Prof. Dr.-Ing. Stefan Deßloch
AG Heterogene Informationssysteme
Geb. 36, Raum 329
Tel. 0631/205 3275
dessloch@informatik.uni-kl.de

Technische Universität Kaiserslautern

Multimedia-Datenbanken

Kapitel 10: (Objekt-)Relationale Multimedia-Datenbank-Verwaltungssysteme

Erweiterung des Relationenmodells

Text, Image, ... sind zugelassene atomare Domänen **(Wertebereiche), d. h. Attribute können vom Typ Text, Image, ... sein**

- Beispiele:

```
Angestellter   (Pers-Nr          integer,
                ... ,
                Passbild         image ,
                Fingerabdrücke   image)
Insasse        (Ins-Nr           integer,
                ... ,
                Vorderansicht    image,
                Seitenansicht    image)
Auto           (Hersteller       varchar(50),
                Baujahr  integer,
                ... ,
                Foto             image,
                Motorgeräusch    sound)
```

Relationenschema-Typ 1 **(1 : 1-Beziehung, Attributbeziehung)**

Relationenschema-Typ 2

- 1 : N-Beziehung
  - bei variabler Anzahl von Texten, Bildern etc. zu einem Entity: separate Relation einführen (Erste Normalform)

    Patient (Name varchar(100),
    ... ,
    Passbild image)

    Röntgenbild (Patientenname varchar(100),
    Datum date,
    Ansicht varchar(30),
    Körperteil varchar(40),
    Aufnahme **image**)
  - immer noch Attributcharakter
  - Patientenname Teil des Primärschlüssels, d. h. nur Röntgenbilder speicherbar, zu denen Patient bekannt
- Zugriff:
  - zur gemeinsamen Ausgabe von Patientendaten und Röntgenbildern: Verbund-Operation (Join)

Relationenschema-Typ 3

- N : M-Beziehung
  - Bilder können mehrere Entities zugleich zeigen: weitere Relation einführen

    Pferd (Name varchar(50),
    Alter integer)

    Rennfoto (Archivnr integer,
    Datum date,
    Ort varchar(80),
    Aufnahme **image**)

    Ist_dargestellt_auf (Pferdename varchar(50),
    Archivnr integer,
    Position varchar(10))
  - Fotos nun selbständige Entities, d. h. auch speicherbar ohne Zuordnung zu einem Gegenstand (hier: Pferd)
- Zugriff:
  - zwei (i. Allg. teure!) Verbund-Operationen

Bewertung

- Beziehungen
  - alle drei Typen (1 : 1, 1 : N, N : M) zwischen Medienobjekten und Entities darstellbar
  - allerdings ohne besondere Semantik
- Medienobjekte
  - können als Attribute oder als Entities verwaltet werden
- umständlich:
  - verschiedene Typen von Entities zu einem Medienobjekt, z. B. Schiff, Auto und Flugzeug auf einem Bild
  - mehrere Ist-dargestellt-auf-Relationen, dadurch noch mehr Joins
- stabile Umgebung:
  - relationale Datenbanksysteme verbreitet im Einsatz,
  - inkrementeller Lernaufwand für Benutzer,
  - aufwärtskompatible Ergänzung existierender Datenbanken
- in dieser Form (leidlich) realisierbar in objektrelationalen DBS (s. unten)
- geeignet zum Testen der neuen Datentypen:
  - welche Operationen braucht man wirklich?

Erweiterung der Datenbank-Anfragesprache

- hier: Benutzung von SQL (Standard)
  - Beispielrelation:

    Luftbildaufnahmen (Nr integer,
    Bild image)
  - **Eingabe**
    - vom Programm aus:

      insert into Luftbildaufnahmen
      values (:nr, **image** (:pr, :cm));
    - interaktiv (aus einer Datei):

      insert into Luftbildaufnahmen
      values (14537, **image** ("Aufnahme8.neu"));
  - Typen der angegebenen Werte müssen zu den Domains der entsprechenden Attribute passen (Type Checking zur Übersetzungszeit)

Erweiterung der DB-Abfragesprache (2)

- **Modifikation**

```
update Luftbildaufnahmen
   set Bild = Bild.replaceColormap(:cm)
   where Nr = 1286;
```

