SO-Header: gehören zu alter Version
  - Kopieren und Ändern nur derjenigen Seiten, die sich in der neuen Version unterscheiden



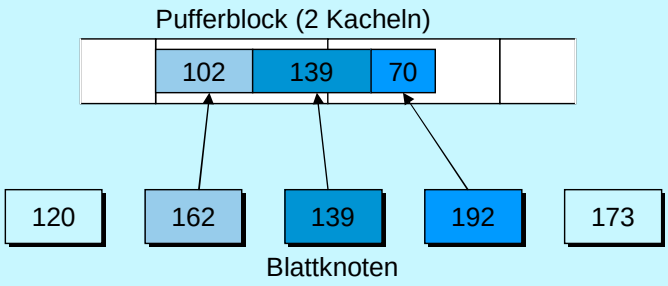
EXODUS-Speicherobjektverwalter (5)

- **Synchronisation**
  - Zwei-Phasen-Sperrprotokoll auf Byte-Intervallen
  - optional auch Sperren des ganzen SO
  - auf den B^+-Bäumen der langen SO Kurzzeitsperren (ohne 2PL, s. Mehrebenen-Transaktion)
- **Wiederherstellung im Fehlerfall**
  - kurze SOs:
    - Aufzeichnung von Before- und After-Images sowie Update in Place
  - lange SOs:
    - Kombination von Schattenspeicherkonzept und logischer Übergangsprotokollierung
    - Erzeugen einer neuen Version, dann Header überschreiben; Name und Parameter der Änderungsoperation werden protokolliert

EXODUS-Speicherobjektverwalter (6)

- **Pufferverwaltung**
  - Belegen ganzer Pufferblöcke (aufeinanderfolgende Kacheln)
  - Byte-Sequenzen eines langen SO aus den Blattknoten lesen und dicht in den Pufferblock packen:

    (Seitengröße 200 Byte, gelesenes Byte-Intervall 181 – 491)



  - Pufferblock wird beschrieben durch „**Scan Descriptor**“:
    - Herkunft der Bytes

13.3 Leistungsverhalten

- **Echtzeitanforderungen bei Ton und Video**
  - Kopier- und Abbildungsvorgänge minimieren (Zielkonflikt mit Datenunabhängigkeit)
- **bisheriger Ansatz (z. B. ORION):**
  - Gerätebedienung an MMDBS übertragen
  - Anwendungsprogramm aus Übertragungs- und Abbildungskette herausnehmen
- **auch innerhalb des MMDBS zusätzlich sinnvoll:**
  - **Pipelining**, d. h. gleichzeitig
    - Block n – 1 schreiben
    - Block n umsetzen, codieren, komprimieren, ...
    - Block n + 1 lesen
  - Unterstützung durch Betriebssystem nötig

13.4 Verwaltung der Speichergeräte

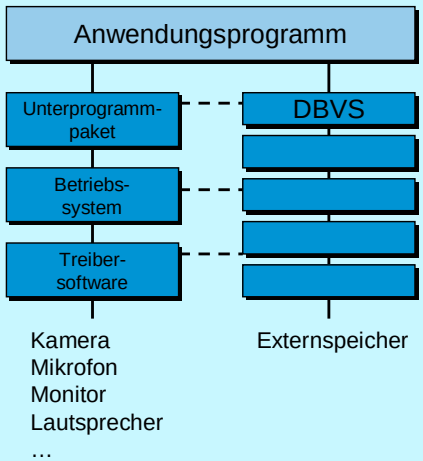
- **neue Geräte mit abweichenden Eigenschaften (WORM, Videorecorder)**
  - wann benutzen? für welche Daten?

1. **Benutzer entscheidet**
   - System bedient die Geräte nur
   - kein „Vergleich“, keine Bewertung der Geräte durch das System
2. **System entscheidet (evtl. als Default)**
   - Benutzer beschreibt Eigenschaften der Daten
   - System beobachtet Zugriff, verlagert ggf. die Daten (Größe, sequenzielles Lesen, Änderungshäufigkeit, ...)

