Inhaltsverzeichnis

Kapitel 1

Das Projekt eNoteHistory

Das Projekt eNoteHistory ist ein Gemeinschaftsprojekt des Instituts für Musikwissenschaft der Universität Rostock, des Datenbanklehrstuhls der Informatik und des Fraunhofer Institut für Grafische Datenverarbeitung (IGD). Ziel dieses Projektes ist es die Schreibererkennung von Notenhandschriften mit Hilfe von Computeranalysen zu vereinfachen. Die Internetseiten sowie das lauffähige System des Projektes stehen unter www.enotehistory.de zur Verfügung.

1.1 Einführung in die Aufgabenstellung

Für die Musikwissenschaft ist es von großer Bedeutung, Notenhandschriften ihren Schreibern zuordnen zu können. Dadurch kann man eingrenzen, wann, wo, warum und für wen eine Komposition geschrieben bzw. abgeschrieben wurde, und auch Aussagen über die Verbreitung der Komposition treffen. Anhand der individuellen Handschrift kann man einen Schreiber identifizieren; auch verraten das verwendete Papier, die Tinte und das Wasserzeichen weitere Einzelheiten. Die Rostocker Universitätsbibliothek hat ihre Sammlung von Notenhandschriften dafür zur Verfügung gestellt, die aus den verschiedensten Gebieten Europas und vorwiegend aus dem 18. Jahrhundert stammen. Diese Notenblätter dienen als Informationsquelle zur Schreibererkennung In den folgenden Abschnitten werden die einzelnen Aufgabenbereiche vorgestellt, die zur Realisierung dieses Projektes beitragen. Kapitel 2 beschreibt die technische Umsetzung des Systems und Kapitel 3 das Datenmodell in der Datenbasis. Die zu Grunde liegende Logik des Schreibererkennungssystem besteht aus Data Mining Techniken, welche im Kapitel 4 vorgestellt werden. Die Bedienung des lauffähigen Systems wird anschließend im Kapitel 5 erklärt. Als weiterführende Aufgabe gilt es, das Projekt in eine Content Managment Umgebung zu migrieren, welches im Kapitel 6 dargestellt wird.

1.2 Musikwissenschaftliche Grundlagen

Um eine Menge von Notenhandschriften ihren Schreibern zuordnen zu können, geht man davon aus, dass jeder Schreiber eine individuelle Schrift hat. Somit werden zuerst die Merkmale von Schrift definiert, anhand deren man eine Schreibercharakteristik erstellen kann, die zur Unterscheidung der Schriftstile und somit der einzelnen Schreibers verwendet wird. Die Musikwissenschaftler haben die Notenschrift mit ihren Merkmalen in 13 Merkmalsklassen eingeteilt: Schlüssel, Taktvorzeichen, Viertelpausen, Schriftneigung, Kaudierung, Form der Fähnchen, Schreibgewohnheiten, Laufweite, Bebalkung, Vorzeichen, Form der Notenköpfe, Schlusszeichen und Rastral. In diesen Merkmalsklassen gibt es insgesamt ungefähr 80 Merkmale, Features genannt, welche die Schrift eines Schreiber repräsentieren. Es wurde das hierarchische Ordnungssystem *Feature Base* innerhalb der Merkmalsklassen entwickelt, welches die Zerlegung komplexer Zeichen in ihren einzelnen Grundelementen gestattet.

1.3 Aufbau der Feature Base

Ein Merkmal, das zur Beschreibung einer Handschrift dient, wird *Feature* genannt. Diese Features werden in hierarchischer Form angelegt und bilden die *Feature Base*, auch Feature Dictionary genannt. Sie kann als azyklischer gerichteter Baum betrachtet werden, dessen Knoten von folgenden Typen sein können:

- Der **Feature-Prefix**-Knoten spezifiziert den Namen des Features
- Der **Feature**-Knoten ist der letzte Knoten des Baumes und spezifiziert ebenfalls den Namen des Features
- Der **Value**-Knoten ist das Blatt des Baumes, befindet sich direkt unter dem Feature-Knoten und repräsentiert den Wert des Features
- Der **Value-Prefix**-Knoten dient nur zur Weiterleitung in der Hierarchie

Jeder Knoten wird durch eine Punktnotation, *Feature-Code* genannt, identifiziert, und sie beschreibt auch gleichzeitig den Pfad von Wurzel bis zum Feature. Eine Liste aller existierenden Features ist in der Tabelle 1.1 zu sehen.

Die Feature Base enthält auch alle möglichen Werte eines Features, die sich genau unter dem Feature-Knoten in der Baumhierarchie befinden. Die Werte können Piktogramme oder Textbeschreibungen des Features sein. Die Abbildung 1.1 zeigt die Baumhierarchie in der Merkmalsgruppe der Fähnchen mit den Feature-Codes und den Feature-Werten.

Die Wertebereiche eines Features können auch durch mehrere Ebenen repräsentiert werden. Jede Ebene beschreibt ein Teilstück eines komplexen Symbols auf einem Notenblatt.

Feature-Code	Beschreibung
1.1.1	ein einzügiges Grundelement eines G-Schlüssel
1.1.2	mehrzügiger G-Schlüssel, Zusatzzeichen (Punkte, Striche etc.) können zur bereits ermittelten Schlüsselform auftreten
1.2.1	linkes Element bei einem C-Schlüssel
1.2.2.1	Markierung der C-Linie bei einem C-Schlüssel durch ein Element
1.2.2.2.1	oberes Element, wenn die Markierung der C-Linie bei einem C-Schlüssel durch mehrere Elemente erfolgt
1.2.2.2.2	unteres Element, wenn die Markierung der C-Linie bei einem C-Schlüssel durch mehrere Elemente erfolgt
1.2.3	rechtes Element bei einem C-Schlüssel
1.3.1	Hauptelement bei einem F-Schlüssel
1.3.2.1	Zusatzelement links bei einem F-Schlüssel
1.3.2.2	Zusatzelement (Hauptelement schneidend) bei einem F-Schlüssel
1.3.2.3	erstes Zusatzelement rechts bei einem F-Schlüssel
1.3.2.4	zweites Zusatzelement rechts bei einem F-Schlüssel
9.1.1.1.1	oberer Bügel bei dem Taktvorzeichen C (von C/2)
9.1.1.1.2	frei hinzugefügte Zeichen bei dem Taktvorzeichen C (von C/2)
9.1.1.2	unterer Bügel bei dem Taktvorzeichen C (von C/2)
9.1.2.1	Bügel bei dem Taktvorzeichen 2 (von C/2)
9.1.2.2	Basisstrich bei dem Taktvorzeichen 2 (von C/2)
9.1.2.3	Mittelmarke bei dem Taktvorzeichen 2 (von C/2)
9.1.3.1.1	oberer Abschluss bei dem Taktvorzeichen Strich/Doppelstrich (von C/2)
9.1.3.1.2	unterer Abschluss bei dem Taktvorzeichen Strich/Doppelstrich (von C/2)
9.1.3.2	Schnittpunkte, Lage weiterer Marken bei dem Taktvorzeichen Strich/Doppelstrich (von C/2)
9.2	Taktvorzeichen 1
9.3.1	Bügel bei dem Taktvorzeichen 2
9.3.2	Basisstrich bei dem Taktvorzeichen 2
9.3.3	Mittelmarke bei dem Taktvorzeichen 2
9.4.1	Zahl bei dem Taktvorzeichen 3
9.4.2	Zusatzstrich bei dem Taktvorzeichen 3
9.5.1	Winkel bei dem Taktvorzeichen 4
9.5.2	Senkrechte bei dem Taktvorzeichen 4
9.5.3	Sonderformen bei dem Taktvorzeichen 4

9.6	Taktvorzeichen 8
9.7.1	Bruchstrich bei dem Taktvorzeichen
9.7.2	Ligatur von Zähler und Nenner bei dem Taktvorzeichen
13.1	Viertelpause
2.1.1	Schriftneigung für die aufwärts kaudierte Halbe Note
2.1.2	Schriftneigung für die abwärts kaudierte Halbe Note
2.2.1	Schriftneigung für die aufwärts kaudierte Viertelnote
2.2.2	Schriftneigung für die abwärts kaudierte Viertelnote
2.3.1	Schriftneigung für die aufwärts kaudierte Achtelnote
2.3.2	Schriftneigung für die abwärts kaudierte Achtelnote
2.4.1	Schriftneigung für die aufwärts kaudierte Sechzehntel Note
2.4.2	Schriftneigung für die abwärts kaudierte Sechzehntel Note
3.1.1	Kaudierung der halben Note aufwärts kaudiert
3.1.2	Kaudierung der halben Note abwärts kaudiert
3.2.1	Kaudierung der Viertelnote aufwärts kaudiert
3.2.2	Kaudierung der Viertelnote abwärts kaudiert
3.3.1	Kaudierung der Achtelnote aufwärts kaudiert
3.3.2	Kaudierung der Achtelnote abwärts kaudiert
3.4.1	Kaudierung der Sechzehntel Note aufwärts kaudiert
3.4.2	Kaudierung der Sechzehntel Note abwärts kaudiert
4.1.1	Fähnchen bei der aufwärts kaudierten Achtelnote
4.1.2	Fähnchen bei der abwärts kaudierten Achtelnote
4.2.1	Fähnchen bei der aufwärts kaudierten Sechzehntel Note
4.2.2	Fähnchen bei der abwärts kaudierten Sechzehntel Note
11.1	Kustoden
11.2	Vorhandensein von Schlüsseln
11.3	Vorhandensein von Tonartvorzeichen
11.5	b oder # als Auflösungszeichen
11.4	Format des Notenblattes (hoch oder quer)
12	Laufweite, Abstand der Noten
5.1	Relation Balken - Stiele bei der Bebalkung
5.2	Verlauf des Balkens
5.3	Balkenabstand bei der Mehrfachbebalkung
6.1	Ligatur mit jeweiligem Schlüssel
6.2	Ligierung und Vervielfachung von Tonartvorzeichen
6.3	Vorzeichen b
6.4	Auflösungszeichen
6.5	Vorzeichen #

8.1	schwarzer Notenkopf
8.2.1.1	weißer einzügiger Notenkopf aufwärts
8.2.1.2	weißer einzügiger Notenkopf abwärts
8.2.2.1	weißer mehrzügiger Notenkopf aufwärts
8.2.2.2	weißer mehrzügiger Notenkopf abwärts
10.1	erstes Zeichen beim Schlusszeichen
10.2	weiteres Zeichen beim Schlusszeichen
7.1	Größe des Rastral
7.2	Linienabstände im System
7.3	Anordnung der Notensysteme (Art des Rastrals)

Tabelle 1.1: Features der Handschriftcharakteristik

Jedem Feature in der Feature Base kann ein Wert zugeordnet werden. Somit entsteht ein Feature-Vektor, der die individuelle Handschriftcharakteristik eines Schreibers repräsentiert. Es können auch mehrere Werte, so genannte *komplexe Feature-Werte*, für ein Feature existieren. Die Feature Base und die Feature-Vektoren mit ihren Werten werden in der Datenbank gespeichert. Im Kapitel 3 wird speziell auf die gespeicherten Daten mit den entsprechenden Relationenschemata eingegangen.