- **Suche und Ausgabe**
  - vom Programm aus:

```
select Bild.getPixrect(), Bild.getColormap() into :pr, :cm
  from Luftbildaufnahmen
  where Nr = :k;
select Bild.height(), Bild.width() into :hoehe, :breite
  from Luftbildaufnahmen
  where Bild.pixelcount (:dunkelbraun) < 1000
  and Bild.noOfColors () >= 4095;
```

  - interaktiv (direkt auf ein Gerät):

```
select Bild.display (Device ("/dev/clr02"))
  from Luftbildaufnahmen
  where Nr = 715;
```


Probleme mit der SQL-Einbettung

- nicht unterstützt:
  - wiederholter Zugriff auf dasselbe Attribut
    (erst Höhe und Breite des Bildes, dann Farbtabelle, ...)
- nicht erlaubt:
  - Kombination von values und subselect im insert
    (Ausschnitt eines Bildes woanders speichern)
- update
  - Semantik des update ist **Wertersetzung**, nicht Modifikation
  - besser wäre etwas wie:

```
update Luftbildaufnahmen
  apply Bild.replaceColormap ( … )
```

- Programmierspracheneinbettung als Anweisung
  - nicht als funktionaler Ausdruck:

```
var := select … from … where … ;
writeScreen (select Bild.getPixrect () … );
```

Objektrelationale DBS

- bieten im Strukturteil:
  - Typen, Typkonstruktoren, ADTs
  - Objektidentitäten für komplexe Tupel in Relationen (Komponenten über Identität dargestellt)
  - Klassen- und Typhierarchie (getrennt), Klassen entsprechen Relationen (Tabellen)
  - Vererbung und evtl. auch Overriding
- und im Operationenteil:
  - generische, relationale Operationen (SQL)
  - Methoden
- Beispiele:
  - am Anfang: (University-) Ingres, 1984
  - dann das kommerzielle Ingres und Postgres
  - UniSQL
  - Illustra und Informix Universal Server
  - DB2
  - Oracle 8

SQL:1999

ISO- und ANSI-Norm,
Weiterentwicklung von SQL-89 und SQL-92
Merkmale:

- Abstrakte Datentypen, mit create type definiert
- Objekt-Identifikatoren für einige ADTs („Objekt-ADTs“) neben Tupel-Identifikatoren für Tupel in Tabellen
- ADT-Hierarchien (ähnlich den Typhierarchien), die Substituierbarkeit von Objekten zusichern
- Tabellenhierarchien (ähnlich der Inklusion von Extensionen zwischen Unter- und Oberklassen, hier bezogen auf Tabellen); alle Attributnamen und Schlüssel werden geerbt, dürfen aber auch verändert werden
- Definition von Funktionen für ADTs
- Überladen des Funktionsnamens mit Möglichkeiten zur dynamischen Auswahl der Implementierung (ähnlich dem Overriding)
- komplexe Datentypen wie Arrays und Tupel
  (Mengen bzw. Multimengen in SQL:2003 eingeführt)

SQL:1999 (2)

- Tupel und Tabelle spielen unverändert Sonderrolle:
  - Persistenz an das Einfügen von Tupeln in Tabellen gebunden, andere Objekte oder Werte nicht persistent
  - Anfragen nur an Tabellen möglich, ergeben auch wieder Tabellen
- Abstrakte Datentypen
  - für Objekte (Objekt-ADT, mit Identität) oder für Werte (Wert-ADT)
  - Attribute und Funktionen:

```
create type Student
        under Person
        as (
                MatrNr integer,
                Studienfach varchar(30),
                …
        )
        instance method Durchschnittsnote ()
                returns real
                language SQL
                deterministic
                contains SQL
…;
```

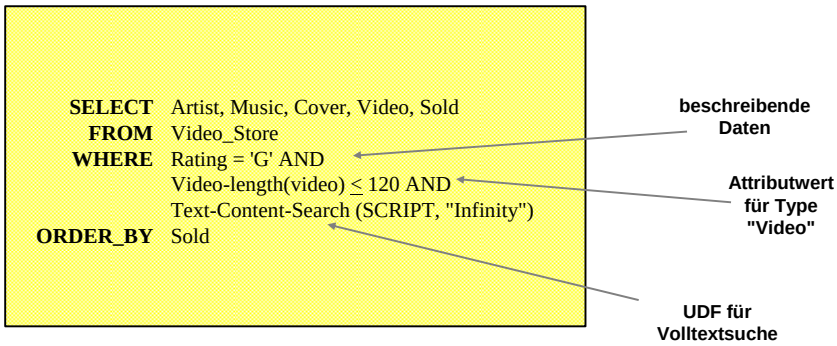

SQL:1999 (3)