- **Benutzer gibt Zustandsinformation zu den Medienobjekten:**
  - noch in Arbeit
  - stabil, festgeschrieben, freigegeben, …
- **aber auch „interne“ Benutzung durch das System:**
  - Blöcke nie ändern, immer neue benutzen (ORION)
    → auch auf WORM machbar!

13.5 MMDBS-Architektur

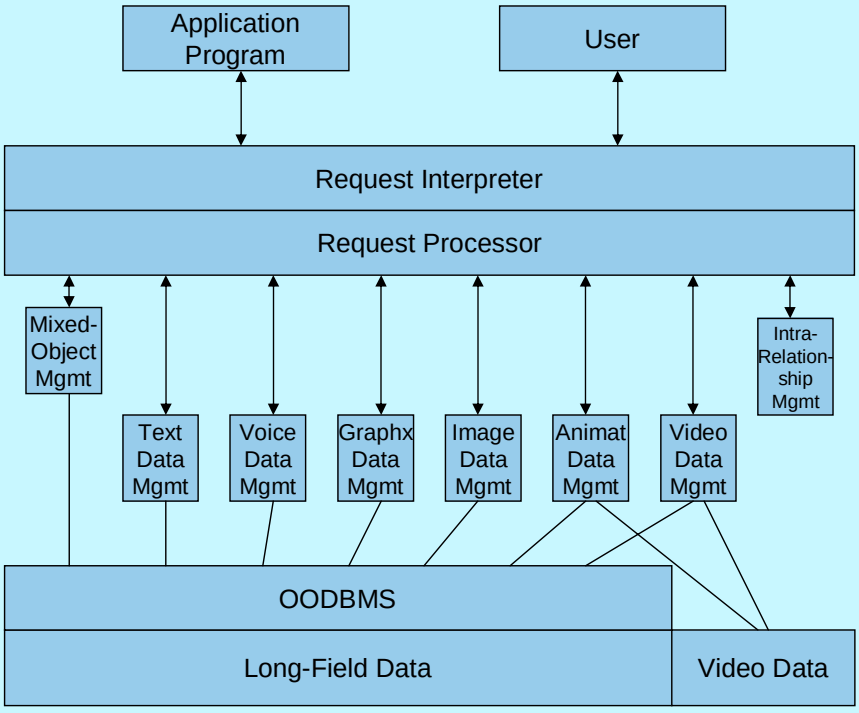
- Ausgangspunkt: 5-Schichten-Architektur
- in gleicher Weise Abstraktionsschichten für E/A-Geräte definieren:



- Übergänge bzw. Integration auf verschiedenen Ebenen möglich!
- Performance: in den tiefen Schichten

MMDBS-Architektur (2)

anderer Vorschlag [Lock88b]:



MMDBS-Architektur (3)

- **Workstation-Server-Kopplung**
  - hohe Übertragungsleistung erforderlich
  - transiente Ein-/Ausgabe ohne Speicherung in der Workstation
    versus
    Auslagerung und (Wieder-) Einlagerung („Überspielen“)

- **unterschiedliche Sichten und Organisationsformen für die Datenbestände denkbar:**
  - Hypermedia
    auf der Workstation
  - relationales oder objektorientiertes DBS
    (mit Mediendatentypen)
    auf dem Server

