Multimedia-Datenbanken

Kapitel 12: Erweiterung objektorientierter DBS

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

Prof. Dr. Klaus Meyer-Wegener

Wintersemester 2002 / 2003

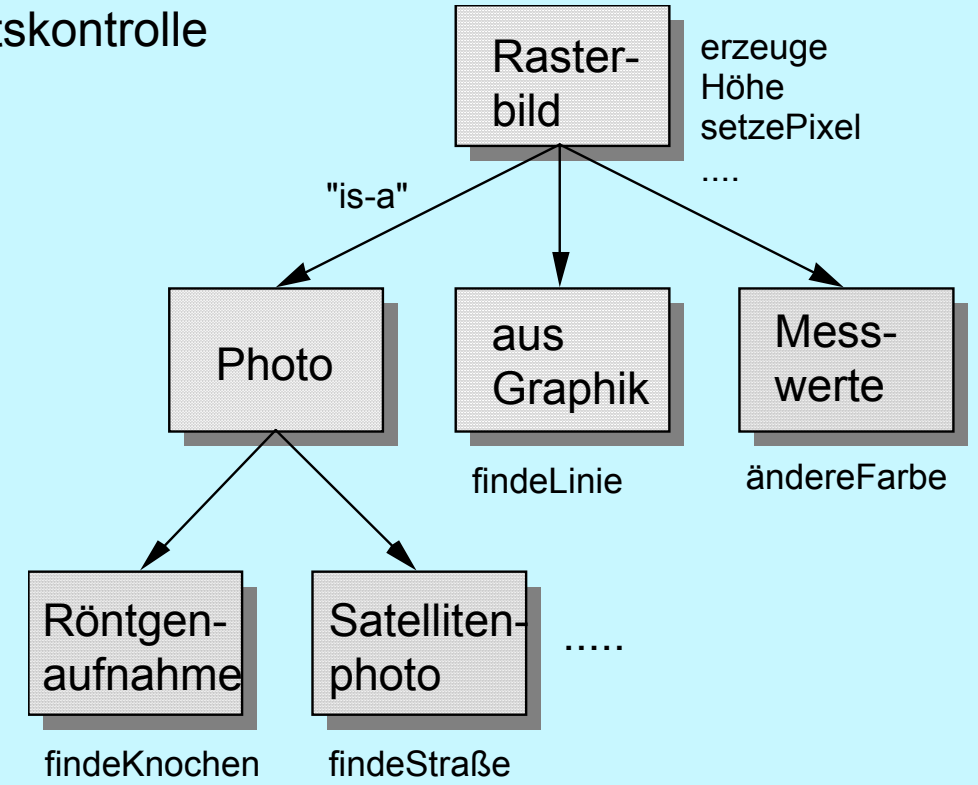
Technische Universität Kaiserslautern
Fachbereich Informatik
AG Datenbanken und Informationssysteme

Dr. Ulrich Marder

Wintersemester 2003 / 2004

12. Erweiterung objektorientierter DBS

- **MM-Datenobjekte sind Instanzen von Klassen**
- **Klassenhierarchie und Vererbung**
  - vereinfachen die Datenmodellierung
  - bieten mehr Integritätskontrolle



Erweiterung objektorientierter DBS (2)

- **Vererbung**
  - alle Methoden, die auf Instanzen der Klasse „Rasterbild“ anwendbar sind (d. h. die oben eingeführten Operationen des Datentyps IMAGE), sind auch auf die Instanzen der Subklassen anwendbar
  - jedes Foto kann als Rasterbild behandelt werden
- **Spezialisierung**
  - die Subklassen können eigene Methoden hinzufügen, die nur auf ihre Instanzen angewendet werden können
  - eine Operation „findeKnochen“ kann auf ein Röntgenbild angewendet werden, aber nicht auf andere Bilder
- **Überdeckung**
  - die Subklassen können Methoden, die sie ererben, neu implementieren
  - wenn man weiß, dass ein Rasterbild aus einer Graphik gewonnen wurde, kann man „findeLinie“ sehr viel einfacher (und zuverlässiger) implementieren, als wenn man von einem beliebigen Rasterbild ausgehen muss