  - zu jedem Attribut eines ADTs zwei Funktionen automatisch generiert:
    - Observer: lesender Zugriff, Anfrage
    - Mutator: Ersetzung des Attributwerts
  - ebenfalls equals als Vergleichsoperation zu jedem ADT
  - Konstruktor mit Namen des ADTs
- Einkapselung:
  - **keine** Sichtbarkeitsstufen wie public, protected, private
  - bei Objekt-ADTs kann automatisch gekapseltes und nicht änderbares Surrogat-Attribut erzeugt werden (self-referencing column)
  - "equals oid" erlaubt dann auch Test auf Identität, im Gegensatz zur normalen Zustandsgleichheit ("equals state")

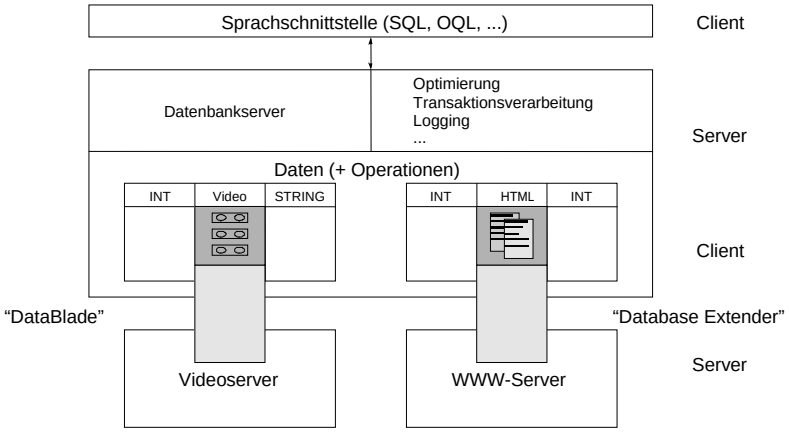
SQL:1999 (4)

- Unterscheidung von
  - Funktionen, Methoden (Rückgabewert, Anfrage) und
  - Prozeduren (kein Rückgabewert, Änderung)
- Parameter
  - von Prozeduren können mit in, out und inout gekennzeichnet werden
  - bei Funktionen nur in
- Funktionsdeklaration
  - mit SQL:1999 selbst beschreiben,
  - als "stored procedure" im System selbst abgelegt