13.6 Eigene Arbeiten

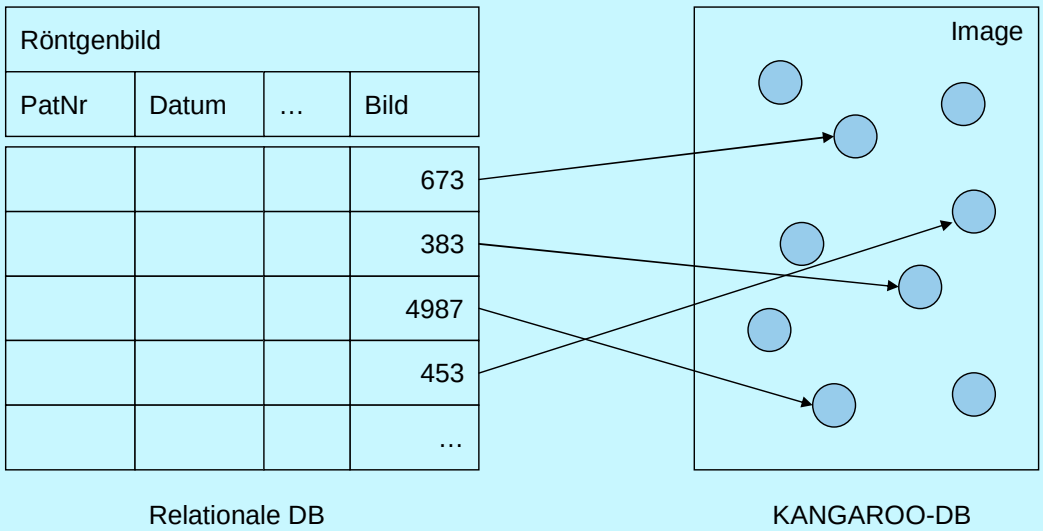
- **KANGAROO**
  - Herausforderung: Datenunabhängigkeit + Echtzeit
  - „Kernel Architecture for Next Generation Archives of Realtime-Operable Objects“
  - ausschließlich Mediendatenobjekte („Single Media Objects“ - SMOs) – keine formatierten Daten, keine Multimedia-Objekte
  - festes Schema, medienspezifische Abstrakte Datentypen (MADTs)
  - inhaltsorientierte Suche
  - flexible Anbindung von Speichergeräten
    (vgl. austauschbare Storage Manager in Starburst)
  - Komprimierungstechniken
  - Pufferverwaltung unter Echtzeit-Bedingungen
    (Ausgabe in Netze und auf Geräte)
  - Ressourcenbelegung
  - Prozess-Konzepte, Zusammenarbeit mit Betriebssystem
  - Performance-Untersuchungen

KANGAROO-Datenmodell (1)

- **fester Satz von MADTs:**
  - TEXT, GRAPHIC, IMAGE, SOUND, VIDEO
- **Objektidentifikatoren**
  - „Media Object Id“ (MOID)
- **Zugriff**
  - über MOID
  - Scan (typgebunden: alle Bilder)
  - Suche (eigener Ansatz: semantisches Matchen von natürlichsprachlichen Inhaltsangaben, s. oben)
- **Einsatz**
  - in Kooperation mit relationalem oder objektorientiertem DBS (s. Projekt Imos unten)

KANGAROO-Datenmodell (2)

- **MOIDs in strukturierten DBS als Attributwerte gespeichert:**



KANGAROO-Datenmodell (3)

- **Benutzung im Anwendungsprogramm:**

```
select Bild into :id from Röntgenbild
where PatNr = 37894578;
```

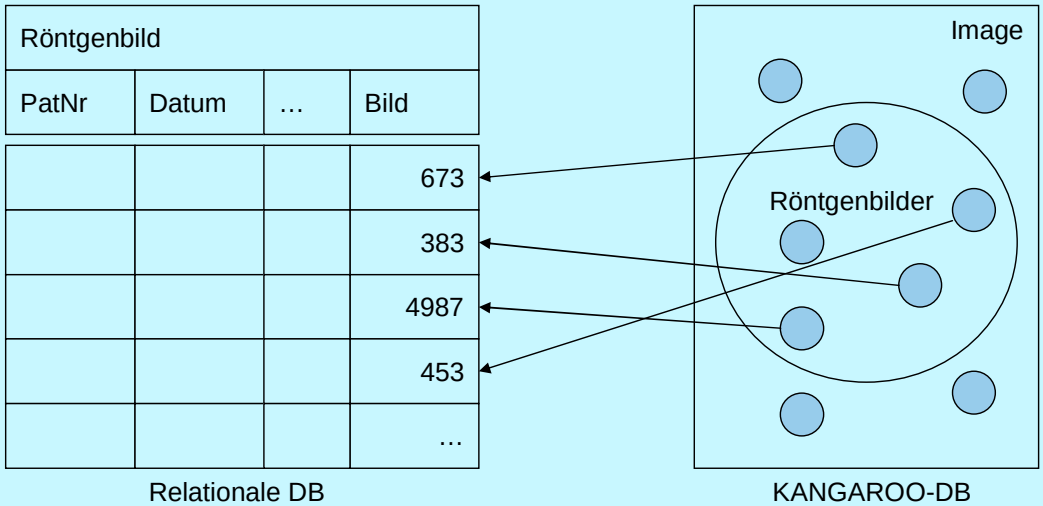
- **dann Zugriff auf KANGAROO:**

```
pr = Image.getPixrect( id );
```