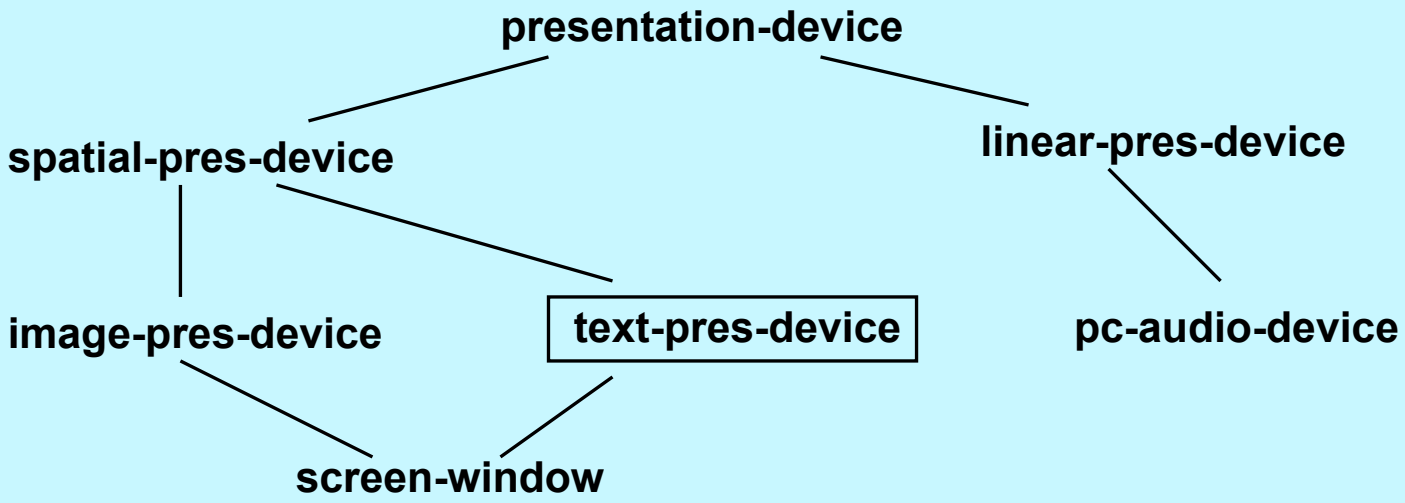
Erweiterung objektorientierter DBS (3)

- **Anwendungen**
  - definieren eigene Subklassen
  - und bekommen zahlreiche Methoden „geschenkt“
- **Systeme, Prototypen**
  - Vielfalt!
  - subtile Unterschiede in den Konzepten
  - Vereinheitlichung: ODMG, noch nicht verbreitet
  - keine Unterstützung für Zeitabhängigkeit!
- **Einsatz**
  - noch zu früh, sich mit kritischen Anwendungen auf eines der Systeme zu verlassen
  - Erfahrungen sammeln!
    Anwendungen (Programme) wegwerfen und reimplementieren

ORION als Beispiel

- **allgemein:**
  - MCC (Austin, Texas)
  - seit ca. 1985: Entwicklung eines Multimedia-DBS
  - frühe Entscheidung: muss objektorientiert sein
  - Prototyp-Implementierung in Common LISP auf Symbolics und auf SUN auf dem Markt unter dem Namen ITASCA (geringe Bedeutung)
  - voll objektorientiert
- **Multimedia Information Manager (MIM)** [Woel87b]
  - Paket von Klassen und Methoden ("Class Library" unter ORION)
  - erweiterbar: eigene Subklassen für spezielle Formate
- **Spezialität:**
  - auch Geräte (E/A und Speicher) als Objekte modelliert, genauer: Teile und Einstellungen

Ausgabegeräte



Anmerkung: umrahmte Klassen gehören nicht zum Lieferumfang, sondern stellen mögliche Benutzer-definierte Erweiterungen dar.

- **Instanzen beschreiben außer Gerät auch noch:**
  - wo dargestellt wird (z. B. Position)
  - welcher Teil eines MM-Objekts dargestellt wird
- **ein physisches Gerät also durch mehrere Instanzen dargestellt („Ausgabeformate“)**

Ausgabegeräte (2)

- **spatial-pres-device**

  Attribute: **upper-left-x**
  upper-left-y
  width
  height

  (Ausschnitt des Multimedia-Objekts)

