Fachpraktikum 1590
„Erweiterbare Datenbanksysteme“

Aufgaben Phase 2

Wintersemester 2004/2005

Ralf Hartmut Güting, Dirk Ansorge, Thomas Behr, Markus Spiekermann

Praktische Informatik IV, FernUniversität in Hagen

58084 Hagen

Vorwort

Liebe Studierende,

hiermit erhalten Sie nun die Aufgabenvorschläge zur zweiten Phase des Fachpraktikums. Arbeiten Sie die Aufgabenstellungen bitte durch und überlegen Sie sich Fragen dazu. Eine Zuteilung von Aufgaben zu Gruppen erfolgt in der zweiten Präsenzphase. Innerhalb der Gruppen findet dann eine ausführliche Diskussion der zugeteilten Aufgabe mit dem Ziel, eine möglichst genaue Spezifikation der Programmierarbeiten zu erstellen, statt.

Inhalt der Aufgaben ist die Einbettung von Multimedia-Datentypen (MIDI, JPEG, MP3). Für die Ein- und Ausgabe dieser Datentypen dürfen bereits existierende Bibliotheken verwendet werden, d.h. ein MP3-Player muß nicht selbst implementiert werden. Welche Bibliotheken darüberhinaus benutzt werden dürfen, ist mit den Gruppenbetreuern abzusprechen.

Viel Spaß mit den Aufgaben

Ihre Praktikumsbetreuer

1 MIDI in SECONDO

In dieser Aufgabe geht es darum, das MIDI[1]-Format in SECONDO einzuführen. Das MIDI-Protokoll ist eine binäre Beschreibungssprache für Musikstücke. Ursprünglich wurde es für Keyboards entwickelt und wird inzwischen in allen Bereichen rund um die Musik eingesetzt. Ein MIDI-File beinhaltet dabei eine Sequenz von Befehlen wie z.B. `Note_On(Tonhöhe, Lautstärke)`, denen stets ein Wert für die *DeltaTime*, die Zeitspanne für die Ausführungsdauer des Befehls, vorangeht. Neben diesen *Event Commands* gibt es noch eine Reihe von *Meta Events*, mit denen beispielsweise das Tempo, in dem ein Musikstück abgespielt werden soll, festgelegt werden kann.

Für diese Aufgabe ist es zunächst einmal wichtig, das MIDI-Format zu verstehen. Im Web finden Sie leicht eine Vielzahl von Beschreibungen dieses Formats. Einige Musikstücke im MIDI-Format, ein Hex/Binary-Editor, eine MIDI-fähige Soundkarte und ggf. noch ein Programm, das Musikstücke in notierter Form anzeigen kann, sind die weiteren Hilfsmittel, die Sie für diese Aufgabe benötigen.

Die von Ihnen zu lösenden Teilaufgaben sind:

1. Erstellen einer MIDI-Algebra, die (mindestens) die folgenden Operationen enthält:

- `extract_track(midi,track_no)` erzeugt ein MIDI-Objekt, das nur noch den gewählten Track enthält.
- `merge_tracks(midi,track_no1,track_no2)` erzeugt ein MIDI-Objekt, in der der zweite Track in den ersten Track gemischt wird.
- `transpose_track(midi,track_no,halftone_no)` transponiert den gewählten Track um die angegebene Anzahl von Halbtönen nach oben (positiver Wert) oder nach unten (negativer Wert).
- `extract_lyrics(midi)` erzeugt einen String, der den Text des Musikstücks enthält
- `contains_words(midi,words)` liefert *true*, wenn die Lyrics des Musikstücks die gesuchte Textpassage vom Typ `string` enthalten.
- `contains_sequence(midi,sequence,allow_trans)` liefert die Position im Musikstück zurück (bzgl. *DeltaTime*), an der die gesuchte Sequenz gefunden wurde, sonst -1. `sequence` ist dabei ein einfaches, auf Zeichenketten basierendes Format um eine Notenfolge ohne Tonlängen anzugeben, z.B. „C-D-E-F-G-G“. Benutzt wird die englische Schreibweise (d.h. „B“ statt „H“); für Halbtöne werden „Cb“ und „C#“ verwendet.
- `expand(midi,track_no)` erzeugt ein neues MIDI-Objekt in der jeder Kanal einen einzelnen Track belegt.