1.4 Import der Daten

Die Rostocker Universitätsbibliothek hat ihre Sammlung von Notenhandschriften dem Projekt enote-History zur Verfügung gestellt. Zuerst wurde eine Testmenge von ungefähr 4000 Notenblätter als TIFF-Format mit 300 dpi und 24 bit Farbtiefe eingescannt. Einerseits wurden die Digitalisate mit hoher Qualität gespeichert, um die Extraktion von visuellen Charakteristiken bei den Bildbearbeitungsalgorithmen zu gewährleisten. Andererseits wurde auch ein komprimiertes Bild gespeichert, welches zur Ansicht und als Navigationshilfe in Form eines Thumbnails für die Nutzerschnittstelle verwendet wird.

Weiterhin wurden Informationen aus dem Bibliothekskatalog wie Signatur, Name des Komponisten, Dokumentbeschreibungen und andere Kommentare in die Datenbank integriert. Um die in LaTex vorliegenden Bibliotheksdaten auf das Datenbankschema abzubilden, wurde eine XML-Struktur als Zwischenformat gewählt. Um die LaTex-Daten in die gewünschte XML-Struktur umzuwandeln, wurden Perl-Scripte verwendet. Das Perl-Script `pre_transform.pl` konvertiert die LaTex-Bibliotheksdaten zu HTML, und das Perl-Script `extract_all.pl` die resultierende HTML-Struktur in eine XML-Struktur, dessen Inhalte anschließend in die Datenbank gespeichert werden konnten.

Die Daten der Feature Base lagen in einer hierarchischen Form von HTML-Dateien vor, und konnten auch in die Datenbank mittels des entwickelten Kommandozeilen-Tools `populateFeatureBaseTable` integriert werden. Zur Berechnung eines Ähnlichkeitsmaßes

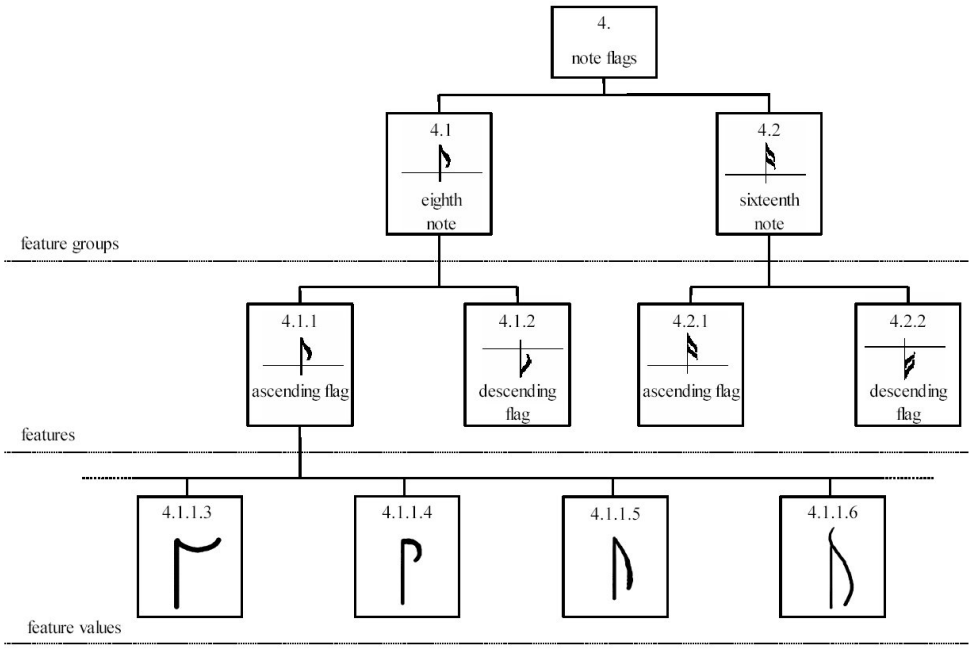


Abbildung 1.1: Baumhierarchie in der Merkmalsgruppe der Notenfähnchen

zwischen Handschriften werden Distanzmatrizen verwendet. Diese wurden von den Musikwissenschaftlern in Form von Excel-Dateien erstellt und in die Datenbank übertragen.

Anschließend wurden die Merkmale anhand der Feature-Base für 150 Notenhandschriften extrahiert, und die resultierenden Feature-Vektoren der Handschriften in der Datenbank gespeichert. Es wurden Klassen mit ähnlichen Handschriftmerkmalen erstellt, die genau einen Schreiber repräsentieren. Nun können neue unbekannte Feature-Vektoren klassifiziert werden, indem sie einer dieser Klassen zugeordnet werden. Dieser Prozess ist in Abbildung 1.2 dargestellt.

Der Zugriff auf die Daten erfolgt durch eine Menge von Schnittstellenfunktionen, die von einer Client-Anwendung genutzt werden. Sie bietet Funktionen zum Einfügen, Bearbeiten, Anfragen und Durchsuchen der Daten.

Eine Webschnittstelle mit Navigations- und Suchfunktionalität steht jedem Nutzer unter www.enotehistory.de zur Verfügung. Kapitel 5.1 gibt eine detaillierte Funktionsweise bzw. Benutzungshinweise dieser Webseite. Der Nutzer kann historische Informationen abfragen, auf die digitalisierten Notenhandschriften und deren Metadaten zugreifen und Anfragen an das System stellen. Wenn der Nutzer einen Schreiber einer unbekannten Notenhandschrift bestimmen möchte, muss er die Merkmale der unbekannten Handschrift manuell bestimmen, und das System ordnet mittels Klassifikationstechniken die Handschrift einer oder mehrerer Schreiberklassen zu. Dazu werden mit Hilfe einer Distanzfunktion die Distanzen zu den vorhandenen Feature-Vektoren berechnet. Liegt die Distanz zu einem Feature-Vektor unter einem gewissen Schwellwert, wird der dazugehörige Schreiber als

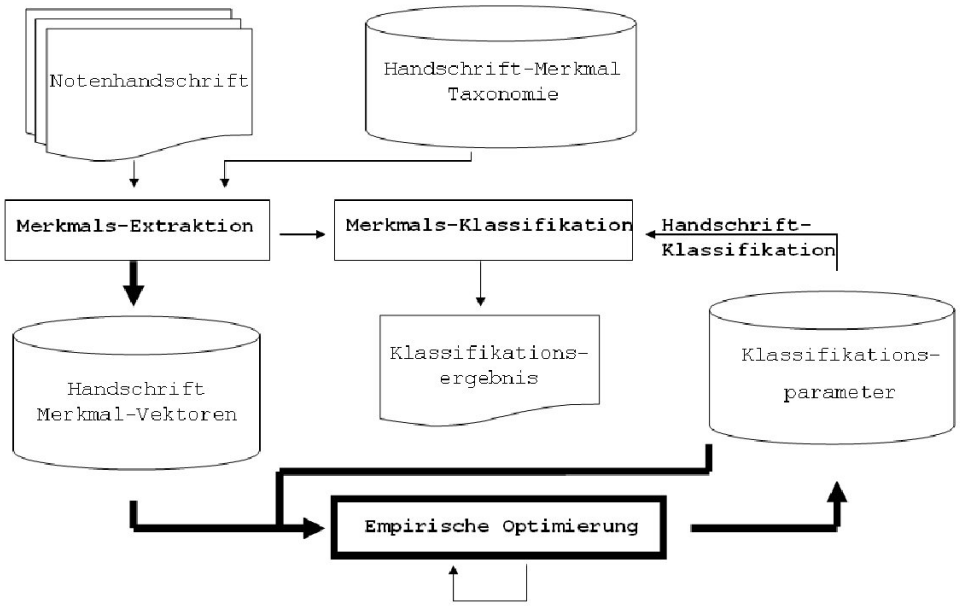


Abbildung 1.2: Klassifizierung von Notenhandschriften

möglicher Treffer ausgegeben. Die Berechnung der Distanzen wird durch eine UDF in der DB2 Datenbank realisiert. Eine nähere Beschreibung der Distanzfunktion und den UDFs in der Datenbank wird im Kapitel 4 gegeben.

Die Abbildung 1.3 zeigt den Workflow der Musikdaten vom Scannen der Original-Notenblätter über die Analyse und Speicherung der Digitalisate bis zum Zugriff auf die Daten.

1.5 Automatische Handschriftenanalyse

Neben der manuellen Analyse der Notenhandschriften wird ein zweiter Ansatz zur automatischen Handschriftenanalyse vom Projektpartner Fraunhofer-Institut verfolgt. Ziel dieses Ansatzes ist es, die Merkmale der Notenhandschrift durch rechnergestützte Bildverarbeitungsfunktionen zu extrahieren. Dieser Prozess besteht aus mehreren Stufen. In der Bildvorbearbeitung werden Verfahren angewendet, die das Bildrauschen verringern, eine Trennung von Vordergrund und Hintergrund vornehmen und andere Bearbeitungen, um die weitere Analyse zu gewährleisten. Anschließend erfolgt die Erkennung des Notensystems, die Erkennung von Primitiven wie Notenköpfe und -hälse, Taktstriche und komplexere Symbole. Eine genaue Objekterkennung erfolgt indem die Primitiven verglichen und entsprechend zusammengesetzt werden. Daraufhin ist eine Analyse der Charakteristik der Handschrift mittels genauer Messungen möglich. Zur Unterstützung der automatischen Handschriftenanalyse wurde ein BV-Softwaresystem entworfen, das sich im Laufe des Projektes zu einem komplexen Analysesystem entwickeln wird. Für detaillierte Beschreibungen

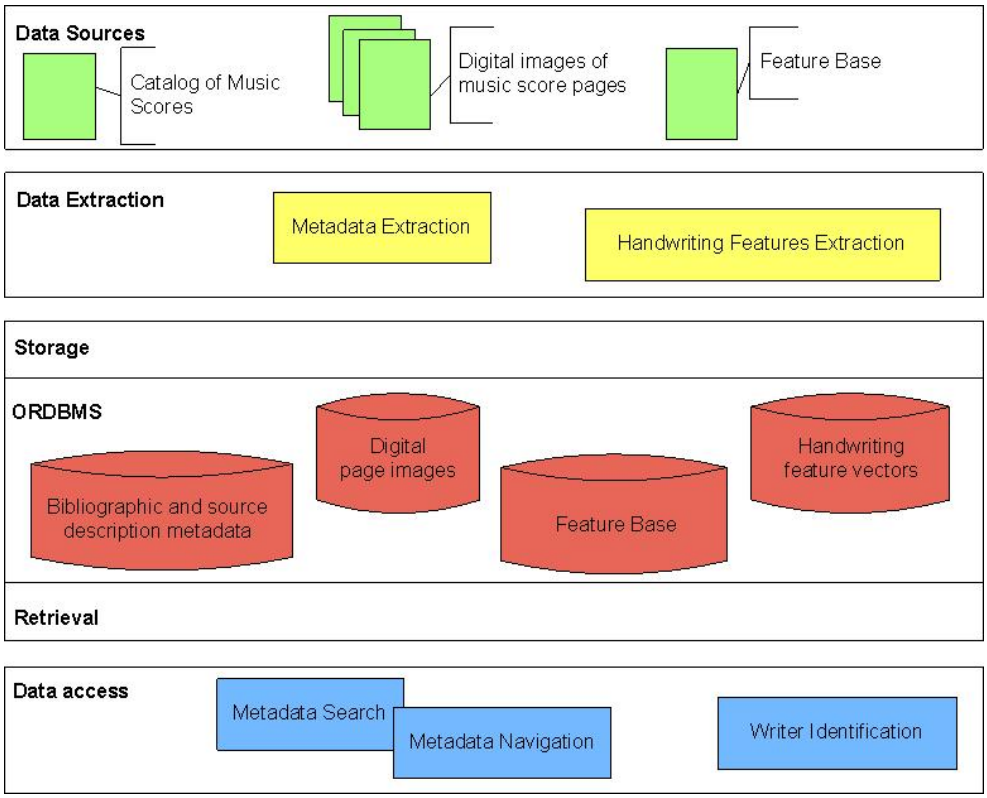


Abbildung 1.3: Workflow der Musikdaten

der einzelnen Verfahren zur automatischen Analyse von Handschriftencharakteristiken wird auf die Berichte des Fraunhofer-Instituts für Graphische Datenverarbeitung in Rostock verwiesen.