Mischung traditioneller relationaler Daten mit MM-Daten: Anfrage



Objektrelationale Datenbanksysteme

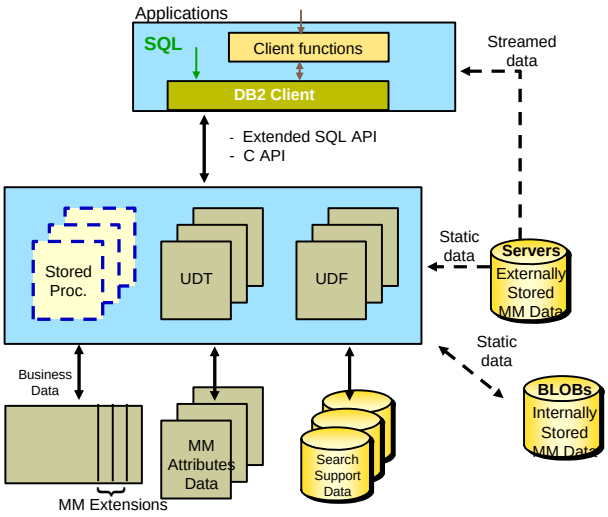


Beispiele: objekt-relationale Datenbanksysteme DB2, Oracle

Speicherung und Indexierung

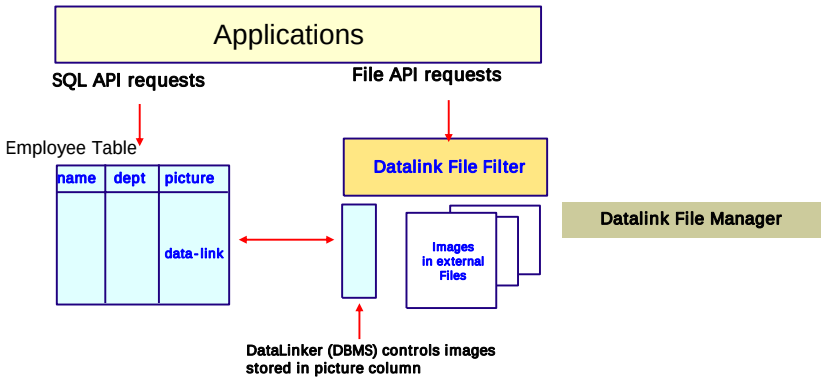
- Speicherung
  - Multimedia-Dokument ablegen in externer Datei, Byte-String oder anderer DBMS-Dateistruktur
  - sinnvoll bei Geo-Daten und anderen Bilddaten (etwa R-Baum)
- Indexierung
  - statt Multimedia-Dokument nun abgeleitete Informationen über das Dokument im DBMS ablegen
  - sinnvoll bei allen MM-Daten (etwa R-Baum, X-Baum, LSD-Baum, .. Andere Indexstrukturen für hochdimensionale Daten)
- Verbreiteter Fall
  - Multimedia-Inhalt in Byte-String (BLOB) oder externe Datei, unter Kontrolle des DBMS (Management of External Data)
  - Indexstruktur im DBMS

Architektur



Integration großer Datenbestände mittels DataLinks

- ... oder als Datalinks
  - Referentielle Integrität
  - Zugriffskontrolle
  - Konsistentes Backup und Recovery
  - Transaktionssicherheit



DataLinks in SQL/MED

- Ziel
  - externe Speicherung, Manipulation von Dateien beibehalten
  - Integritätskontrolle, Recovery, Zugriffskontrolle von SQL-Daten auf Dateien ausdehnen
- Konzepte
  - datalink ist eine Instanz des DATALINK Datentyps
    - Dateireferenz (URL), verweist of Datei die von einem externen File Manager verwaltet wird
  - datalink Optionen (pro Column)
    - definieren welche Kontrollmöglichkeiten der SQL Server über die Dateien haben soll
      - integrity, read/write access, recovery
    - spezifiziert die Semantik der link/unlink-Operationen
  - datalinker (Systemkomponente)
    - implementierungsabhängig
    - realisiert die Mechanismen zur Garantie der Datalink-Eigenschaften, Integritätskontrolle, Recovery, Zugriffskontrolle

Funktionen und Operationen

- Neue SQL-Funktionen für datalinks
  - Konstruktor: DLVALUE, ...
  - (Komponenten von) URLs: DLURLCOMPLETE, …
- SQL statements (Beispiele)
  - insert (“link”)

```
INSERT INTO Movies (Title, Minutes, Movie)
    VALUES ('My Life', 126,
    DLVALUE('http://my.server.de/movies/mylife.avi'))
```

  - select (inkl. URL access token)

```
SELECT Title, DLURLCOMPLETE(Movie)
    FROM Movies
    WHERE Title LIKE '%Life%'
```

Data Link Options

- Link control (NO, FILE)
  - NO LINK CONTROL
    - URL-Format des datalink
    - keine weitere Kontrolle, Datei ist nicht "linked"
  - FILE LINK CONTROL
    - Datei ist "linked", muss existieren!
    - Grad der Kontrolle über weitere Optionen definiert
- Integrity control option (ALL, SELECTIVE, NONE)
  - INTEGRITY ALL
    - Datei kann nicht gelöscht oder umbenannt werden
  - INTEGRITY SELECTIVE
    - Datei kann nur über den File Manager gelöscht oder umbenannt werden, falls kein Data Linker installiert ist
  - INTEGRITY NONE
    - Datei kann über den File Manager gelöscht oder umbenannt werden
      - nich kompatibel mit FILE LINK CONTROL

Data Link Options (2)