2. Einbinden der Algebra in SECONDO, incl. der Integration der neuen Datentypen für Relationen.

3. Schreiben eines Viewers/Players für die Datentypen der MIDI-Algebra.

- Abspielmöglichkeit für MIDI-Objekte, auch ab einer spezifizierten Position
- Anzeige der verschiedenen in der Datei enthaltenen Informationen
- ggf. Anzeige der Musikstücke in notierter Form

1. Musical Instrument Digital Interface

2 Integration von JPEG-Bilddaten

In diesem Projekt sollen Bilddateien von SECONDO verarbeitet werden. Die wichtigsten Funktionen, nämlich die Konvertierung von einem komprimierten JPEG-Format in eine Rastergrafik und umgekehrt, sind in der Bibliothek `libjpeg`[1] der Independent JPEG-Group zu finden. Für die Verarbeitung von JPEG-Daten in Java-Programmen sollten die Möglichkeiten des Java Image I/O (javax.imageio)[2] von SUN ausreichend sein.

Die `PictureAlgebra` besteht aus dem Datentyp `picture`, der eine interne Darstellung für eine JPEG-Datei implementiert. Einfache Operationen zur Abfrage von Metadaten sind:

- `height: picture -> int`
- `width: picture -> int`
- `isgrayscale: picture -> bool`

Die folgenden Operationen sollen Informationen extrahieren, die entweder beim Import eines Bildes in SECONDO angegeben werden, oder - falls vorhanden bzw. möglich - im Bilddatenformat bereits enthalten sind.

- `filename: picture -> string`
- `category: picture -> string`
- `date: picture -> instant`
- `isportrait: picture -> bool`

Um auch das Suchen auf Bilddaten zu ermöglichen, soll zusätzlich der Datentyp `histogram`, der auf einer Skala 0-255 von diskreten Helligkeitswerten den Prozentsatz der auf jeden Wert entfallenden Pixel darstellt, realisiert werden. Folgende Operationen sind zu implementieren:

- `colordist: picture x int -> histogram`
- `equals: picture x picture x int x int -> bool`
- `like: picture x int x int x int x int -> bool`

Der `int` Parameter für `colordist` entspricht bei Farbbildern einem Farbkanal: 0=red, 1=green, 2=blue und für ein Graustufenbild dem Wert 3=brightness. Der Operator `equals(`b_1`,`b_2`,n,p)` erlaubt den Vergleich der Histogramme eines Bildes b_1 mit einem Referenzbild b_2. Falls für jeweils `n` aufeinanderfolgende Helligkeitswerte der Mittelwert m_1 um höchstens `p` Prozent vom Mittelwert m_2 abweicht, sind die Bilder als gleich anzusehen. Der Operator `like(b,p,t,l,u)` hingegen ermöglicht eine Suche ohne Referenzbild, indem geprüft wird, ob $p \pm t$ Prozent der Pixel im Helligkeitsintervall `[l, u]` enthalten sind.

Bilddaten sollen über die folgenden Operationen manipuliert werden können:

- `scale: picture x int x int -> picture`
- `cut: picture x int x int x int x int -> picture`

---

1. http://www.ijg.org. Eine Dokumentation der Programmierschnittstelle ist in der Datei `libjpeg.txt` in der Distribution der Bibliothek enthalten.
2. Die reine über Javadoc generierten API-Spezifikationen sind i.d.R. etwas verwirrend und eher als Nachschlagwerk beim Programmieren zu verstehen. Als Einstieg sollte daher die Developer Documentation unter http://java.sun.com/j2se/1.4.1/docs/guide/imageio/index.html, die einen konzeptuellen Überblick vermittelt, studiert werden.

- `flipleft: picture x int -> picture`
- `mirror: picture x bool -> picture`

Der Operator `scale` skaliert das Bild maßstabsgetreu, so daß das Bild in die durch die Parameter angegebene Bounding Box paßt. `cut` liefert einen angeforderten Bildausschnitt. Die Operation `flipleft(n)` dreht das Bild `n` mal 90 Grad nach links und `mirror` spiegelt das Bild vertikal falls der `bool` Parameter den Wert *true* hat, andernfalls horizontal.

Hinweis: Die `libjpeg` Bibliothek komprimiert bzw. dekomprimiert Bilder standardmäßig aus Dateien. Um dieses Verhalten zu ändern, muß ein eigener Source- bzw. Destination-Manager implementiert werden.