Kapitel 2

Technische Umsetzung

Für das eNoteHistory-Projekt wurde ein Server, namens ISIS, in dem Institut für Informatik der Universität Rostock errichtet. Zur technischen Ausstattung ist zu erwähnen, dass der Server mit einem Intel Xeon Prozessor MP mit 2,8 GHz für Multi-Prozessor Servern und mit 3 GB RAM ausgestattet ist. Weiterhin wurde das Betriebssystem Windows 2000 und zur Verwaltung der Daten wurde das Datenbanksystem DB2 (DB2 Universal DB Version 8) mit den Extender-Techniken (XML- und Net Search Extender) installiert, in dem die Datenbank `enote` angelegt wurde.

2.1 Systeminstallation unter Win32

Zur Umsetzung des Projektes bzw. zum Aufbau des eNoteHistory-Servers wurden folgende Komponenten verwendet. Um eine Entwicklungsumgebung für Java einzurichten, bot sich an, Eclipse zu benutzen, da es frei verfügbar ist und viele Vorteile gegenüber anderen Alternativen bietet. Als Webserver empfiehlt sich Tomcat, da es leicht zu installieren ist, frei verfügbar und einfach mit Eclipse mit Hilfe eines Plugins verwendbar ist. Zuerst müssen die folgenden Komponenten herunter geladen, installiert und entsprechend konfiguriert werden.

Tomcat

Als Applikationsserver fungiert Tomcat 4.1.24, dessen neuste Version man unter *http://jakarta.apache.org/tomcat/index.html* herunterladen kann. Tomcat 4.1 unterstützt Standards von Servlet 2.3 und JSP 1.2. Unter Windows ist Tomcat einfach zu installieren, da sich das Installer-Script um die meisten Konfigurationsschritte kümmert. Als Tomcat Context ist "Web Application" anzugeben, und merken Sie sich das Administratorpasswort und den Installationspfad. Man sollte abschließend testen, ob die Seite http://localhost:8080/ korrekt funktioniert.

Sun Java SDK

Es empfiehlt sich die neuste SDK Version herunter zu laden; momentan J2SE 1.4.2 SDK von Sun (j2sdk-1_4_2_04-windows-i586-p.exe = 48,2 MB). Das Eclipse Tomcat plugin benötigt ein SDK mit einem Java Compiler. Sobald das SDK installiert wurde, kann die Eclipse workbench gestartet werden.

Eclipse Tomcat launcher plugin von Sysdeo

Das Plugin von Sysdeo kann unter http://www.sysdeo.com/eclipse/tomcatPlugin.html (aktuell tomcatPluginV3alpha1.zip) herunter geladen werden. Das Unterverzeichnis sollte nach $< eclipse > \backslash plugins$ entpackt werden. Unter "$Help > AboutEclipsePlatform > Plugin - Details$" sollte nun das Sysdeo Plugin auftauchen. Es steht ein neuer Projekttyp zur Verfügung, sowie Steuerungsmöglichkeiten für Tomcat direkt aus Eclipse. Nach der Installation wird sich die Symbolleiste und das Menü entsprechend Abbildung 2.1 verändern.

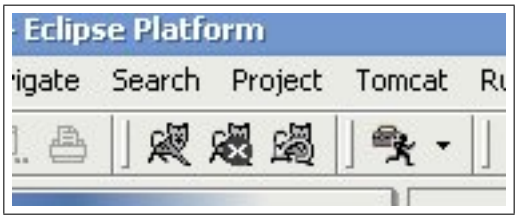


Abbildung 2.1: Tomcat-Plugin in Eclipse

Eclipse IDE

Eclipse kann man auf den Seiten www.eclipse.org unter Downloads herunterladen werden; momentan ist Eclipse 3.0M9 (eclipse-SDK-3.0M9-win32.zip) aktuell. Nach dem Entpacken dieser Datei, muß der Ordner des Tomcat plugins in den `plugins`-Ordner von Eclipse (Eclipse/plugins) kopiert werden. Nach dem Installieren von Eclipse werden einige Konfigurationsschritte nötig:

- den Workspace festlegen
- die installierte SDK JRE (Java Runtime Environment) als default JRE für Eclipse auswählen: unter "$Window > Preferences > Java > InstalledJREs$" eine JRE definieren, die dem vollen JDK entspricht: JRE home directory: C:\j2sdk1.4.2_03 (in unserem Beispiel)
- den Application Server Tomcat in Eclipse einbinden: unter "$Window > Preferences > Tomcat$" Version und Pfade festlegen (unter JVM dieselbe JRE festlegen):

Tomcat home: *C:\Program Files\Apache Group\Tomcat 4.1*
Tomcat base: *C:\Program Files\Apache Group\Tomcat 4.1*
Configuration file: *C:\Program Files\Apache Group\Tomcat 4.1\conf\server.xml*

- die Tomcat Menüpunkte aktivieren: unter "*Window > CustomizePerspective > Commands*" die Tomcat Checkbox aktivieren; Anschließend wird es ein paar neue Knöpfe in der Symbolleiste und einen neuen Menüeintrag Tomcat wie in Abbildung 2.1 geben.

Weitere Dokumentationen über Installation, Konfiguration und erste Beispiele findet man unter:

http://www-106.ibm.com/developerworks/opensource/library/os-ectom/
http://javaboutique.internet.com/tutorials/three/
http://www.joller-voss.ch/tools/eclipse/EclipseTomcatStarthilfe.pdf
http://www.3plus4software.de/eclipse/tomcat.html
http://www.keyboardsamurais.de/mt/archives/000053.html

Installation von DB2

Zur Installation des IBM Datenbanksystems DB2, benötigt man die Installations-CD-ROM. Während dem Installationsprozess braucht man nur den Anweisungen folgen. Anschließend lädt man das Fixpack 5 auf der Seite http://www-306.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v8fphist.d2w/report herunter (FP5_WR21334_ESE.exe = 498,5 MB).

Anlegen eines Projektes in Tomcat

Ein neues Tomcat Projekt kann man mit "*File > New > Project*" anlegen, und dann im Ordner "Java" das Symbol für Tomcat auswählen. Als Name wählt man "enotehistory". Die Tomcat-Konfiguration wird bei den Tomcat Project Settings angepasst:

Context Name: /enote1
Can update server.xml: yes
Subdirectory to set as web application root: /

Zum Testen der Integration legt man index.jsp an, und browst zur Adresse http://localhost:8080/enote1/

Import der Daten in DB2

Man startet das Control Center von DB2, und erstellt mit einem Rechtsklick auf Databases eine Datenbank namens enote1. Um existierende Daten als Backup (enote1.0.zip = 658 MB) in die neue Datenbank zu importieren, wählt man Restore beim Rechtsklick auf "Datenbank", und wählt die folgenden Optionen:

> Restore to an existing database
> Media Type: File System
> Pfad: das Verzeichnis, dass das ENOTE1.0-Verzeichnis nach dem Entpacken der zip-Datei `enote1.0.zip` enthält
> Datum: Es ist wichtig, dass das Datum mit dem Backupdatum übereinstimmt. Der Tag entspricht dem untersten Verzeichnisnamen und die Zeit entspricht dem Dateinamen.

Import der bereits existierenden Webpräsens

Das komplette Webverzeichnis muss kopiert werden, damit es keine Probleme mit dem Deployment Descriptor und den Einstiegspunkten gibt.

Wichtige Dokumentationen

Servlet / JSP API (lokal): http://localhost:8080/tomcat-docs/servletapi/index.html
Tomcat Dokumentation wird mitgeliefert: http://localhost:8080/tomcat-docs/
DB2 Dokumentation:
http://www-306.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v8pubs.d2w/en_main

Artikel

ONJava: Using Tomcat Rubrik: http://www.onjava.com/pub/ct/33
ONJava: JSPs und Servlets: http://www.onjava.com/topics/java/JSP_Servlets
Tomcat und DB2: http://www.midrangeserver.com/fhg/fhg051204-story01.html und http://www.logemann.org/day/archives/000059.html
Netbeans 3.6 ist keine passende Alternative:
http://www.onjava.com/pub/a/onjava/2002/07/17/netbeans.html
Eclipse: http://www.onjava.com/pub/a/onjava/2002/12/11/eclipse.html

2.2 Datenbank

Die Daten des Projektes, wie die Digitalisate und die Metadaten der Notenhandschriften, die Struktur der Feature Base und die Handschriftencharakteristiken der Schreiber sind in

einer DB2 Datenbank, namens `enote` gespeichert. Man kann auf zwei Arten auf die Daten in dieser Datenbank zugreifen.

- Es gibt einige zum Datenbanksystem DB2 zugehörige Anwendungen, die nur mit einem direkten Zugang zum Server genutzt werden können.
- Es wurde eine Web-Anwendung mit der Verwendung von Java Servlets und Tomcat entwickelt, die einen eingeschränkten Zugriff auf die Daten ermöglicht. Das dazugehörige Web-Interface findet man unter *www.enotehistory.de/analyse.html*. Dort können Anfragen gestellt werden, in denen man einige Schriftmerkmale angibt, und sich dazu die möglichen Schreiber ausgeben lässt. Ein einfaches Auflisten der Schriftmerkmale, die Ausgabe der eingescannten Notenblätter und anderen Informationen sind auch möglich.

