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 9: Datenmodelle für Multimedia

Multimedia-Datenbanken - Teilprobleme

- zur Erinnerung: Speichern und Wiedergewinnen
  - Ziele dabei:
    - Geräte- und Formatunabhängigkeit
    - Beziehungen (für navigierenden Zugriff)
    - inhaltsorientierte Suche
    - Echtzeit
- zu lösende Teilprobleme (nach [Chri85b]):
  1. Definition der Datentypen mit Operationen
     - abstraktes Modell / Schema der Medienobjekte
     - Zugriff, Änderung
     - Extrahieren von Information (in formatierte Daten)
     - Transformationen (Medien-Umsetzung)
  2. Inhaltsadressierung
     - indirekte Suche (in zugeordnetem Text)
     - Ähnlichkeit statt Gleichheit
     - räumliche Beziehungen
     - automatische Inhaltserschließung?
     - Vergleichsoperationen

Teilprobleme

3. Techniken des Information Retrieval
   - Integration mit DB-Techniken
4. Mehrbenutzerbetrieb, Recovery, Zugriffskontrolle, Unterstützung von Versionen
   - Granulate
5. Speichergeräte mit großer Kapazität
   - optische Platten
   - Überwachung von Kopien, Komprimierung
6. Performance (Leistungsfähigkeit)
   - Zugriffsmethoden für die neuen Datentypen
   - Echtzeit
   - Hardware-Architektur, physischer DB-Entwurf, Optimierung von Anfragen
7. System-Architektur
   - Erweiterung existierender DBVS
   - separate Spezial-DBVS mit gemeinsamer Benutzerschnittstelle
   - komplett neues DBVS
8. Einsetzbare Prototypen
   - Erfahrungen sammeln

Multimedia-Datentypen

1. Einführung neuer (Basis-) Datentypen:
   TEXT, GRAPHICS, IMAGE, SOUND, VIDEO
   mit darauf anwendbaren Operationen
   (→ Abstrakte Datentypen)

2. Einbettung in existierende Datenmodelle
   - Relationenmodell
   - Objekt-relationales Modell
   - objektorientiertes Modell

Nutzung der verfügbaren Modellierungskonstrukte
und Anfragesprachen

Basisdatentypen

integer
- Operationen:
  + , – , * , / , ... : integer × integer → integer
  = , ≠ , ≤ , ≥ , ... : integer × integer → boolean

real / float
- Operationen:
  analog zu integer

char
- Operationen:
  Umsetzung in integer und umgekehrt, Ausgabe (drucken), ...

boolean / bit
- Operationen:
  and, or, ...

d. h. Typen bestimmt durch ihre Operationen!

Typkonstruktoren

(„generische“ oder „parametrisierbare“ Typen)

- listOf *Typ (min, max)*
  - Operationen:
    Länge feststellen, Zugriff auf einzelne Elemente, Konkatenation, Teilliste, reduce wie in LISP, ...
  - Beispiele:
    byte = listOf boolean (8,8)
    string = listOf char (0,*)
  - kanonische Fortsetzung aller Operationen auf dem Elementtyp:
    Liste3 := Liste1 * Liste2
    elementweise ausgeführt
- setOf *Typ (min, max)*
  - Operationen:
    Anzahl Elemente, for each, Vereinigung, Differenz, Element hinzufügen oder entfernen, ...

Der Datentyp Text

- Was ist das abstrakte Modell von Text?
  - nur das, was „gleichen“ Texten gemeinsam ist – darstellungsunabhängig!
  - mehr als nur listOf char!
- anwendbare Funktionen (in Java-Notation):
  - lesender Zugriff:

```
interface Text {
  public int length ();
  public int alphabet ();  // 0 == ISO Latin-1, …
  public int alphabetSize ();
  public int language ();  // 0 == English, 1 == German, …
  public char charAt (int n);
  public byte [ ] getASCII ();
  public byte [ ] getEBCDIC ();
  public String getUnicode ();
  … }
```


Der Datentyp Text (2)

  - mit Worttrenner (Leerzeichen, „White Space“) und Zeilenende:

```
public byte [ ] word (int wordNo);
public byte [ ] line (int lineNo);
public int wordCount ();
public int lineCount ();
```