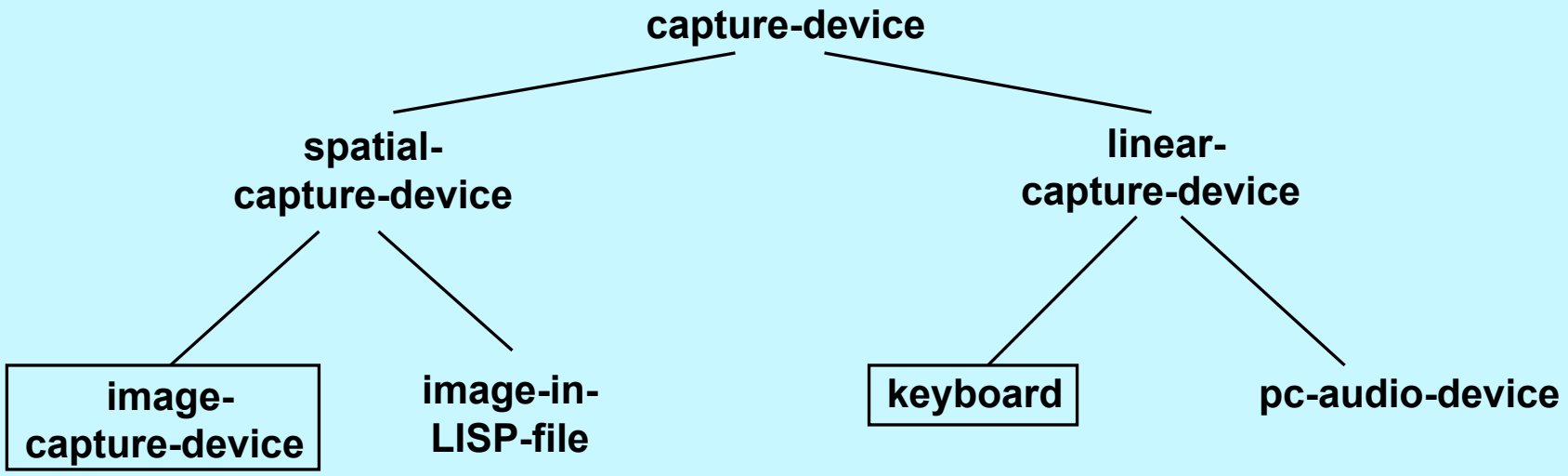
- **screen-window**

  Attribute: **win-upper-left-x**
  win-upper-left-y
  win-width
  win-height

  (Ausschnitt des Bildschirms)

  Methoden: **present**
  capture
  persistent-pres

Eingabegeräte



- **Instanzen wieder mehr als die spezifischen Geräte:**
  - welcher Teil eines Multimedia-Objekts erfasst wird
  - Einstellung des Geräts

Eingabegeräte (2)

- **spatial-capture-device**

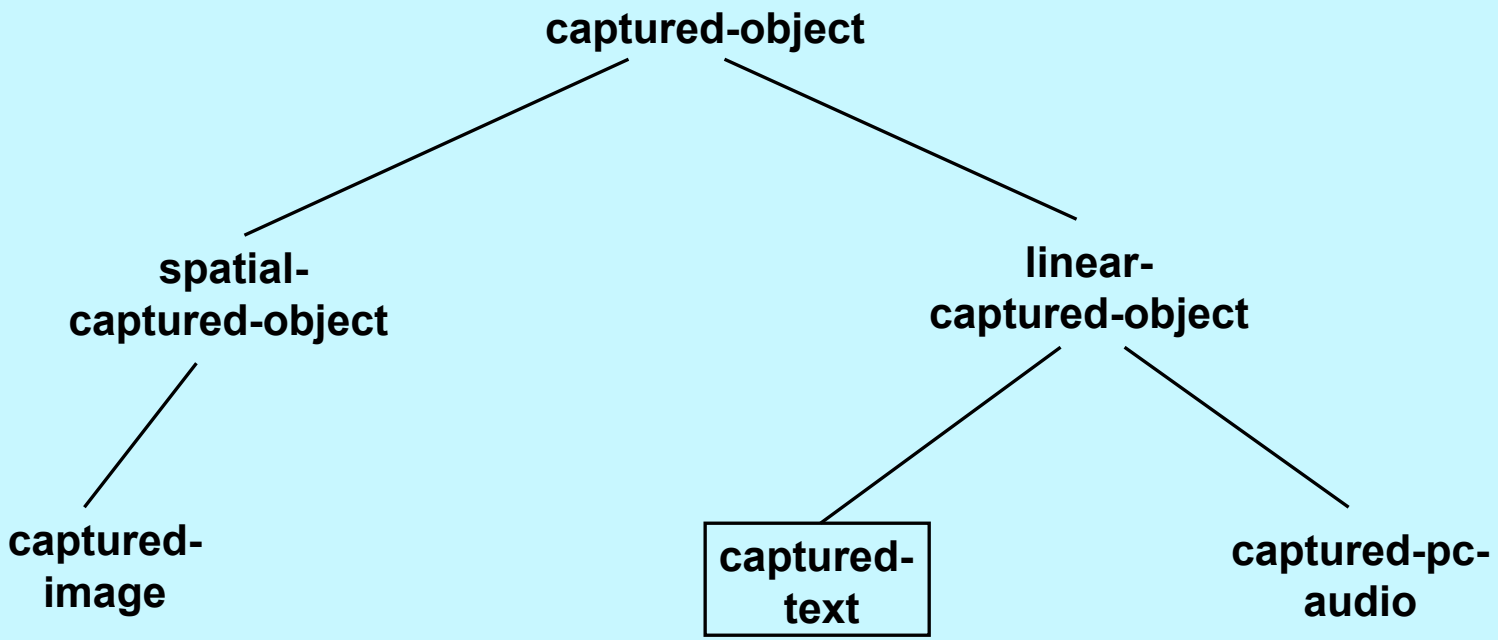
  Attribute: **upper-left-x**
  upper-left-y
  width
  height