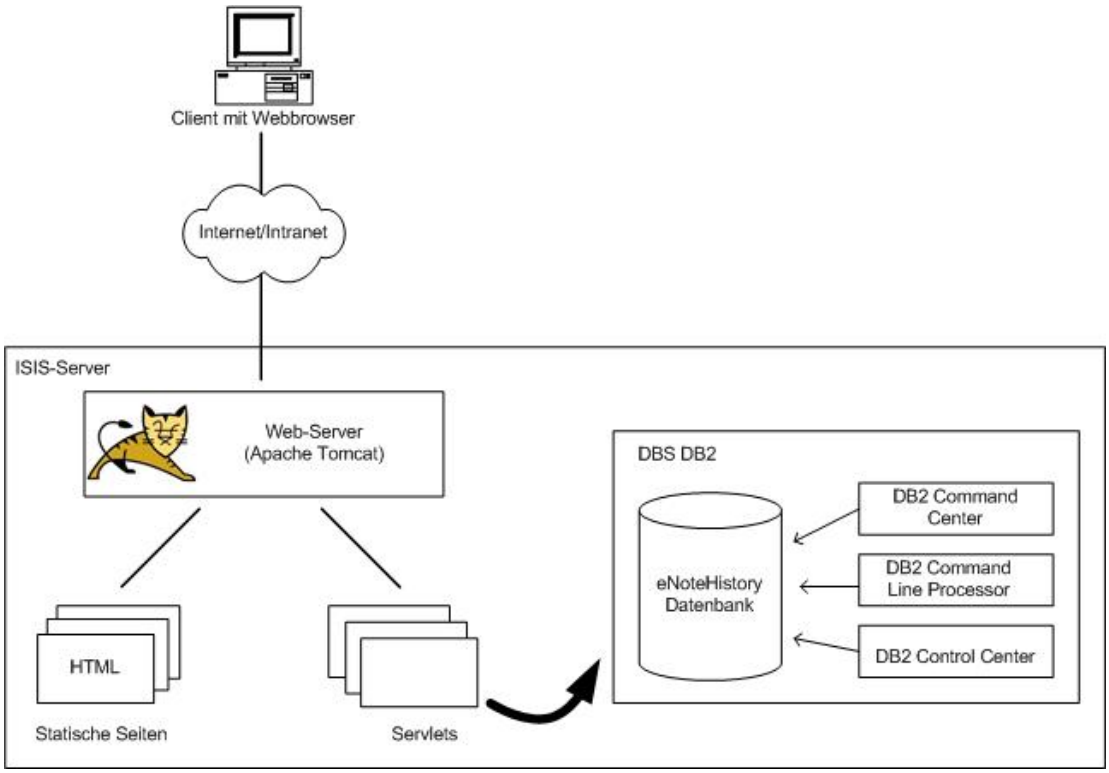


Abbildung 2.2: Systemarchitektur des eNoteHistory-Servers

In der Abbildung 2.2 ist die Architektur des Servers dargestellt. Die Daten befinden sich in der Datenbank, auf die man einen unbeschränkten Zugriff durch die zum Datenbanksystem DB2 zugehörigen Anwendungen hat. Die im Web-Server existierenden Servlets greifen nur beschränkt, abhängig von der Implementierung, auf die Daten zu. Nun werden beide Möglichkeiten des Datenzugriffs genauer vorgestellt.

2.2.1 Webinterface

Unter der Adresse *www.enotehistory.de* befindet sich die Webseiten des Projektes. Neben vielen Informationen über das Projekt, steht auch die manuelle Schreiberanalyse unter *www.enotehistory.de/analyse.html* zur Verfügung. Der Aufbau der Seiten, die Servlets, sowie auch das Benutzen der Analyseseiten wird im Kapitel 5 erklärt.

2.2.2 DB2 zugehörige Anwendungen

Mit dem Datenbanksystem DB2 werden eine Reihe von Anwendungen mitgeliefert, mit denen man Kontrolle über die Daten hat sowie das Datenbanksystem administrieren kann. Mit den Anwendungen DB2 Control Center, DB2 Command Center und DB2 Command Line Processor hat man direkten Zugriff auf die Daten in der Datenbank.

DB2 Control Center

Das Control Center kann zum Managen des Systems und für allgemeine administrative Aufgaben an den DB2 Instanzen, den einzelnen Datenbanken, den Datenbankobjekten und den Nutzer bzw. Nutzergruppen verwendet werden. Es bietet auch die Möglichkeit, die Datenbank remote zu administrieren und gibt Hilfestellung bei komplexeren Aufgaben. Mögliche ausführbare Aufgaben sind:

- Erstellen, Ändern und Löschen von Datenbanken, Tabellen, Sichten, Indexe, Triggers und Schematas
- Laden, Importieren und Exportieren von Daten
- Ausführen eines Back-Up bzw. Restore von Datenbanken
- Managen von Verbindungen zur Datenbank

Mit dem Control Center kann man auch die anderen Anwendungen zum Administrieren aufrufen.

DB2 Command Center

Mit dem Command Center kann man folgende Aufgaben erledigen:

- Ausführen von SQL-Statements und DB2-Kommandos und die Ansicht ihrer Resultate
- Benutzen des SQL Assist, um komplexe Anfragen zu formen

- Speichern von mehreren SQL-Statements und DB2-Kommandos in eine Skriptdatei

Wie im letzten Punkt zu sehen ist, können mehrere Kommandos und SQL-Statements zusammengefasst und ausgeführt werden. Im DB2 Command Line Processor kann man dagegen nur ein SQL-Statement bzw. Kommando eingeben und ausführen lassen.

DB2 Command Line Processor

Diese Anwendung ist textorientiert, welches bedeutet, dass der Benutzer die DB2-Kommandos, Systemkommandos und SQL-Statements direkt in einen Prompt eingibt. Es werden drei verschiedene Modi unterschieden:

- Im **Command mode** muss "db2" dem auszuführenden Kommando vorangestellt werden, und Kommandos mit speziellen Zeichen für das Betriebssystem müssen in Anführungszeichen eingeschlossen werden (Bsp.: db2 "select * from personen").
- Im **Interactive input mode** ist das Kommando *db2* bereits vorangestellt, was durch den Prompt "db2 =>" gekennzeichnet ist. Ansonsten gelten die gleichen Vorgaben wie im Command mode.
- Im **Batch mode** werden Kommandos ausgeführt, die in einer Batch-Datei (ASCII-Textdatei) gespeichert sind. Dazu wird das Kommando "db2 -f *dateiname*" eingegeben.

2.3 System- und Datensicherheit

Das Projekt eNoteHistory ist ein laufendes Projekt, welches einer ständigen Weiterentwicklung unterliegt. Damit ist eine tägliche Pflege und Wartung des Systems und des Servers verbunden. Um einen Verlust der Daten zu vermeiden, wird ein tägliches Backup auf Band gemacht, wenn eine Änderung des Datenbestandes erfolgte. Mit einer Neukonfiguration des Systems und einem Restore der Daten kann das System wieder hergestellt werden.

Kapitel 3

Datenmodell

Zur Verwaltung der eingescannten Notenblätter mit ihren Daten wurde ein DB2 Datenbank namens `enote` erstellt. In ihr werden nicht nur die Notenhandschriften gespeichert, sondern vorallem ihre zugehörigen bibliothekarischen Metadaten und die Merkmale der Schreibercharakteristik in Form von Feature-Vektoren.

Zur logischen Abgrenzung der Daten wurden für die Tabellendefinitionen der enote-Datenbank vier verschiedene Schemata verwendet.

DICT Das Schema DICT wird zur Definition von Tabellen verwendet, in denen Handschriftenmerkmale gespeichert werden. Sie repräsentieren die Feature-Base.

FEATURES Die Merkmale bzw. die Resultate von semi-automatischen Analysen werden in den Tabellen des Schemas FEATURES gespeichert.

METADATA Dieses Schema dient zur Definition von Tabellen zur Beschreibung der Quellen bzw. zur Speicherung der Bibliotheksdaten und der digitalisierten Notenhandschriften.

IPFV Die Merkmale der Notenhandschriften (insbesondere der Notenköpfe und Notenzeilen) werden in den Tabellen des Schemas IPFV gespeichert.

Im Folgenden werden die einzelnen Tabellen in der Datenbank genauer vorgestellt. Die Attributnamen in fettgedruckter Schrift kennzeichnen den Primärschlüssel einer Tabelle.

3.1 Datenbankschema DICT

Im Abschnitt 1.3 wurde die Feature Base vorgestellt, dessen Baumstruktur die Charakteristik der Notenhandschriften darstellt. Die folgenden Tabellen im Datenbankschema DICT repräsentieren diese Baumstruktur:

- **Node_Type** gibt an, welche verschiedenen Typen von Knoten (Nodes) es gibt (3.1.1)
- **Nodes** beinhaltet Daten über die Merkmale von Handschriften (3.1.2)
- **Distances** speichert die Distanz (Abstand) zwischen zwei Knoten bzw. zwischen zwei Handschriftenmerkmalen (3.1.3)

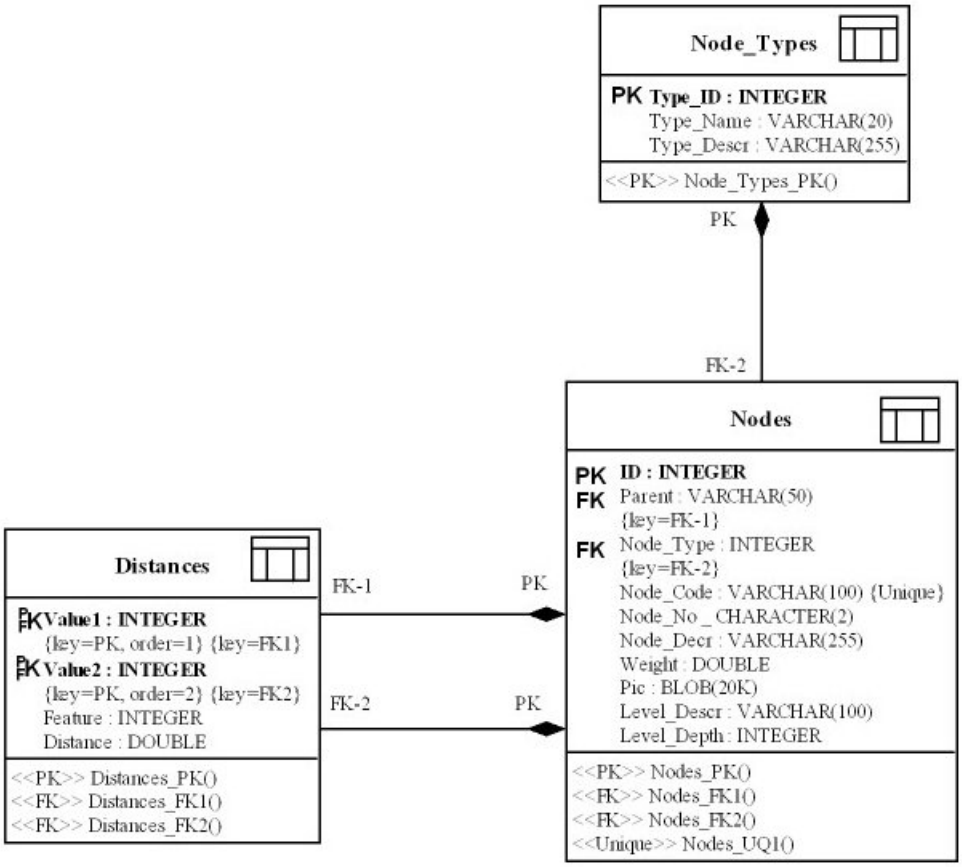


Abbildung 3.1: Relationales Datenmodell des Schemas DICT

3.1.1 Tabelle Node_Type

Jeder Knoten in der Feature Base ist von einem bestimmten Typ, welche in dieser Tabelle gespeichert werden. Mögliche Typen sind `empty, feature prefix, feature, value` und `value prefix`. In der Tabelle `Nodes` wird durch ein Attribut als Fremdschlüssel auf diese Tabelle `Node_Type` verwiesen, und somit jedem Knoten ein Knotentyp zugeordnet.