  - ganzheitlich, z. B. Anzeigen und Ausdrucken:

```
public boolean print (Printer p);
public boolean display (Window w);
```

  - ändernder Zugriff (mit Konsistenzerhaltung!):

```
public void replaceLine (int lineNo, byte [ ] newLine);
public void insertLine (int lineNo, byte [ ] newLine);
public void concatenate (Text t2);
```

- generelles Problem: Prozedur oder Funktion?
  - Prozedur ändert direkt (Beispiele oben)
  - Funktion erzeugt neues Objekt:

```
public Text replaceLine (int lineNo, byte [] newLine);
```

  - im Kontext von SQL (siehe unten) derzeit nur Funktionen nutzbar

Der Datentyp Text (3)

- Erzeugen:

```
class TextClass implements Text {
  public TextClass (
            int length,
            int charLength,
            int code,  // 0 == ASCII, 1 == EBCDIC, …
            int formatter,  // 0 == none, 1 == PostScript, …
            byte endOfLine,
            byte [ ] characters
  ) { … };
  …
}
```

oder in einem spezifischen Kontext (Unix) auch:

```
  public TextClass (String filename) { … };
```


Der Datentyp Image

- lesender Zugriff :

```
interface Image {
  public int height ();
  public int width ();
  public int pixelcount (byte [] pixelvalue);
  public Pixrect getPixrect ();
  public boolean display (Device d);
  …
}
```

- ändernder Zugriff:

```
  public Image window (int x0, int y0, int x1, int y1); // (crop-Semantik)
  public Image replaceColormap ( … );
  public Image replacePixelvalue ( … );
```

Der Datentyp Image (2)

- Erzeugen:

```
class ImageClass implements Image {
  public ImageClass (
            int height, int width, int depth,
            float aspectRatio,
            Code encoding,
            int colormapLength, int colormapDepth,
            int [ ] [ ] colormap,
            byte [ ] pixelmatrix
  );
  …
}
```

in einem spezifischen Kontext (SUN) auch :

```
  public ImageClass (Pixrect pr, Colormap cm);
```

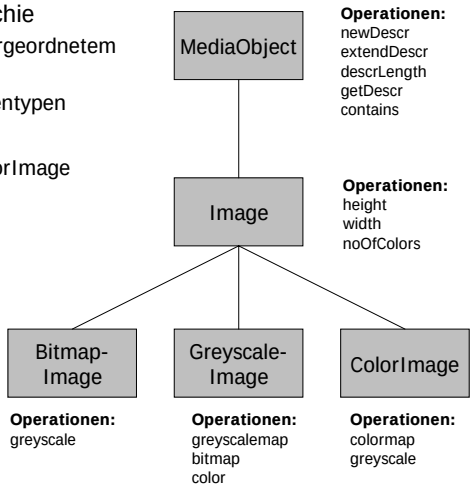

Inhaltsangaben und Vergleiche

- Inhaltsangaben
  - abhängige Komponenten, mit dem Medienobjekt gekapselt
  - erfordert zusätzliche Operationen
    - hinzufügen, erweitern von Inhaltsangaben
    - abfragen der Länge und des Inhalts
    - prüfen, ob eine Suchanfrage zur Inhaltsangabe "passt"
  - Beispiel: Image

```
interface Image {
  …
  public void newDescr (String descr);
  public void extendDescr (String descr);
  public int descrLength ();
  public String getDescr ();
  public boolean contains (String query);
}
```

Generalisierung

- Aufbauen einer Generalisierungshierarchie
  - Generische Operationen werden in übergeordnetem MediaObject-Datentyp definiert
  - Möglichkeit der Verfeinerung von Medientypen
- Beispiel: Image
  - Verfeinerung in Bitmap, Greyscale, ColorImage
  - Operationen zur Umwandlung
- zusätzliche Aspekte
  - Disjunktheit, Vollständigkeit
    - erfordert Konstrukturen für Subtypen
- alternative, anwendungsorientierte Verfeinerungen denkbar