→ Zwei-Phasen-Freigabe (2PC)
→ keine systemübergreifenden Anfragen

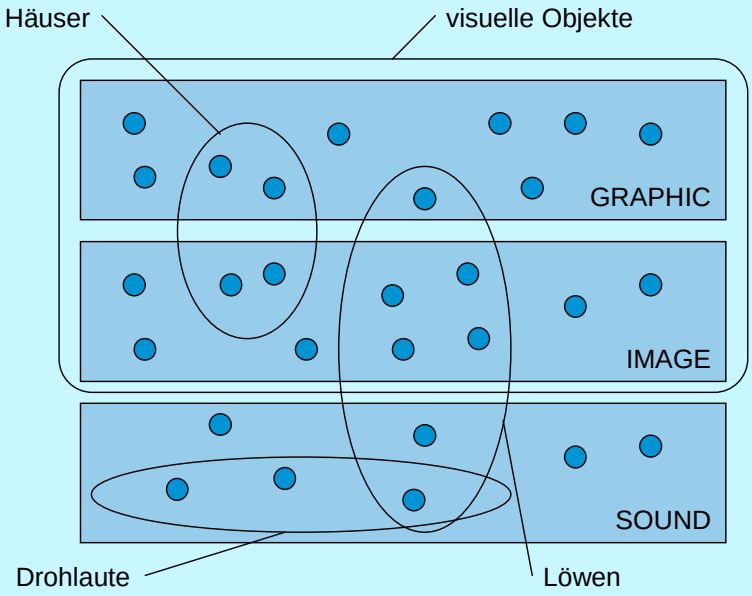
KANGAROO-Datenmodell (4)

- **ABER:**
  - Suchmechanismus in KANGAROO benutzen;
  - nicht in allen Bildern, sondern nur in Röntgenbildern!
  - dann mit MOID in die Relation:



MO-(Such-)Mengen (1)

- haben einen Namen und eine Id
- n:m-Beziehung zu MOs



MO-(Such-)Mengen (2)

- Integritätsbedingungen spezifizierbar:
  - Teilmenge, Disjunktheit
    `Image ⊇ Röntgenbilder`
    `Häuser ∩ Röntgenbilder = ∅`
- zu jedem MADT systemdefinierte Menge
  - mit Sonderbehandlung:
    insert und remove nicht erlaubt
- Operatoren auf den Mengen:
  - insert und remove
  - Scan
  - Suche
  - Vereinigung, Schnitt, Differenz

KANGAROO-Architektur

- **drei Schichten:**
  - Schnittstellen
    - *Interoperabilität:*
      Kooperation mit anderen Servern (Dsmily, DBS)
    - *Anfragen:*
      Suchmengen, Scans
    - *MADTs*:
      Operationen
  - interne Service-Module
    - Systemadministration, Transaktionsverwaltung, Logische Zugriffspfade, Medienobjekte (Basis), Datentransport
  - Speicherungssystem
    - Externspeicherverwaltung, Locking/Logging/Recovery, Physische Indexstrukturen, Objektpufferung mit Formatfilterung, Threads mit Echtzeit-Scheduling
    - Shore