Attributname	Beschreibung	Beispielwert
Type_id	ID als Primärschlüssel	
Type_name	Benennung des jeweiligen Knotentyps	Zulässige Werte sind: *empty, feature prefix, feature, value, value prefix*
Type_description	ergänzende Beschreibung des jeweiligen Knotentyps	*value*: a selectable value *value prefix*: a non-selectable value

Tabelle 3.1: Beschreibungen von Node_Type

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Index	Suche
Type_id	integer	+	+	-	+	+	Attr
Type_name	varchar(20)	+	-	-	-	+	Attr
Type_description	varchar(100)	-	-	-	-	-	FullText

Tabelle 3.2: Node_Type

3.1.2 Tabelle Nodes

Diese Tabelle speichert die Knoten der Feature Base. Jedem Knoten wird ein Typ (Fremdschlüssel auf die Tabelle `Node_Type`) und einen Vaterknoten (Verweis auf die gleiche Tabelle) zugeordnet.

Attributname	Beschreibung	Beispielwert
ID	Primärschlüssel	
Parent	Umsetzung der Baumstruktur, die Menge aller Knoten mit dem gleichen Vater werden Level genannt; Fremdschlüssel auf Nodes.ID	vorhandene ID in dieser Tabelle `Nodes`: 1
Node_Code	Punktnotation des Knotens	1.1.
Node_No	Position des Knotens im Level, entspricht der letzten Zahl in der Punktnotation	1
Node_Descr	Name und Beschreibung	G-Schlüssel
Weight	Gewicht der Knoten für die Kalkulation der Abstandsfunktion	0.34546
Pic	Bild als BLOB gespeichert	
Level_Descr	Beschreibung des Levels, zu dem die untergeordneten Knoten zählen	?
Level_Depth	Baumtiefe, beginnend bei Wurzel = 0	1 (ein Knoten unter der Wurzel)
Node_Type	bezeichnet den Knotentypen, ist Fremdschlüssel auf Node_Type.Type_id (0, 1, 2, 3, 4) 0 - Prefix, 1 - Feature, 2 - Value etc.	siehe Tabelle `Node_Type`

Tabelle 3.3: Beschreibung von Nodes

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Index	Suche
ID	integer	+	+	-	+	+	Attr
Parent	integer	-	-	+	-	+	Attr
Node_Code	varchar(100)	+	-	-	+	+	FullText
Node_No	char(2)	+	-	-	-	-	Attr
Node_Descr	varchar(255)	-	-	-	-	-	FullText
Weight	double	-	-	-	-	-	Attr
Pic	BLOB 20K	-	-	-	-	-	
Level_Descr	varchar(100)	-	-	-	-	-	FullText
Level_Depth	integer	-	-	-	-	-	Attr
Node_Type	integer	-	-	+	-	-	Attr

Tabelle 3.4: Nodes

3.1.3 Tabelle Distances

In der Tabelle `Distances` werden die Distanzmatrixen gespeichert, die zur Berechnung der Abstände zwischen den Wertepaare verwendet werden. Wenn kein Wert für zwei Knotenpaare in dieser Tabelle enthalten ist, dann ist die Distanz gleich 1.

Attributname	Beschreibung	Beispielwert
Value1	Feature Value ID 1 - verweist auf einen Feature-Value-Knoten in der Tabelle Nodes	2.1.1
Value2	Feature Value ID 2 - verweist auf einen Feature-Value-Knoten in der Tabelle Nodes	2.1.2.
Feature	Feature ID - verweist auf einen Feature-Knoten in der Tabelle Nodes	2.1.
Distance	Ein skalarer Wert, der den Abstand (Ähnlichkeitsmaß) zwischen zwei Feature-Value-Knoten repräsentiert	0.5

Tabelle 3.5: Beschreibung von Distances

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Index	Suche
Value1	integer	+	+	+	-	+	Attr
Value2	integer	+	+	+	-	+	Attr
Feature	integer	+	-	+	-	+	Attr
Distance	double	+	-	-	-	-	Attr

Tabelle 3.6: Distances

3.2 Datenbankschema FEATURES

Das Schema FEATURES beinhaltet Tabellen, siehe Abbildung 3.2, welche die Ergebnisse von der Handschriftenanalyse speichern. Das Ergebnis jeder Handschriftenanalyse ist ein Feature-Vektor, der eine ausgewählte Menge von Handschriftenmerkmalen vereint. Ein Feature-Vektor kann zu einem oder auch mehreren Musik-Manuscripten und auch ein oder mehreren Scribes zugeordnet werden. In dem Datenbankschema FEATURES sind folgende Tabellen definiert:

- **Featurevectors** repräsentiert die Feature-Vektoren (3.2.1)
- **MMS_FV** speichert die Zuordnungen zwischen Feature-Vektoren und den Musik-Manuscripten (3.2.2)
- **FVvalues** speichert die Werte eines Features und ordnet sie zu den Feature-Vektoren zu (3.2.3)

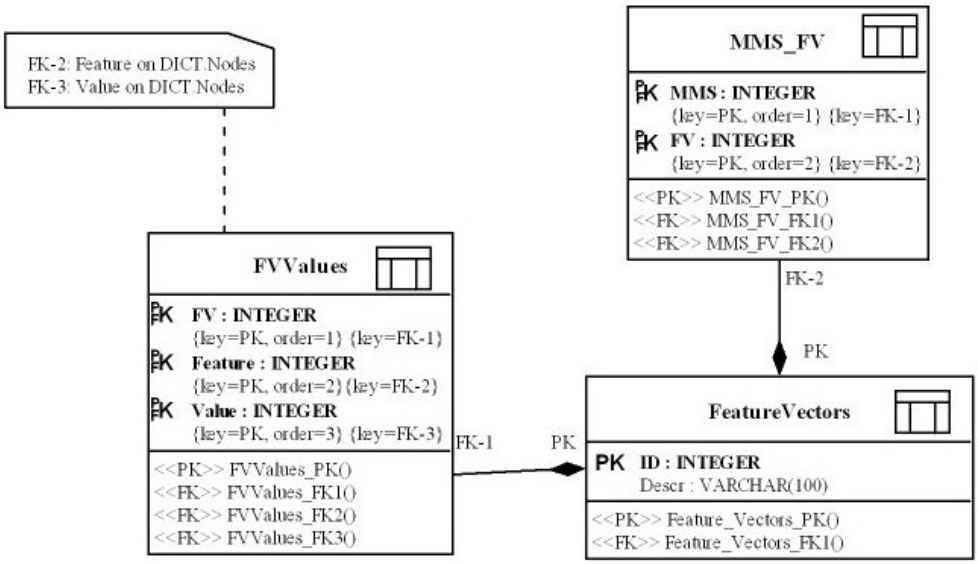


Abbildung 3.2: Relationales Datenmodell vom Schema `Feature`

3.2.1 Tabelle FeatureVectors

In der Tabelle `FeatureVectors` werden allgemeine Informationen zur Schreibercharateristik bzw. zum Feature-Vektor gespeichert. Die eigentlichen Werte eines Feature-Vektors werden in der Tabelle`FVValues` gespeichert, in welcher der Feature-Vektor dieser Tabelle referenziert wird.

Attributname	Beschreibung	Beispielwert
ID	Primärschlüssel	
Descr	Beschreibung des Feature-Vektors	Schriftstadium 1

Tabelle 3.7: Beschreibungen von Featurevectors

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
ID	integer	+	+	-	+	Attr
Descr	varchar(100)	-	-	-	-	FullText

Tabelle 3.8: Featurevectors

3.2.2 Tabelle MMS_FV

Die Tabelle `MMS_FV` speichert die Beziehungen (n:m) zwischen den Feature-Vektoren und den Musik-Manuscripten. Ein Feature-Vektor kann zu mehreren Musik-Manuscripten zugeordnet werden. Umgekehrt können zu einem Musik-Manuscript mehrere Feature-Vektoren gehören, da Musikwissenschaftler verschiedener Meinungen sein können bzw. mit der Zeit man sich auf eine andere Schreibercharakteristik einigt, die dann zusätzlich gespeichert wird.

Attributname	Beschreibung	Beispielwert
MMS	Referenziert auf ein Musik-Manuscript; Fremdschlüssel auf die Tabelle `METADATA.Music_Manuscript_Scribe`	
FV	Referenziert einen Feature-Vektor; Fremdschlüssel auf die Tabelle `FEATURE.Featurevectors`	

Tabelle 3.9: Beschreibungen von MMS_FV

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
MMS	integer	+	+	+	-	Attr
FV	integer	+	+	+	-	FullText

Tabelle 3.10: MMS_FV

3.2.3 Tabelle FVValues

Die Tabelle `FVValues` speichert die Werte zu den Features im Feature-Vektor. Dabei wird der entsprechende Feature-Vektor referenziert, und dazu das Feature und seinen Wert gespeichert.

Attributname	Beschreibung	Beispielwert
FV	Referenziert einen Feature-Vektor; Fremdschlüssel auf der Tabelle `FEATURE.Featurevectors`	2
Feature	Referenziert einen Feature-Knoten; Fremdschlüssel auf der Tabelle `DICT.Nodes`	2.2.
Value	Referenziert einen Feature-Value-Knoten; Fremdschlüssel auf der Tabelle `DICT.Nodes`	2.2.1.3

Tabelle 3.11: Beschreibungen von FVValues

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
FV	integer	+	+	+	-	Attr
Feature	integer	+	+	+	-	Attr
Value	integer	+	+	+	-	Attr

Tabelle 3.12: FVValues

3.3 Datenbankschema METADATA

In diesem Schema sind Tabellen definiert, die Daten zu den Notenhandschriften speichern, wie zum Beispiel: die Bilddateien, allgemeine Informationen zu den einzelnen Notenblättern aus dem Bibliothekskatalog, die Komponisten und die Kopisten.