- **image-capture-device**

  Attribute: **cam-width**
  cam-height
  bits-per-pixel

  Methoden: **capture**

Gespeicherte Objekte



Gespeicherte Objekte (2)

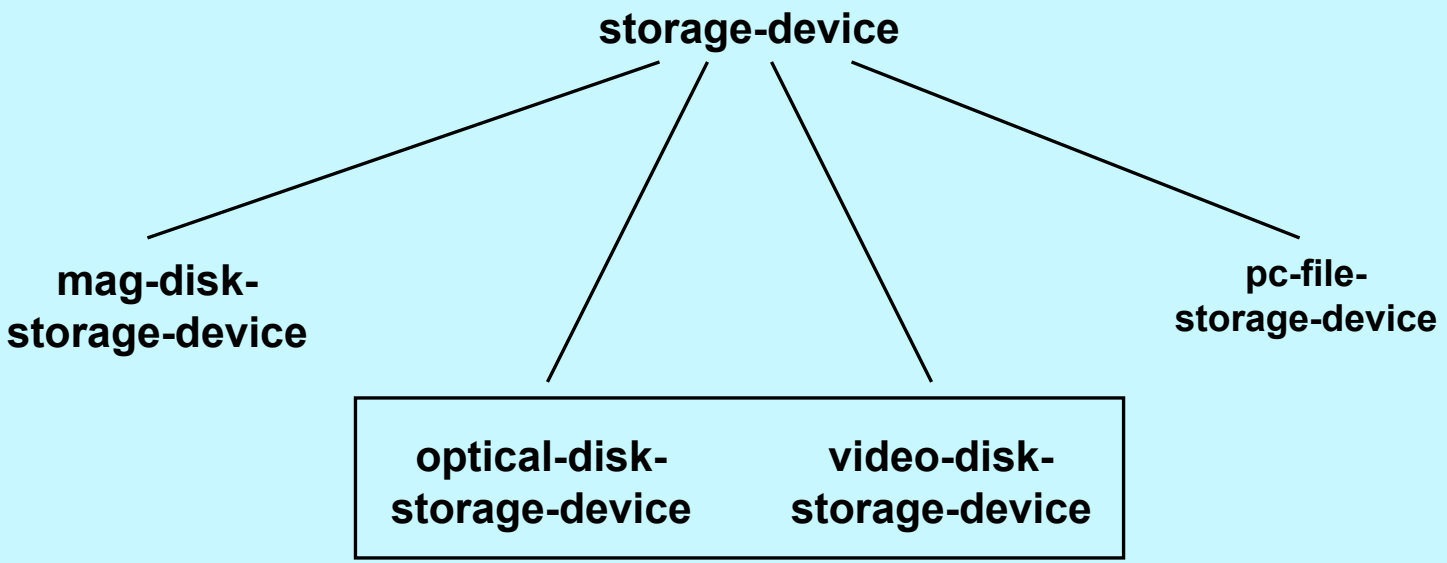
- **Attribute von captured-object:**

storage-object:	verweist auf Instanz der Klasse **storage-device** (s. unten)
logical-measure:	Einteilung der Daten aus Benutzersicht z. B. Sekunden bei Audio, Einzelbilder bei Video
phys-logic-ratio:	Bytes pro Sekunde usw.

- **Attribute von spatial-captured-object:**
  - **width**
  - **height**
  - **row-major:** Abspeicherung zeilenweise oder spaltenweise
  - **bits-per-pixel**

  (Registrierungsdaten)

Speichergeräte

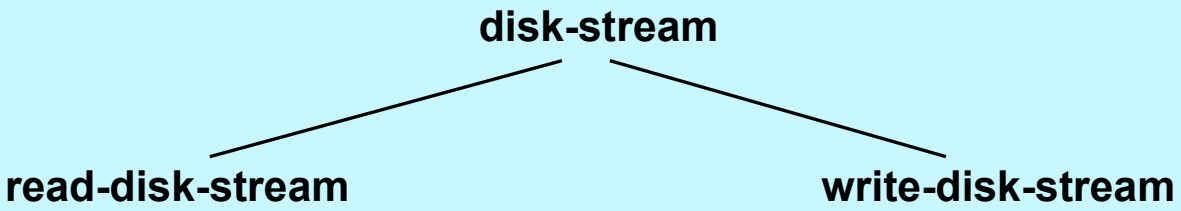


- beschreiben nur den Teil eines Speichers, der von einem Multimedia-Objekt genutzt wird

Speichergeräte (2)

- **Attribute von mag-disk-storage-device:**
  - **block-list:**
    Blocknummern aller physischen Blöcke, die das Multimedia-Objekt belegt
  - **allocated-block-list:**
    die tatsächlich angelegten Blöcke (s. Versionen)
  - **min-object-size-in-disk-pages:**
    Anzahl der Blöcke, die bei jeder Vergrößerung des Multimedia-Objekts hinzugenommen werden
  - **seg-id:**
    das Segment der Platte, in dem neue Blöcke belegt werden können