Im Javagui soll ein Viewer implementiert werden, der `picture` und `histogram` Objekte darstellen kann. Weiterhin sollte der Viewer auch Ergebnisrelationen, die Bilddatenattribute bzw. Histogramme besitzen, übersichtlich darstellen. Denkbar wäre z.B. ein Filmstreifen mit vielen kleinen Ansichten und daneben eine größere Ansicht eines ausgewählten Bildes zusammen mit der Anzeige weiterer Relationenattribute.

3 Einbindung von MP3-Dateien in SECONDO

MP3 ist ein sehr verbreitetes Format zur Speicherung von Audiodaten. Ein großer Vorteil dieses Formats ist vor allem das geringe Datenvolumen bei sehr guter Qualität. Das Verhältnis zwischen den beiden kann durch die verwendete Bitrate den eigenen Bedürfnissen angepaßt werden. Ein weiterer Vorteil ist, daß zusätzliche Informationen in eine solche Datei integriert werden können. Ihre Aufgabe ist es, MP3-Dateien und einige Erweiterungen in SECONDO verfügbar zu machen. Weiterhin soll die Java-Oberfläche derart erweitert werden, daß alle Typen Ihrer Algebra angezeigt (bzw. angehört) werden können.

Ihre Algebra soll 3 Typen enthalten, nämlich `MP3`, `ID3` und `LYRICS`. Für die `id3`-Erweiterung gibt es unterschiedliche Möglichkeiten. Ihre Algebra soll die Versionen 1 und 1.1 unterstützen.

Es sind die folgenden Operatoren auf den Typen zu implementieren:

- `id3 : MP3 -> ID3`
  Extrahiert die Information des `ID3`-Tags.
- `store_id3 : MP3 x ID3 -> MP3`
  Legt die Information aus `ID3` im gegebenen `MP3` ab.
- `remove_id3 : MP3 -> MP3`
  Entfernt die Information aus einem `MP3`.

Die oben angegeben Operationen sind äquivalent für `LYRICS` anzubieten.

- `author : ID3 -> string`
  Extrahiert die Information aus einem `ID3` Wert. Entsprechende Operationen sind auch für alle anderen enthaltenen Informationen anzubieten.
- `words : LYRICS x int -> string`
  Gibt das Wort (oder die Wortfolge) für einen gegebenen Zeitpunkt zurück.

- `bitrate : MP3 -> int`
  Ermittelt die Bitrate eines `MP3`. Entsprechende Operationen sind auch für Frequenz, Dauer, Codierung, Anzahl der Frames usw. zu implementieren.
- `submp3 : MP3 x int x int -> MP3`
  Extrahiert die Folge von Frames, die durch die beiden hinteren Parameter gegeben sind, aus einem `MP3`. Beachten Sie dabei auch, daß in evtl. enthaltenen `LYRICS` Daten entsprechende Änderungen vorzunehmen sind.
- `concat: MP3 x MP3 -> MP3`
  Fügt die gegebenen `MP3`-Daten zusammen.

Entwerfen Sie einen Viewer, der die von Ihnen implementierten Datentypen darstellen kann. Der Viewer soll in die vorhandene Java-Oberfläche eingebettet werden. Zusätzlich soll der Viewer auch Relationen, die Werte der Algebratypen als Attribute enthalten, verarbeiten können.

Allgemeine Hinweise zur Darstellung von Binärdaten im Nested-List Format

Binärdaten können in Textatomen dargestellt werden, dazu werden die Daten als Base64[1] codierte Zeichenfolge dargestellt. Dabei werden je 3 Bytes der Binärdaten auf 4 Zeichen eines 64 Zeichen umfassenden Alphabets abgebildet. In C++ und Java wurden Methoden für das codieren und decodieren von Base64 Daten in SECONDO implementiert. Der initiale Import von Daten wird durch die Bereitstellung eines `<file></file--->` Tags erleichtert. Beim `restore` Befehl ersetzt der Nested-List Parser diesen Ausdruck durch ein Textatom, in dem die Daten aus der Datei als Base64 codierte Zeichenfolge enthalten sind. Am Beispiel der `BinaryFileAlgebra` kann man die Benutzung der Base64-Klasse studieren.

1. Spezifiziert im RFC 1341 (MIME - Multi Purpose Internet Mail Extensions). Quelle, z.B.: http://www.faqs.org/rfcs