SQL/MM – Ein SQL-Standard für Medienobjekte

- Verwendung von Mediendaten und –operationen in vielen Anwendungen
- Ausnutzung der Erweiterbarkeit von ORDBMS für die Definition von Medienobjekten
- Bereitstellung von Erweiterungen in Paketen
  - erleichtert Verwaltung (Installation, Upgrade, Entfernen) und Wiederverwendung (ein Paket kann anderes benutzen)
- Proprietäre Pakete existieren bereits für
  - Informix: Excalibur Text Search DataBlade, Excalibur Image DataBlade, Informix Video Foundation Data-Blade Module
  - DB2: Image Extender, Audio Extender, Video Extender, Text Extender
  - Oracle: Visual Information Retrieval (VIR) Cartridge, ConText Cartridge, InterMedia
- Standardisierung erlaubt:
  - gemeinsame „Sprache“
  - Datenaustausch
  - Anwendungen laufen auf verschiedenen Implementierungen

(O)RDBMS als Basis

- Nachfolger der RDBMS
- standardisiert in SQL:1999 (SQL:2003)
- erweiterbar um
  - Datentypen
    (User Defined Types, UDT)
  - dazugehörige Funktionen
    (User Defined Functions, UDF),
    Operatoren und Methoden
- UDT's können verwendet werden als
  - Typ einer Tabellenspalte
  - Typ eines Attributs in einem anderen UDT
  - Obertyp für einen abgeleiteten UDT

Überblick SQL/MM

- gehört zum SQL-Standard, ist aber eigenständig
  - SQL: ISO/IEC 9075, SQL/MM: ISO/IEC 13249
  - Voller Name: *SQL Multimedia and Application Packages*
- besteht aus mehreren Teilen
  - Teil 1: SQL/MM **Framework** (2000)
  - Teil 2: SQL/MM **Full Text** (2000)
  - Teil 3: SQL/MM **Spatial** (2000)
  - Teil 5: SQL/MM **Still Image** (2001)
  - ...
- Teil 1 gibt Überblick und spezifiziert Konformität
- jeder weitere Teil
  - ist ein Paket für eine Art von Mediendaten
  - besteht aus UDT's, Methoden und Funktionen
    gemäß SQL:1999

SQL/MM Full Text

- Version vom 10.12.2001
- spezifiziert
  - UDT **FullText** für Text-Daten und
  - UDT **FT_Pattern** für Suchmuster
- FullText:
  - vier Suchmethoden;
    - zwei unterscheiden sich jeweils nur im Parameter:
      Zeichenkette oder Muster vom Typ FT_Pattern (Overloading)
    - Contains-Methoden: Boolesche Suche ⇒ Ergebnis: ja/nein
    - Rank-Methoden: Ranking ⇒ Ergebnis: impl.-abh. Real-Wert
  - zwei Konstruktoren (Zeichenkette, Zeichenkette + Sprache)
  - Funktion FullText_to_Character zum Erzeugen einer Zeichenkette

UDT-Definitionen

```
create type FullText as (
   Contents character varying(FT_MaxTextLength),
   Language character varying(FT_MaxLanguageLength),
   …
   )
method Contains (pattern FT_Pattern) returns integer
method Contains (
   pattern character varying(FT_MaxPatternLength)
   ) returns integer
method Rank (pattern FT_Pattern)
   returns double precision
method Rank …
```

UDT-Definitionen (2)

```
method FullText (
   String character varying(FT_MaxTextLength)
   ) returns FullText
method FullText (
   String ... ,
   Language character varying(FT_MaxLanguageLength)
   ) returns FullText;
create cast (FullText as
   character varying(FT_MaxTextLength)
   with FullText_to_Character);
create type FT_Pattern as
   character varying(FT_MaxPatternLength);
```

- Werte von FT_Pattern müssen Ausdrücke einer in BNF beschriebenen Sprache sein
- Auswertung durch Regeln über Symbolen der Sprache beschrieben

Suchmuster für Contains und Rank

- Textbeispiel

  aText: „In diesem Abschnitt wird der Standard SQL/MM vorgestellt. Dieser Standard definiert Typen und Routinen für Medienobjekte.“
- einzelnes Wort

  ```
  aText.Contains (' "Abschnitt" ') = 1
  ```
- Menge von Worten
  - Wildcards

    ```
    aText.Contains (' "Abschnitt_" ') = 0
    ```
  - Erweiterungsmuster (ähnliche Worte, allgemeinere W., speziellere W., Synonyme, Abstammung)