Ein- / Ausgabeströme



- **Instanzen repräsentieren einen Lese- oder Schreibvorgang; dynamisch erzeugt**
- **Attribut von disk-stream:**
  - **storage-object:** Verweis auf Instanz von **storage-device**
- **Attribut von read-disk-stream:**
  - **read-block-list:** Cursor; nächster zu lesender Block des Multimedia-Objekts

  (write-disk-stream entsprechend)

Ablauf einer Ausgabe-Operation

- **am Beispiel Rasterbild**
- **Anwendung definiert Klasse Fahrzeug mit Attributen**

  Abbildung : captured-image;

  Ausgabegerät : image-pres-device;
- **Methode Zeige-Bild:**
  - soll Bild des Fahrzeugs auf gespeichertem Gerät ausgeben; sendet Nachricht present an Gerät mit gespeichertem Bild als Parameter (in LISP-Notation!):

    (message recipient parameters)

    (**present** Ausgabegerät Abbildung)
  - durch Ausgabegerät identifizierte Instanz der Klasse image-pres-device führt Methode present aus
  - Attribute (upper-left-x, upper-left-y, width, height) spezifizieren zu zeigenden Ausschnitt des Bildes

Ablauf einer Ausgabe-Operation (2)

- Umrechnung des rechteckigen Ausschnitts in lineare Koordinaten (geht nur mit Zugriff auf das durch Abbildung identifizierte captured-image)
- dann Öffnen des gespeicherten Bildes zum Lesen:
  (**open-for-read** Abbildung [start-offset])
- durch Abbildung benannte Instanz von captured-image führt Methode open-for-read aus
- erzeugt neue Instanz von read-disk-stream und liefert deren Namen (Identifier) an image-pres-device zurück
- sendet dem Stream Nachricht zum Lesen:
  (**get-next-block** read-disk-stream)
- Stream selbst sendet:
  (**get-block** storage-object read-block-list)

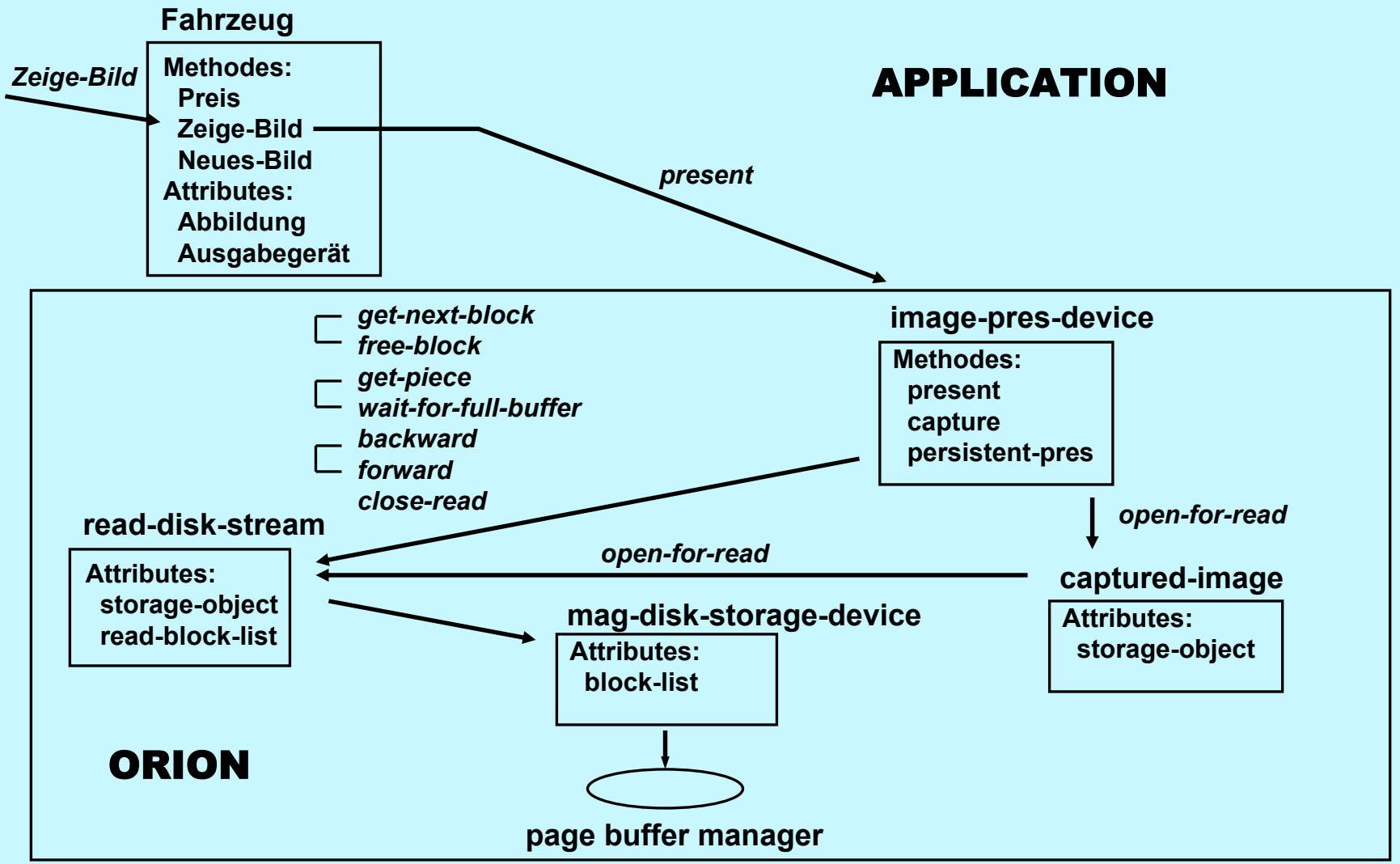
Ablauf einer Ausgabe-Operation (3)