In dem Datenbankschema METADATA sind folgende Tabellen definiert:

- **Music_Manuscript** (3.3.1)
- **Music_Works_in_Manuscripts** (3.3.2)
- **Music_Works_Composers** (3.3.3)
- **Music_Works_Text_Authors** (3.3.4)
- **Incipit** (3.3.5)
- **Manuscript_Section** (3.3.6)
- **Music_Manuscript_Page** (3.3.7)
- **Page_Images** (3.3.8)
- **Music_Manuscript_Scribe** (3.3.9)
- **Libraries** (3.3.10)
- **Music_Scores_Collections** (3.3.11)
- **Scribes** (3.3.12)
- **Music_Works** (3.3.13)
- **Text_Authors** (3.3.14)
- **Composers** (3.3.15)
- **Incipit_Types** (3.3.16)
- **Tones** (3.3.17)
- **Roles** (3.3.18)
- **Section_Types** (3.3.19)

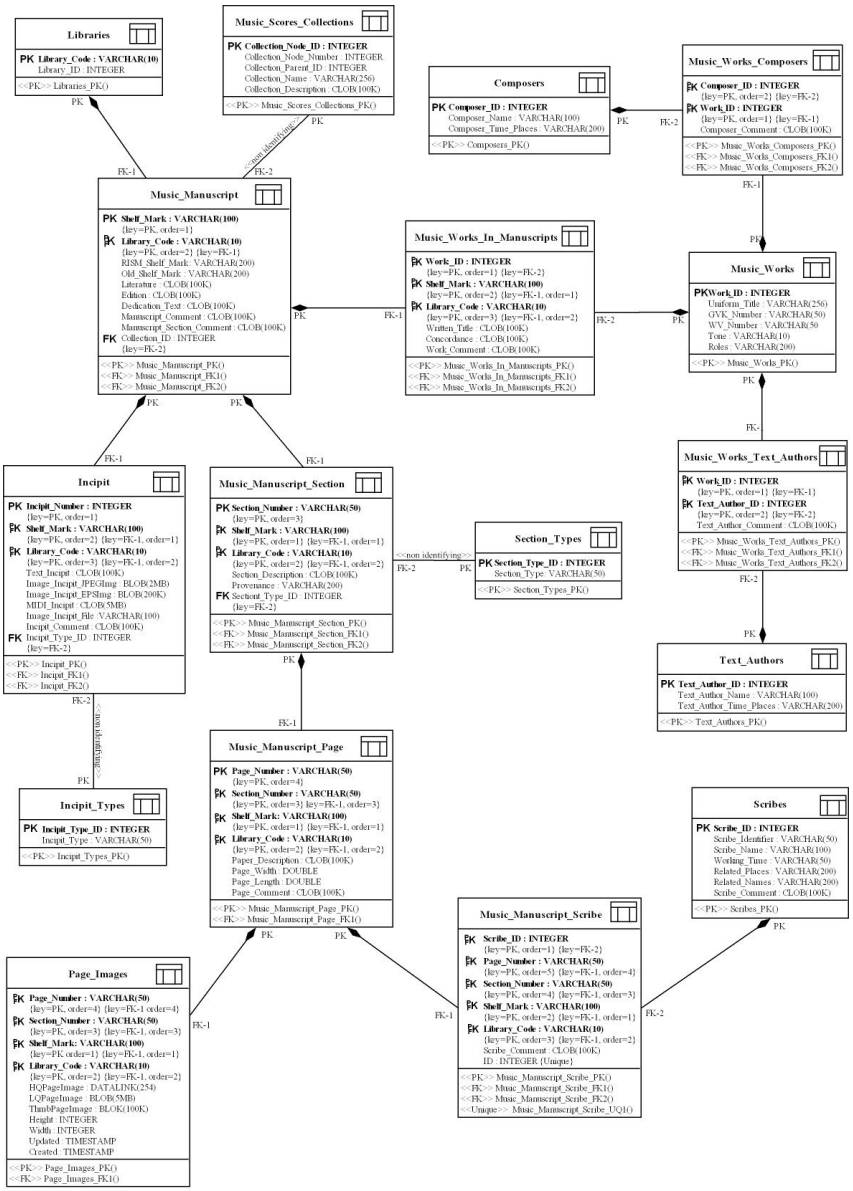


Abbildung 3.3: Relationales Datenmodell des Schemas Metadata

3.3.1 Tabelle Music_Manuscript

In dieser Tabelle werden allgemeine bibliothekarische Informationen zu den Musik-Manuscripten gespeichert. Dabei gilt das RISM-Siegel der Bibliothek als Schlüssel des Musik-Manuscripts, welches die Bibliothek und die Bibliothekssignatur der Notenhandschrift eindeutig kennzeichnet.

Attributname	Beschreibung	Beispielwert
library_code	RISM-Siegel der Bibliothek	D-ROu
shelf_mark	Bibliothekssignatur der Notenhandschrift	Musica Saec. XVIII.-57.5
RSIM_Shelf_Mark	RISM-Signatur der Notenhandschrift	A/II: 001.567.311
Old_Shelf_Mark	Alte Bibliothekssignatur der Notenhandschrift	
Collection_ID	Identifikator der Kollektion, zu der die Notenhandschrift gehört; siehe Tabelle "Music_Score_Collections"	
Literature	Literaturverweise	
Edition	Edition - Ort, Verlag etc.	
Dedication_Text	Widmung	
Manuscript_Comment	Anmerkungen zur Handschrift	Verm. Hamburger Kopist (2. Hälfte 18. Jh.); Schriftstadium 1, vgl. E. Krüger 2002, Bd. 2, S. 648-652
Manuscript_Section_Comment	Allgemeine Beschreibungen zum Notenhandschriftenteil	

Tabelle 3.13: Beschreibungen von Music_Manuscript

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
library_code	varchar(10)	+	+	-	+	Attr
shelf_mark	varchar(100)	+	+	+	+	Attr/ Full-Text
RSIM_Shelf_Mark	varchar(200)	-	-	-	-	Attr
Old_Shelf_Mark	varchar(200)	-	-	-	-	Attr
Collection_ID	integer	-	-	+	-	Attr
Literature	text, 100K	-	-	-	-	FullText
Edition	text, 100K	-	-	-	-	FullText
Dedication_Text	text, 100K	-	-	-	-	FullText
Manuscript_Comment	text, 100K	-	-	-	-	FullText
Manuscript_Section_Comment	text, 100K	-	-	-	-	FullText

Tabelle 3.14: Music_Manuscript

3.3.2 Tabelle Music_Works_in_Manuscripts

Die Tabelle beinhaltet die einzelnen Musikstücke, die in einem Musik-Manuscript vorkommen, inklusive den Titel und die dazu gehörigen Bemerkungen. Andere Informationen werden in anderen Tabellen gespeichert und per Fremdschlüssel referenziert.

Attributname	Beschreibung	Beispielwert
library_code	RISM-Siegel der Bibliothek	D-ROu
shelf_mark	Bibliothekssignatur der Notenhandschrift	Musica Saec. XVIII.-57.5
Work_ID	Identifikator des Werkes; siehe Tabelle "Music_Works"	
Written_Title	Titel des Werkes, genau so wie es in der Notenhandschrift steht	
Concordance	Konkordanzen	
Work_Comment	Anmerkungen zum Werk der Notenhandschrift	

Tabelle 3.15: Beschreibungen von Music_Works_in_Mansucripts

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
library_code	varchar(10)	+	+	+	+	Attr
shelf_mark	varchar(100)	+	+	+	+	Attr
Work_ID	integer	+	+	+	+	Attr
Written_Title	Text, 100K	-	-	-	-	Attr
Concordance	Text, 100K	-	-	-	-	Attr
Work_Comment	Text, 100K	-	-	-	-	Attr

Tabelle 3.16: Music_Works_in_Mansucripts

3.3.3 Tabelle Music_Works_Composers

In der Tabelle Music_Works_Composers werden die Zuordnungen zwischen Komponist und Musikstück gespeichert.

Attributname	Beschreibung	Beispielwert
Composer_ID	Identifikator des Komponisten; siehe Tabelle "Composers"	
Work_ID	Identifikator des Werkes; siehe Tabelle "Music_Works"	
Composer_Comment	Anmerkungen zum Komponisten	

Tabelle 3.17: Beschreibungen von Composer

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Composer_ID	integer	+	+	+	+	Attr
Work_ID	integer	+	+	+	+	Attr
Composer_Comment	Text, 100K	-	-	-	-	Attr

Tabelle 3.18: Composer

3.3.4 Tabelle Music_Works_Text_Authors

Diese Tabelle speichert die Zuordnungen zwischen Musikstück und Textautor, die den Text zum Musikstück geschrieben haben.

Attributname	Beschreibung	Beispielwert
Text_Author_ID	Identifikator des Textautors; siehe Tabelle "Text_Authors"	
Work_ID	Identifikator des Werkes; siehe Tabelle "Music_Works"	
Text_Author_Comment	Anmerkungen zum Textautor	

Tabelle 3.19: Beschreibungen von Music_Works_Text_Authors

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Text_Author_ID	integer	+	+	+	+	Attr
Work_ID	integer	+	+	+	+	Attr
Text_Author_Comment	Text	-	-	-	-	Attr

Tabelle 3.20: Text_Author

3.3.5 Tabelle Incipit

Die Tabelle speichert die ersten Noten, mit denen ein Musikstück beginnt.

Attributname	Beschreibung	Beispielwert
Library_Code	RISM-Siegel der Bibliothek	D-ROu
Shelf_Mark	Bibliothekssignatur der Notenhandschrift	Musica Saec. XVIII.-57.5
Incipit_Number	Laufende Nummer des Inzipit der Notenhandschrift	
Incipit_Type_ID	Identifikator des Inzipit-Typs; siehe Tabelle "Incipit_Types"	
Incipit_Comment	Anmerkungen zum Inzipit	
Image_Incipit_File	Name der Inzipit-Datei	
Image_Incipit_EPSImg	Bild der Noten des Inzipit in .EPS Format	
Image_Incipit_JPEGImg	Bild der Noten des Inzipit in .JPEG Format	
Text_Incipit	Textuelle Beschreibung des Inzipit	
MIDI_Incipit	Darstellung des Inzipit in MIDI Format	

Tabelle 3.21: Beschreibungen von Incipit

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Library_Code	varchar(10)	+	+	+	+	Attr
Shelf_Mark	varchar(100)	+	+	+	+	Attr
Incipit_Number	integer	+	+	+	+	Attr
Incipit_Type_ID	integer	-	-	-	+	Attr
Incipit_Comment	Text, 100 K	-	-	-	-	FullText
Image_Incipit_File	varchar(100)	-	-	-	-	Attr
Image_Incipit_EPSImg	Image, 200 K	-	-	-	-	-
Image_Incipit_JPEGImg	Image, 2M	-	-	-	-	-
Text_Incipit	Text, 100 K	-	-	-	-	FullText
MIDI_Incipit	Text, 5M	-	-	-	-	Attr/FullText

Tabelle 3.22: Incipit

3.3.6 Tabelle Manuscript_Section

Die Tabelle speichert Informationen zu den Notenhandschriftenteilen, wie eine Beschreibung, Entstehungszeit und -ort.