    ```
    aText.Contains ('
       thesaurus "Informatik"
       expand synonym term of "Norm"
    ') = 1
    ```

Suchmuster für Contains und Rank (2)

- Kontextmuster

```
aText.Contains ('
   ("Abschnitt") near "Standard" within 0 sentences in order
') = 1
```

- Konzeptmuster

```
aText.Contains ('
   is about "Internationaler Standard zur Volltextsuche"
') = 1
```

  - einzelne Phrase, Aufzählung von Einzelwortmustern, Mengen von Phrasen, Muster mit Booleschen Operatoren (|, &, NOT)

- Beispielanfrage:

```
select * from myDocs
  where Doc.Rank(' "Standard" ') > 0.8
```


Berücksichtigung der Sprache

- zu jedem Text kann eine Sprache angegeben werden (siehe Definition)
- zu einigen Mustern kann ebenfalls eine Sprache angegeben werden
- wozu?
  - Erkennung von Wort-, Satz- und Absatzgrenzen
  - richtige Expansion, z. B. für ähnliche Worte
  - Behandlung von Stoppworten (engl. 'die' vs. dt. 'die')
  - **Textbeispiel** wieder aText: „In diesem Abschnitt wird der Standard SQL/MM vorgestellt. Dieser Standard definiert Typen und Routinen für Medienobjekte.“

    ```
    aText.Contains (' ("Typen oder Routinen") ') = 1
    ```

  - Wortnormalisierung:
    „Müller“ wird ersetzt durch „Mueller“

SQL/MM Spatial

- Version vom 10.12.2001 (581 Seiten)
- entspricht dem Typ Graphik
- spezifiziert UDT's für
  - 2D-Daten (Punkt, Linie, Fläche)
  - Kollektionen davon
- definiert Routinen für
  - Manipulation, Suche und Vergleich von räumlichen Daten
  - Konvertieren zwischen den UDT's und Zeichen- oder Binärdarstellungen
- zu jedem Geometrieobjekt (ST_Geometry) gehört ein **SRID** (spatial reference system identifier), der das räumliche Referenzsystem spezifiziert
  - beruht auf bekannten Referenzsystemen
    - geographisches Koordinatensystem: Länge, Breite
    - Projektionskoordinatensystem: X, Y
    - geozentrisches Koordinatensystem: X, Y, Z
  - ein Referenzsystem
    - für Elemente einer Kollektion vom Typ ST_Geometry
    - innerhalb einer Spalte vom Typ ST_Geometry

SQL/MM Spatial: Typen

- 0-dim: ST_Point
- 1-dim: ST_Curve
  - Subtypen unterscheiden sich in der Interpolation zwischen den Einzelpunkten
  - **ST_LineString**: lineare Interpolation
  - **ST_CircularString**: kreisförmige Interpolation
  - **ST_CompoundString**: gemischt
- 2-dim: ST_Surface
  - **ST_CurvePolygon**: 1 externe + n interne ST_Compound-String-Umrandungen
  - **ST_Polygon**: nur ST_LineString-Umrandungen
- Kollektionsobjekte
  - gleiches Referenzsystem für alle Elemente
  - **ST_MultiPoint**
  - **ST_MultiCurve, ST_MultiLineString**
  - **ST_MultiSurface, ST_MultiPolygon**

SQL/MM Spatial

- ST_Geometry-Methoden:
  - Durchschnitt (Punktmengendurchschnitt), Differenz, Vereinigung
  - Abstand
  - Tests (contains, overlaps, touches, crosses, …)
  - Ermitteln des Referenzsystems
- weitere Methoden auf Subtypen
  - ST_Curve: length
  - ST_Surface: area, perimeter

SQL/MM Still Image

- Version vom 10.12.2001
- spezifiziert
  - UDT **SI_StillImage** für Bilddaten,
  - UDT **SI_Feature** für Merkmale und
  - UDT **SI_FeatureList** für Listen von Merkmalen
- SI_StillImage:
  - interne Repräsentation offen gelegt (⇨ keine Datenunabhängigkeit)
  - zwei Konstruktoren (BLOB, BLOB + Format)
  - zwei Mutator-Methoden: BLOB-Ersetzung + Formatänderung
  - zwei Observer zur Erzeugung von Miniaturen („Thumbnails“)