- Rückgabewert: Adresse Seitenpuffer mit Block
- Stream erhöht Cursor und gibt Pufferadresse als Ergebnis der Methode get-next-block an das image-pres-device zurück
- Instanz von image-pres-device gibt gelesenen Block auf physischem Ausgabegerät aus und gibt Puffer wieder frei:

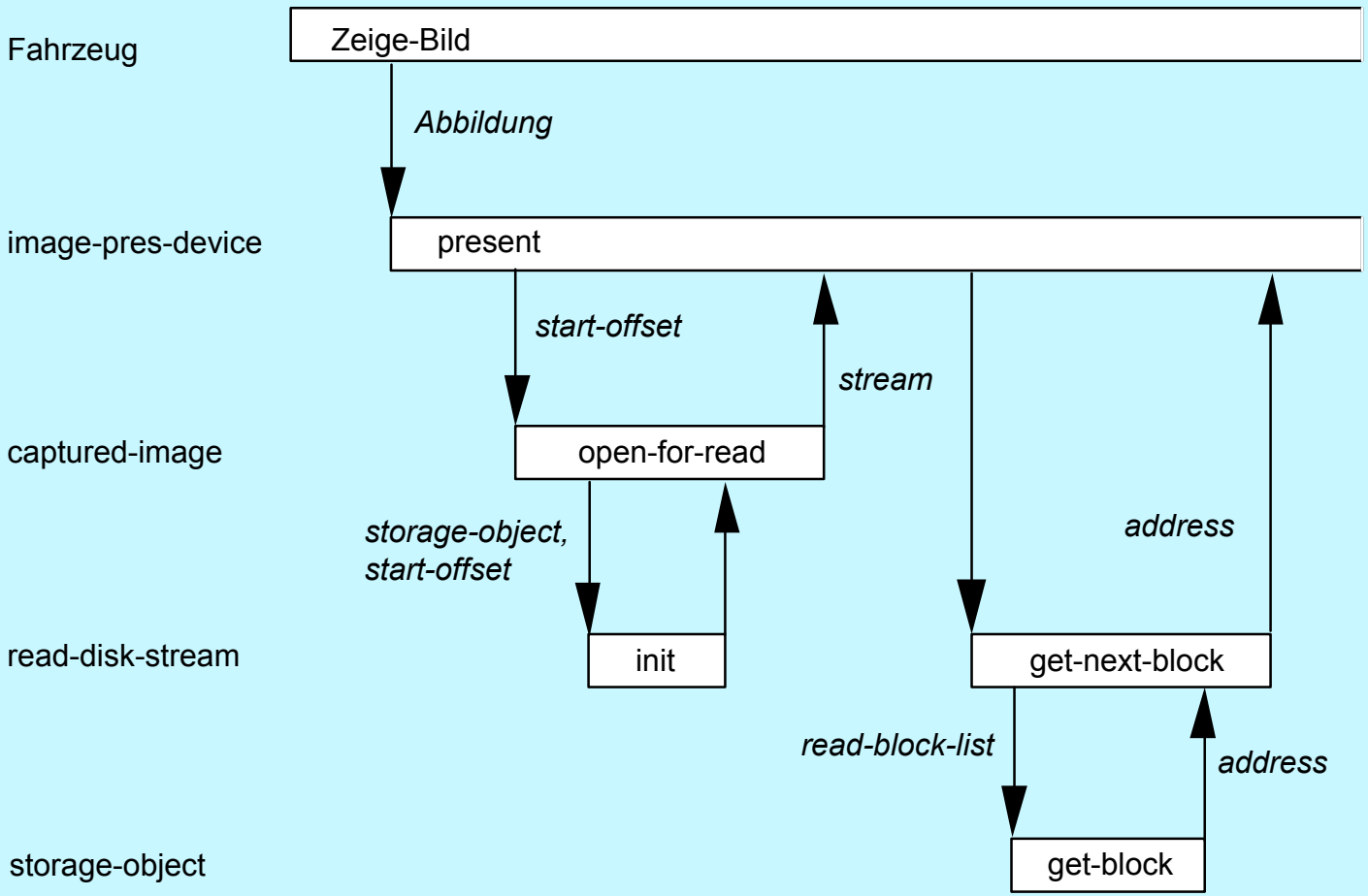
  (**free-block** read-disk-stream)
- nach Lesen aller Blöcke und Freigeben der Puffer:

  (**close-read** read-disk-stream)
- Stream wird wieder gelöscht

Ablauf einer Ausgabe-Operation (4)



Ablauf einer Ausgabe-Operation (5)



Weitere Operationen auf gespeicherten Objekten

- **captured-object:**
  - **make-captured-object-version**
    - erzeugt neue Version des captured-object wie auch seines im Attribut storage-object benannten mag-disk-storage-device
    - alte und neue Version belegen zunächst dieselben Blöcke auf der Platte (s. unten)
  - **copy-captured-object**
    - erzeugt Kopie des captured-object,
    - aber wieder Version des mag-disk-storage-device
    - d. h., Blöcke gemeinsam benutzt
  - **delete-captured-object**
  - **delete-part-of-captured-object** (start-offset, delete-count)
    - ab Byte-Position start-offset so viele Bytes löschen, wie in delete-count angegeben
    - (kann zu Inkonsistenzen führen!)

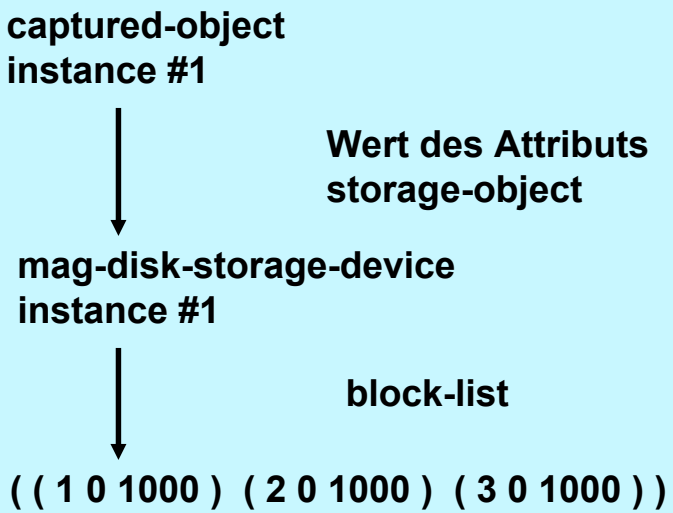
Weitere Arten der Präsentation

- **Persistente Präsentation**
  - Multimedia-Objekt bleibt nach Ausgabe noch im (System-) Speicher, so dass Anwendung es modifizieren kann
  - kann dann von dort auch wieder in DB übernommen werden:
    (**capture** presentation-device captured-object physical-resource)
  - physical-resource wird von persistent-pres zurückgeliefert

- **Steuerung der Präsentation**
  - z. B. beim Abspulen von Tonaufzeichnungen:
    Pause, Fortfahren, schneller Vor- und Rücklauf
  - umgesetzt in Nachrichten des presentation-device an den read-disk-stream:
    (**forward** read-disk-stream count)
    (**backward** read-disk-stream [count])
  - backward ohne count geht an den Anfang zurück
  - count muss in Bytes angegeben werden, Umsetzung der logischen Einheiten also vorher im presentation-device