Attributname	Beschreibung	Beispielwert
Library_Code	RISM-Siegel der Bibliothek	D-ROu
Shelf_Mark	Bibliothekssignatur der Notenhandschrift	Musica Saec. XVIII.-57.5
Section_Number	Laufende Nummer des Notenhandschriftenteils	
Section_Type_ID	Identifikator des Typs des Notenhandschriftenteils; siehe Tabelle "Section_Types"	
Section_Description	Textuelle Beschreibung des Notenhandschriftenteils	Einband mit Papier bezogen; auf dem Vorderdeckel goldgeprägtes Etikett (90x140): LA PASSIONE DI GESU CRISTO SIGNOR NOSTRO, Altsignatur: VI.IV.78., im vorderen Innendeckel Exlibris mit mecklenburgisch-württenbergischem Allianzwappen, 272 S., 225x290
Provenance	Ort und Zeit des Entstehens der Handschrift	Hamburg (Schwerin/Ludwigslust)

Tabelle 3.23: Beschreibungen von Manuscript_Section

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Library_Code	varchar(10)	+	+	+	+	Attr
Shelf_Mark	varchar(100)	+	+	+	+	Attr
Section_Number	varchar(50)	+	+	+	+	Attr
Section_Type_ID	integer	-	-	+	-	Attr
Section_Description	Text, 100 K	-	-	-	-	FullText
Provenance	varchar(200)	-	-	-	-	FullText

Tabelle 3.24: Manuscript_Section

3.3.7 Tabelle Music_Manuscript_Page

In dieser Tabelle werden die Merkmale zu einer Seite gespeichert. Dazu gehören die Eigenschaften des Papiers, wie Höhe, Breite und Format.

Attributname	Beschreibung	Beispielwert
Library_Code	RISM-Siegel der Bibliothek	D-ROu
Shelf_Mark	Bibliothekssignatur der Notenhandschrift	Musica Saec. XVIII.-57.5
Section_Number	Laufende Nummer des Notenhandschriftenteils	
Page_Number	Seitenzahl der Notenhandschrift; entspricht ein Teil des Dateinamens des Digitalisats	Bcp165 (Datei-Name - XVII.18.-11â[Bcp165].tif)
Paper_Description	Textuelle Beschreibung der Eigenschaften und Merkmale des Papiers	a) dreitürmige Torburg, im Tor doppelrandiger Schild mit aufrechtem Löwen b) JCB; DBSM: I 286; IPH D5
Page_Width	Breite der Seite (des Papiers)	240
Page_Height	Höhe der Seite (des Papiers)	350
Page_Comment	Anmerkungen zum Papier, Seite der Notenhandschrift	

Tabelle 3.25: Beschreibungen von Music_Manuscript_Page

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Library_Code	varchar(10)	+	+	+	+	Attr
Shelf_Mark	varchar(100)	+	+	+	+	Attr
Section_Number	varchar(50)	+	+	+	+	Attr
Page_Number	varchar(50)	+	+	-	+	Attr
Paper_Description	Text	-	-	-	-	Attr
Page_Width	Float	-	-	-	-	Attr
Page_Height	Float	-	-	-	-	Attr
Page_Comment	Text, 100 K	-	-	-	-	Attr

Tabelle 3.26: Music_Manuscript_Page

3.3.8 Tabelle Page_Images

Die Tabelle beinhaltet die eingescannten Bilder von den Notenhandschriften, sowie Informationen darüber.

Attributname	Beschreibung	Beispielwert
library_code	RISM-Siegel der Bibliothek	D-ROu
shelf_mark	Bibliothekssignatur der Notenhandschrift	Musica Saec. XVIII.-57.5
section_number	Laufende Nummer des Notenhandschriftenteils	
page_number	Seitenzahl der Notenhandschrift; entspricht ein Teil des Dateinamens des Digitalisats	Bcp165 (Datei-Name - XVII.18.-11Â[Bcp165].tif)
HQPageImage	DATALINK Item - eine Referenz zu der gespeicherten Seite im Digitalisat (Die Datei ist in TIF Format)	
LQPageImage	Eine komprimierte Version des Digitalisats in .JPEG Format, gespeichert als BLOB	
THMBPageImage	Thumbnail des Digitalisats	
updated	Datum der letzten Änderung des Digitalisats	
created	Datum der ersten Speicherung des Digitalisats	
height	Höhe des Bildes in Pixel	
width	Breite des Bildes in Pixeln	

Tabelle 3.27: Beschreibungen von Page_Images

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
library_code	varchar(10)	+	+	+	+	Attr
shelf_mark	varchar(100)	+	+	+	+	Attr
section_number	varchar(50)	+	+	+	+	Attr
page_number	varchar(50)	+	+	+	+	Attr
HQPageImage	Datalink	+	-	-	+	Attr
LQPageImage	Image	-	-	-	-	-
THMBPageImage	Image	-	-	-	-	-
updated	timestamp	-	-	-	-	Attr
created	timestamp	-	-	-	-	Attr
height	integer	-	-	-	-	Attr
width	integer	-	-	-	-	Attr

Tabelle 3.28: Page_Images

3.3.9 Tabelle Music_Manuscript_Scribe

Die Tabelle speichert die Zuordnungen von den Seiten eines Musik-Manuscripts zu den Schreibern. Das Attribut "ID" entstand wurde später zusätzlich eingeführt, und kann als Primärschlüssel (ist es aber nicht!) dieser Tabelle betrachtet werden, da es immer eindeutig für ein Tupel ist. Dieses Attribut wird als Referenz in der Tabelle Feature.MMS_FV verwendet, welche einen Feature-Vektor zu der Seite und dem Schreiber zuordnet.

Attributname	Beschreibung	Beispielwert
Scribe_ID	Identifikator des Schreibers; siehe Tabelle "Scribes"	
Library_Code	RISM-Siegel der Bibliothek	D-ROu
Shelf_Mark	Bibliothekssignatur der Notenhandschrift	Musica Saec. XVIII.-57.5
Section_Number	Laufende Nummer des Notenhandschriftenteils	
Page_Number	Seitenzahl der Notenhandschrift; entspricht ein Teil des Dateinamens des Digitalisats	Bcp165 (Datei-Name - XVII.18.-11$\hat{4}$[Bcp165].tif)
ID	Laufende Nummer, könnte als Primärschlüssel für diese Tabelle angesehen werden, wird zum Referenzieren eines Tupels in dieser Tabelle genutzt	
Scribe_Comment	Anmerkungen zum Schreiber der Notenhandschrift	

Tabelle 3.29: Beschreibungen von Music_Manuscript_Scribe

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Scribe_ID	integer	+	+	+	+	Attr
Library_Code	varchar(10)	+	+	+	+	Attr
Shelf_Mark	varchar(100)	+	+	+	+	Attr
Section_Number	varchar(50)	+	+	+	+	Attr
Page_Number	varchar(50)	+	+	+	+	Attr
ID	integer	+	-	-	+	Attr
Scribe_Comment	Text, 100 K	-	-	-	-	Attr

Tabelle 3.30: Music_Manuscript_Scribe

3.3.10 Tabelle Libraries

Diese Tabelle beinhaltet die Bibliotheken, die durch ihr RISM-Siegel eindeutig gekennzeichnet werden.

Attributname	Beschreibung	Beispielwert
Library_Code	RISM-Siegel der Bibliothek	D-ROu
Library_ID	Identifikator der Bibliothek - Laufende Nummer	28

Tabelle 3.31: Beschreibungen von Libraries

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Library_Code	varchar(10)	+	+	-	+	Attr
Library_ID	integer	-	-	-	-	Attr

Tabelle 3.32: Libraries

3.3.11 Tabelle Music_Scores_Collections

Die Tabelle speichert Informationen zu den Kollektionen von Musik-Manuscripten. Die Kollektion "Beschreibender Katalog" enthält Handschriften, Werke und Werkgruppen von einzelnen Komponisten. Die Kollektion "Anonymus" enthält Handschriften und Werke von nicht identifizierten Komponisten.

Attributname	Beschreibung	Beispielwert
Collection_Node_ID	Identifikator der Notenhandschriftenkollektion - Laufende Nummer	
Collection_Node_Number	Knotenkennzeichen der Kollektion in der Klassifikationshierarchie	2.1
Collection_Parent_ID	Identifikator der übergeordneten Kollektion in der Klassifikationshierarchie	
Collection_Name	Name der Kollektion	Klavierbücher
Collection_Description	Beschreibung der Kollektion	

Tabelle 3.33: Beschreibungen von Music_Scores_Collections

Attributname	Format	NOT NULL	Primärschlüssel	Fremdschlüssel	Unique	Suche
Collection_Node_ID	integer	+	+	-	+	Attr
Collection_Node_Number	integer	+	-	-	-	Attr
Collection_Parent_ID	integer	+	-	-	-	Attr
Collection_Name	varchar(256)	+	-	-	-	Attr/FullText
Collection_Description	Text, 100 K	-	-	-	-	Attr/FullText

Tabelle 3.34: Music_Scores_Collections

3.3.12 Tabelle Scribes

Diese Tabelle speichert die Schreiber inklusive Informationen über ihre Arbeitsperiode, Aufenthaltsorte und Kontaktpersonen.

Attributname	Beschreibung	Beispielwert
Scribe_ID	Identifikator des Schreibers - Laufende Nummer	
Scribe_Identifier	Eindeutiger, lesbarer (referenzierbarer) Identifikator des Schreibers	An305 (Kast 1958)
Scribe_Name	Name des Schreibers	
Working_Time	Arbeitsperiode des Schreibers	1750
Related_Places	in Beziehung stehende Ortsnamen	Leipzig
Related_Names	in Beziehung stehende Personennamen	C.P.E. Bach, J. Schuback

Tabelle 3.35: Beschreibungen von Scribes

Attributname	Format	NOT NULL	Primärschlüssel	Fremdschlüssel	Unique	Suche
Scribe_ID	integer	+	+	-	+	Attr
Scribe_Identifier	varchar(50)	-	-	-	-	Attr
Scribe_Name	varchar(100)	-	-	-	-	Attr/FullText
Working_Time	varchar(50)	-	-	-	-	FullText
Related_Places	varchar(200)	-	-	-	-	FullText
Related_Names	varchar(200)	-	-	-	-	FullText

Tabelle 3.36: Scribes

3.3.13 Tabelle Music_Works

Die Tabelle speichert allgemeine Informationen zu einem Musikstück, wie Titel, Instrumente und Tonart.

Attributname	Beschreibung	Beispielwert
Work_ID	Identifikator des Werkes - Laufende Nummer	
Uniform_Title	Einheitlicher Titel des Werkes	La Passione di Gesu Cristo Signor Nostro
GVK_Number	Nummer des Werkes im "Gemeinsamer Verbundkatalog"	
WV_Number	Nummer des "Werkeverzeichnisses" der Deutschen Bibliothek	Schuback-WV Nr. 68a
Roles	Besetzung, Liste der Instrumente	Soli SATB - 2 Fl.trav, 2 Ob, 2 Cor, 2 Vl, Va, Bc
Tone	Tonart des Werkes	

Tabelle 3.37: Beschreibungen von Music_Works

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Work_ID	integer	+	+	-	+	Attr
Uniform_Title	varchar(256)	-	-	-	-	Attr/FullText
GVK_Number	varchar(50)	-	-	-	-	Attr
WV_Number	varchar(50)	-	-	-	-	Attr
Roles	varchar(200)	-	-	-	-	FullText
Tone	varchar(10)	-	-	-	-	FullText

Tabelle 3.38: Music_Works

3.3.14 Tabelle Text_Authors

In der Tabelle sind die Textautoren mit Namen und ihren Aufenthaltsorten gespeichert.