SQL/MM Still Image: UDT SI_StillImage

```
create type SI_StillImage as (
   SI_content binary large object(SI_MaxContLength),
   SI_contentLength integer,
   SI_format character varying(8),
   SI_height integer,
   SI_width integer,
   …
   )
```

- SI_content:
  - umfasst auch Registrierungsdaten (Header, Farbtabellen usw.)
  - „Container“ für das ganze Bild
- SI_format:
  - unterstützte Formate (das DBS kann sie lesen und Bildeigenschaften extrahieren)
  - benutzerdefinierte Formate

SQL/MM Still Image: UDT SI_StillImage (2)

```
method SI_StillImage (
   content binary large object(SI_MaxContLength)
   ) returns SI_StillImage
method SI_StillImage (
   content binary large object(SI_MaxContLength),
   format character varying(…)
   ) returns SI_StillImage
method SI_setContent (
   content binary large object(SI_MaxContLength)
   ) returns SI_StillImage
method SI_changeFormat (
   targetFormat character varying( … )
   ) returns SI_StillImage
```

nur für unterstützte Formate

SQL/MM Still Image: Merkmale (Features)

- Basistyp SI_Feature hat die Subtypen:
  - **SI_AverageColor**: eine einzige Farbe für das ganze Bild
  - **SI_ColorHistogram**: Häufigkeiten für Gruppen von Farben (s. Kap. 4.3)
  - **SI_PositionalColor**: Zerlegung des Bildes in Rechtecke mit Durchschnittsfarbe
  - **SI_Texture**: Größe von wiederholten Elementen, Helligkeitsvariation, dominierende Richtung
- alle Merkmale haben die Methode SI_Score, die
  - die Distanz eines Bildes zum Merkmal berechnet und
  - einen Real-Wert zwischen 0 und 1 zurückgibt
- alle Subtypen von SI_Feature haben eine Funktion zur Merkmalsextraktion
- Objekte von SI_AverageColor und SI_ColorHistogram können direkt konstruiert werden (aus Konstanten)

SQL/MM Still Image: Merkmale (2)

```
create type SI_Feature
method SI_Score (image SI_StillImage)
   returns double precision
create type SI_AverageColor under SI_Feature as
   (SI_AverageColorSpec SI_Color)
method SI_AverageColor (
   RedValue integer,
   GreenValue integer,
   BlueValue integer
   ) returns SI_AverageColor
create function SI_AverageColor (image SI_StillImage)
   returns SI_AverageColor
```

SQL/MM Still Image: Merkmalsliste

- Liste von Merkmal-Wert-Paaren
- Methode SI_Score liefert gewichteten Mittelwert

```
self.SI_Features[1].SI_Score(img) * self.SI_Weights[1]
+ self.SI_Features[2].SI_Score(img) * self.SI_Weights[2] + …
/ (self.SI_Weights[1] + self.SI_Weights[2] + … )
```

```
create type SI_FeatureList as (
SI_Features SI_Feature array[SI_MaxFeatureNumber],
SI_Weights double precision array[SI_MaxFeatureNumber]
)
method SI_FeatureList (firstFeature SI_Feature, weight
double precision) returns SI_FeatureList
method SI_Append (feature SI_Feature, weight double
precision) returns SI_FeatureList
```


SQL/MM Still Image: Beispiel

```
select * from Logos where
   SI_FeatureList (
      SI_Texture (SI_StillImage(:bspLogo)), 0.8
      ).SI_Append (
         SI_ColorHistogram (SI_StillImage(:bspLogo)), 0.2
         ).SI_Score (Logo) > 0.7
```

SQL/MM – Ein SQL-Standard für Medienobjekte

- Schlussbemerkung
  - erst drei Teile standardisiert: FullText, Spatial, Still Image
  - für Video und Audio keine Teile in Sicht?
  - merkwürdige Uneinheitlichkeit
    (Rank bei FullText, Score bei StillImage) –
    warum nicht Generalisierung zu MM_Object o. ä.?
- Fragen
  - wird der Standard umgesetzt?
    - K. Stolze [Stol01a] zeigt, wie es bei Still Image mit DB2 gehen könnte
  - wie groß ist der Unterschied zu den bereits existierenden Paketen?