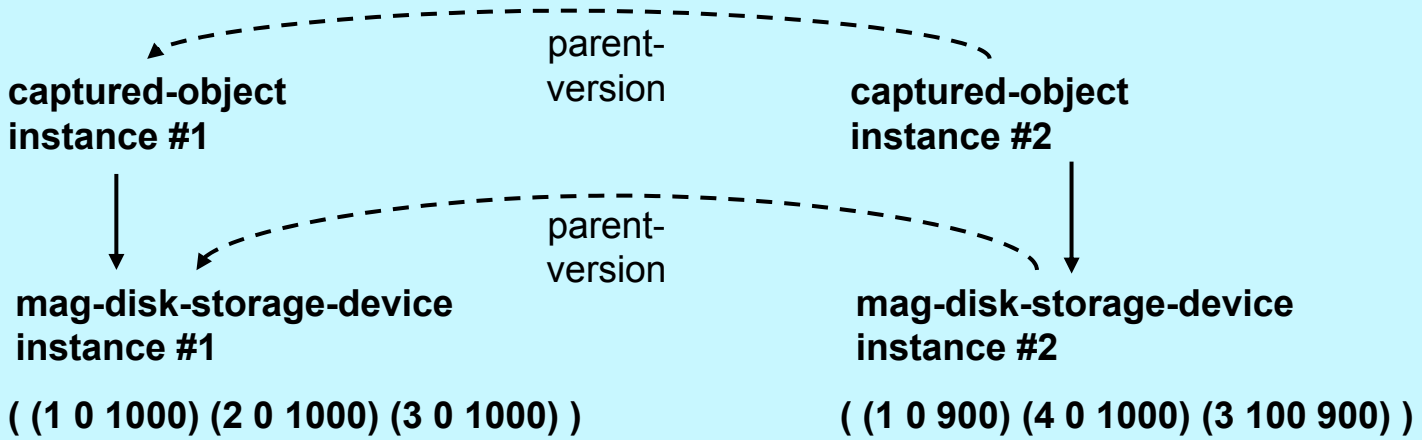
Speicherverwaltung

- **allgemein:**
  - derzeit nur für Magnetplatten (mag-disk-storage-device)
  - Prinzip: kein Überschreiben existierender Seiten auf der Platte
  - Attribut block-list (s. oben)
    enthält Liste von block-entries mit jeweils:
    (block-id start-offset length)
- **Beispiel:**



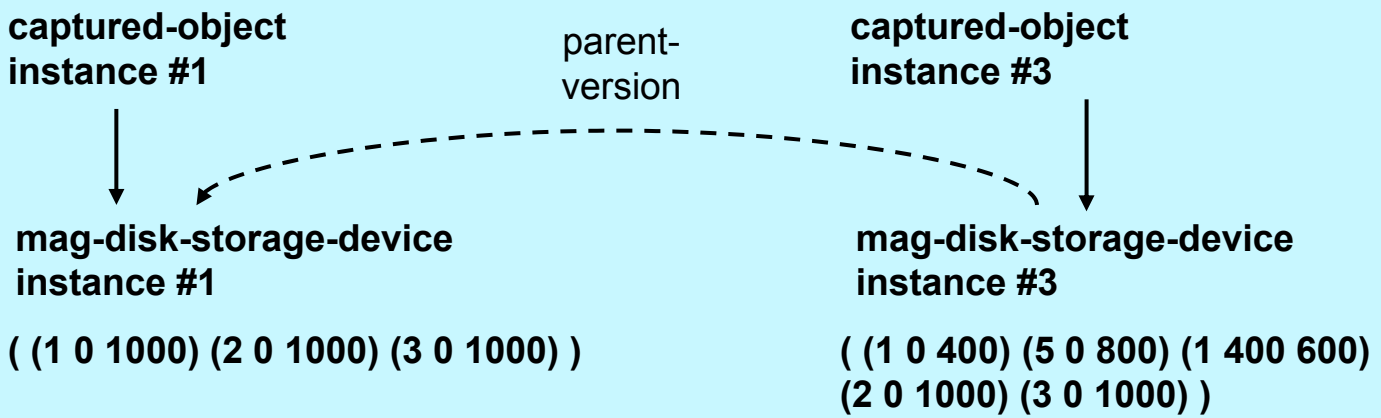
Speicherverwaltung (2)

- **neue Version erzeugen**
  - Nachricht make-captured-object-version (s. oben) an captured-object instance #1
  - Ergebnis:
    captured-object instance #2 und
    mag-disk-storage-device instance #2
- **Änderung der neuen Version:**
  - ersetze hinter den ersten 900 Byte 1200 Byte durch 1000 andere



Speicherverwaltung (3)

- **Kopie erzeugen**
  - Nachricht copy-captured-object (s. oben) an captured-object instance #1
  - Ergebnis:
    captured-object instance #3 und
    mag-disk-storage-device instance #3
- **Änderung in der Kopie:**
  - füge hinter ersten 400 Byte 800 Byte ein



Bewertung

- **umfassender Vorschlag für ein MMDBS**
- **Anwender passt System durch weitere Subklassen an seine Bedürfnisse an**
- **offene Fragen:**
  - Klassenhierarchie ädaquat? Methoden?
    - z. B. Einteilung der Multimedia-Objekte in eindimensionale (linear) und zweidimensionale (räumlich)
    - warum nicht visuell – akustisch?
    - oder zeitabhängig – statisch?
  - viele Alternativen:
    - Gerät x : zeige Bild Y
    - Bild y : stelle dich auf Gerät X dar
    - Bewertungskriterien?
  - Unterscheidung magnetische – optische Platten notwendig?
    - abstrakter: Direktzugriffspeicher, sequenzielle Speicher, Write-Once-Speicher o. ä.
  - Suche? Zeitabhängigkeit?