Dsmily

„Distributed Scalable Multimedia Information retrievaL sYstem“

- **Erweiterung des probabilistischen Information Retrievals auf verteilte und heterogene Server**
  - KANGAROO und andere!
  - *heterogen* = verschiedene Indexierungen
  - *skalierbar* = hierarchisch aufgebaut (Baumstruktur), keine monolithischen Server
  - Server muss zu jedem suchbaren Objekt Id und Feature-Menge (z. B. Schlagworte) liefern und Anfrage bei Bedarf in seine (!) Features übersetzen
  - auf höherer Ebene (innere Knoten des Baums) ganze Server durch Features beschrieben
  - *Auswahlkriterium:*
    Anfrage nicht an alle Server weiterleiten, sondern nur an die, die mit der größten Wahrscheinlichkeit relevante Objekte zur globalen Ranking-Liste beitragen

Imos

- **„Interoperabilität von Medienobjektservern“**
  - Anwendungen benötigen formatierte Daten und Mediendatenobjekte
  - dazu auch noch Suche
  - koordinierter, gleichzeitiger Zugriff auf relationale / objektorientierte DB, KANGAROO, Dsmily und evtl. noch andere
  - heterogenes DBS / Multi-DBS
- **Integrationsstufen:**
  - isoliert, nebeneinander
  - verteilte Transaktionen
  - verteilte Anfragen
  - gemeinsame Anfragesprache
- **untersuchen und bewerten**
- **Anforderungen an beteiligte Server ableiten**
- **Client-Server-Interaktion**
  - inkl. Dienstgüte für die Multimedia-Daten

Memo.real und RETAVIC

- **ehemals Teilprojekt des SFB 358 an der TU Dresden**
- **Entwurf von echtzeitfähigen, datenunabhängigen Medienobjekt-Speicherungssystemen**
  - Nachfolge KANGAROO
  - Konvertierungen von Medienobjekten bei der Ausgabe
  - Messungen an den Konvertern
  - mathematische Modelle für das Verhalten der Konverter
    - schwankungsbeschränkte Ereignisströme
    - statistische ratenmonotone Ablaufplanung
    - ungenaue Berechnungen
  - Nutzung eines Echtzeit-Betriebssystems (DROPS)
  - Ermittlung von Engpässen
  - Beschreibung von kritischen Pfaden im Hinblick auf den Einsatz von Spezial-Hardware (DSP, FPGA, ASIC usw.)

COMQUAD-DB

- **DFG-Forschergruppe, ein Teilprojekt in Erlangen**
- **Komponenten-Software mit quantitativen Eigenschaften**
  - Ressourcen-Bedarf
  - Antwortzeit, Durchsatz
- **und Adaption**
  - Austausch von Komponenten
  - Anpassung von Komponenten durch Setzen von Parametern
- **Teilprojekt DB**
  - Abbildung und Durchsetzung quantitativer Anforderungen an Komponenten
    - „Damit die Komponente eine Antwortzeit von 2 s garantieren kann, braucht sie einen xy-Prozessor und eine E/A-Rate von mindestens …“
  - speziell für Konverter als Komponenten
    - Bezug zu Memo.real

VirtualMedia

- **Gezielte Auswahl und Planung von Operationen (Komponenten) erfordert Kenntnis der Datenformate und Speichergeräte**
- **Widerspruch zur Formatunabhängigkeit!**
- **Anwender soll sich nur auf Funktionalität (Semantik) und (abstrakte) Dienstgüte konzentrieren müssen**
- **Ansatz: VirtualMedia-Modell**
  - MOs und Operationen durch abstrakte Filtergraphen beschrieben
  - Datenflüsse (Pipelining) durch „Magische Kanäle“
    - repräsentieren „unbekannte“ Operationen, z. B. Formatkonvertierung
    - semantische Äquivalenzrelationen erlauben „Entzauberung“
  - heuristische Optimierung
- **Basis für Multimedia-Metacomputing**
  - verteilte Verarbeitung von Mediendaten mittels unbekannter bzw. dynamisch „entdeckter“ Ressourcen
  - geeignet für moderne Verteilungsarchitekturen (P2P, Webservices, Grid)