- Read permission option (FS, DB)
  - READ PERMISSION FS
    - File Manager kontrolliert Lesezugriff
  - READ PERMISSION DB
    - SQL Server kontrolliert Lesezugriff, basierend auf den Leseprivilegien der Datalink Column
    - Einsatz von "read access tokens"
      - wird vom SQL Server in die URL einkodiert
      - durch externen file manager/data linker verifiziert
- Write permission option(FS, ADMIN, BLOCKED)
  - WRITE PERMISSION FS
    - File Manager kontrolliert Schreibzugriff
  - WRITE PERMISSION BLOCKED
    - Schreibzugriff nicht erlaubt
  - WRITE PERMISSION ADMIN [NOT] REQUIRING TOKEN FOR UPDATE
    - Schreibzugriff von SQL Server (und Datalinker) gesteuert
      - benötigt READ PERMISSION DB
    - Einsatz von "write access tokens" für Änderungen am Dateiinhalt
      - muss evtl. dem SQL Server erneut "vorgelegt" werden

Funktionen und Operationen (2)

- "Update-in-place"

  SELECT Title, **DLURLCOMPLETEWRITE**(Movie)
  INTO :t, :url ...

  open using URL, modify ...

  UPDATE Movies SET Movie = **DLNEWCOPY**(:url, 1)
  WHERE Title = :t

- DLNEWCOPY
  - teilt dem SQL Server eine Änderung des Dateiinhalts mit
  - Alternative: DLPREVIOUSCOPY – Anwendung ist nicht an evtl. vorgenommenen Änderungen interessiert, alter Zustand soll wiederhergestellt werden

Data Link Options (3)

- RECOVERY YES/NO
  - gibt an, ob SQL Server (gemeinsam mit Datalinker) Recoverymaßnahmen ermöglichen und koordinieren soll
- Unlink option (RESTORE, DELETE, NONE)
  - ON UNLINK RESTORE
    - ursprüngliche Eigenschaften (ownership, permissions) wiederherstellen
  - ON UNLINK DELETE
    - Datei löschen
  - ON UNLINK NONE
    - ownership und permissions nicht wiederhergestellt
- SQL statement (Beispiel)
  - "Unlink/Replace"

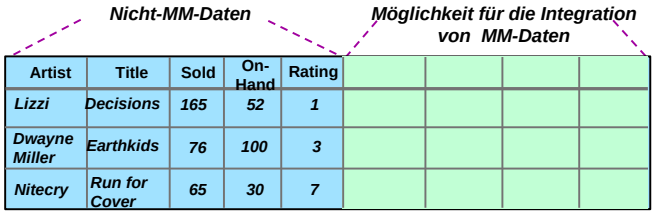
    UPDATE Movies SET Movie =
    DLVALUE('http://my.newserver.de/mylife.avi')
    WHERE Title = 'My Life'

    RESTORE oder DELETE für ".../movies/mylife.avi"

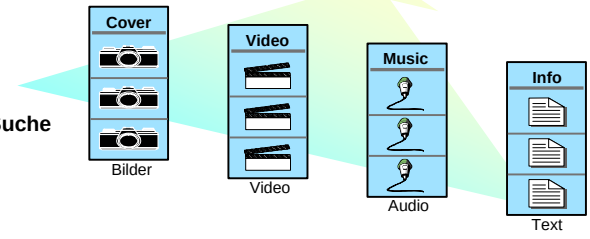
Kommerzielle Systeme

- Database Extenders (IBM DB2)
  - Datentypen und Funktionen
    - Text Extender
    - Image, Audio, Video Extender
    - Spatial Extender
- Data Blades (Illustra, Informix Universal Server)
  - Sammlungen von Datentypen und zugehörigen Funktionen
    - Text, Spatial, Image, Visual Information Retrieval (Virage)
    - weitere für WWW, Statistik, Zeitreihen u. a.
  - Speicherung in der DB (BLOB) oder in separater Datei
  - zugeschnittene Indexstrukturen (z. B. R-Baum)
  - von Fremdfirmen erstellt, von Informix zertifiziert
- Data Cartridges (Oracle)
  - ConText, Virage, usw. wie bei den anderen
- Stand:
  - Experimentierstadium, noch keine Einheitlichkeit, Norm noch nicht umgesetzt
  - keine Echtzeit!

DB2 Extender - Applikationssicht



- **Komplexe Datentypen**
- **Neue Funktionen**
- **Integrierte, innovative Suche**
- **Offenheit**



Erweitertes Datenmodell (Beispiel Image Extender)