Attributname	Beschreibung	Beispielwert
Text_Author_ID	Identifikator des Textautors; Laufende Nummer	
Text_Author_Name	Namen des Textautors	Pietro Metastasio
Text_Author_Time_Places	in Beziehung stehende Orts- und Personennamen	

Tabelle 3.39: Beschreibungen von Text_Authors

Attributname	Format	NOT NULL	Primärschlüssel	Fremdschlüssel	Unique	Suche
Text_Author_ID	integer	+	+	-	+	Attr
Text_Author_Name	varchar(100)	-	-	-	-	Attr/FullText
Text_Author_Time_Places	varchar(200)	-	-	-	-	Attr/FullText

Tabelle 3.40: Text_Authors

3.3.15 Tabelle Composers

Die Tabelle beinhaltet die Komponisten mit Namen und ihren Aufenthaltsorten.

Attributname	Beschreibung	Beispielwert
Composer_ID	Identifikator des Komponisten; Laufende Nummer	
Composer_Name	Namen des Komponisten	Jacob Schuback
Composer_Time_Places	in Beziehung stehende Orts- und Personennamen	

Tabelle 3.41: Beschreibungen von Composers

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Composer_ID	integer	+	+	-	+	Attr
Composer_Name	varchar(100)	-	-	-	-	Attr/FullText
Composer_Time_Places	varchar(200)	-	-	-	-	Attr/FullText

Tabelle 3.42: Composers

3.3.16 Tabelle Incipit_Types

In dieser Tabelle werden die Typen von Incipits (erste Noten am Anfang des Musikstücks) gespeichert. Momentan existieren die Typen “Content” und “Paper Description”.

Attributname	Beschreibung	Beispielwert
Incipit_Type_ID	Identifikator des Inzipit-Typs	
Incipit_Type	Inzipit-Typ Name	

Tabelle 3.43: Beschreibungen von Incipit_Types

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Incipit_Type_ID	integer	+	+	-	+	Attr
Incipit_Type	varchar(50)	-	-	-	-	Attr

Tabelle 3.44: Incipit_Types

3.3.17 Tabelle Tones

In dieser Tabelle werden die Tonarten von den Musikstücken gespeichert. Es existieren die Tonarten A, B, C, D, E, F und G.

Attributname	Beschreibung	Beispielwert
Tone_ID	Identifikator der Tonart	
Tone	Name der Tonart	

Tabelle 3.45: Beschreibungen von Tones

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Tone_ID	integer	+	+	-	+	Attr
Tone	varchar(10)	-	-	-	-	Attr

Tabelle 3.46: Tones

3.3.18 Tabelle Roles

Diese Tabelle beinhaltet die Musikinstrumente, die für die Musikstücke benötigt werden.

Attributname	Beschreibung	Beispielwert
Role_ID	Identifikator des Musikinstruments	
Role	Name des Musikinstruments	
Description	Beschreibung des Musikinstruments	

Tabelle 3.47: Beschreibungen von Roles

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Role_ID	integer	+	+	-	+	Attr
Role	varchar(200)	-	-	-	-	Attr
Description	varchar(255)	-	-	-	-	Attr

Tabelle 3.48: Roles

3.3.19 Tabelle Section_Types

Diese Tabelle speichert die möglichen Typen von Teilen einer Notenhandschrift. Momentan existieren die folgenden Typen: Partitur, Partiturfragment, Stimme und Umschlag.

Attributname	Beschreibung	Beispielwert
Section_Type_ID	Identifikator des Typs eines Notenhandschriftenteils	
Section_Type	Typname des Notenhandschriftenteils	

Tabelle 3.49: Beschreibungen von Section_Types

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Section_Type_ID	integer	+	+	-	+	Attr
Section_Type	varchar(50)	-	-	-	-	Attr

Tabelle 3.50: Section_Types

3.4 Datenbankschema IPFV

In diesem Datenbankschema sind die folgenden Tabellen definiert, welche die Komponenten des Notensystems, wie Notenlinien, Taktstriche, Notenköpfe und Notenhälse speichert:

- **Page_Image_ROI** speichert die zu analysierenden Bereiche auf dem Notenblatt (3.4.1)
- **Staff_Lines** beinhaltet die Notenlinien auf dem Notenblatt (3.4.2)
- **Note_Head** speichert die Notenköpfe des Notenblattes (3.4.3)
- **Note_Stem** speichert die Notenhälses auf dem Notenblatt (3.4.4)
- **Bar_Lines** beinhaltet die Taktstriche des Notenblattes (3.4.5)

Das relationale Datenmodell ist in der Abbildung 3.4 zu sehen.

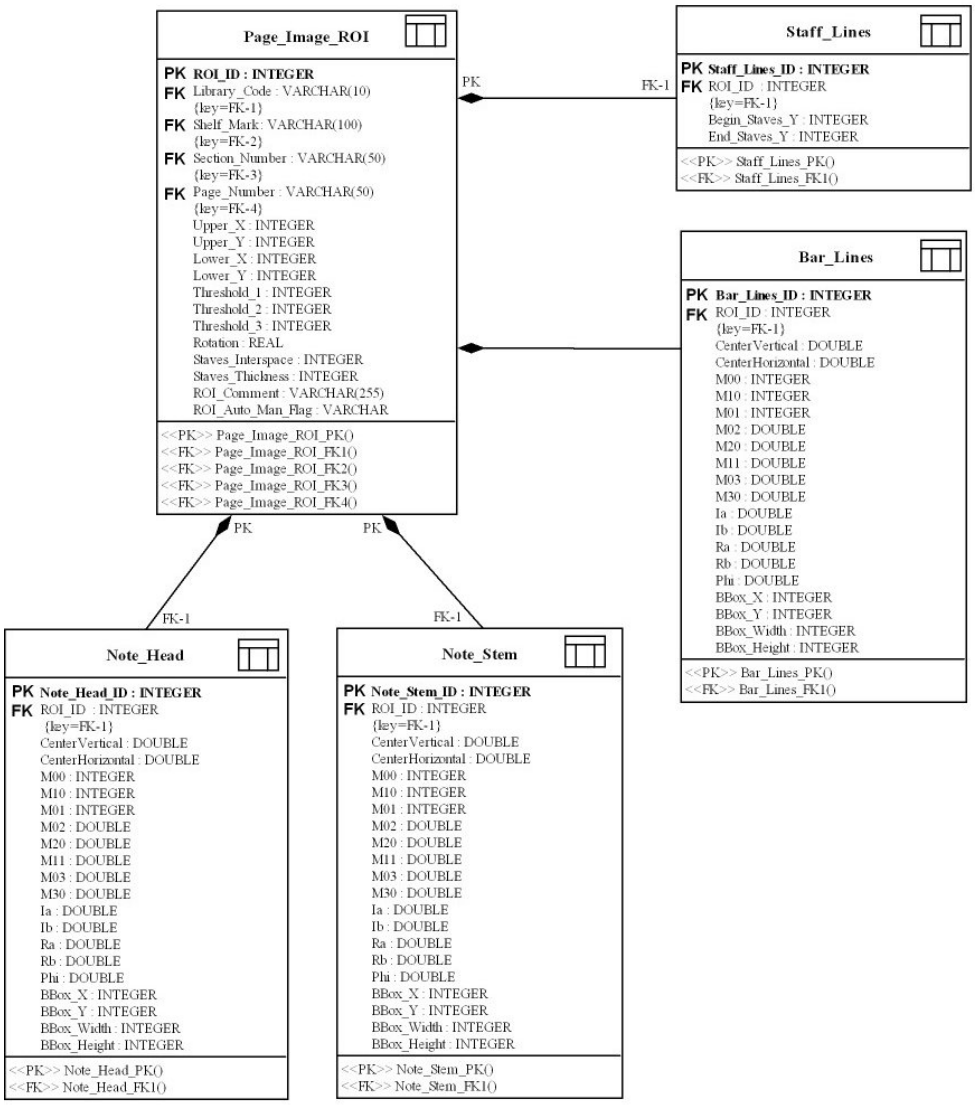


Abbildung 3.4: Relationales Datenmodell des Schemas IPFV

3.4.1 Tabelle Page_Image_ROI

Diese Tabelle speichert die ausgewählten Bereiche des Notenblattes, ROI (Region of Interest) genannt, die zur Schreiberanalyse benötigt werden. Die zu analysierenden Notensysteme werden umrahmt, und dieser Bereich wird durch X- und Y-Koordinaten beschrieben. Somit wird der Rand des Notenblattes nicht betrachtet. Zusätzlich werden der Abstand und die Dicke der Notenlinien in der Tabelle erfasst.

Attributname	Beschreibung	Beispielwert
ROI_ID	Primärschlüssel, Identifikator des Seitenbildes	
Library_Code	RSIM-Siegel der Bibliothek, eindeutiger Identifizierungssiegel einer Bibliothek	D-ROU
Shelf_Mark	Bibliothekssignatur der Notenhandschrift	Musica Saec. XVIII.-57.5
Section_Number	Nummer des Notenhandschriftenteils	
Page_Number	Seitenzahl der Notenhandschrift, entspricht einen Teil des Dateinamens des digitalisierten Notenblattes	Bcp165 (Dateiname - XVII.18.-11$\hat{4}$[Bcp165].tif)
Upper_X	Angabe der X-Koordinate in Pixel der oberen linken Ecke des entsprechenden Bereiches	
Upper_Y	Angabe der Y-Koordinate in Pixel der oberen linken Ecke des entsprechenden Bereiches	
Lower_X	Angabe der X-Koordinate in Pixel der unteren rechten Ecke des entsprechenden Bereiches	
Lower_Y	Angabe der Y-Koordinate in Pixel der unteren rechten Ecke des entsprechenden Bereiches	
Threshold_1	erster Schwellwert (Otsu Threshold Algorithm) für die Binarisierung des Bildes in dem entsprechenden Bereich	
Threshold_2	zweiter Schwellwert (Otsu Threshold Algorithm) für die Binarisierung des Bildes in dem entsprechenden Bereich	
Threshold_3	dritter Schwellwert (Otsu Threshold Algorithm) für die Binarisierung des Bildes in dem entsprechenden Bereich	
Rotation	Angabe des Rotationswinkel in Grad	

Attributname	Beschreibung	Beispielwert
Staves_Interspace	Der Abstand zwischen zwei Notenlinien in einem Notensystem (Durchschnittlicher Wert in Pixel)	
Staves_Thickness	Die Breite einer Notenlinien in einem Notensystem (Durchschnittlicher Wert in Pixel)	
ROI_Comment	Kommentar zu dem entsprechenden Bereich	
ROI_Auto_Man_Flag	1 - Dieser Bereich wurde automatisch erstellt; 0 - Dieser Bereich wurde manuell erstellt	

Tabelle 3.51: Beschreibungen von Page_Image_ROI

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
ROI_ID	integer	+	+	-	+	Attr
Library_Code	varchar(10)	+	-	+	-	Attr
Shelf_Mark	varchar(100)	+	-	+	-	Attr/FullText
Section_Number	varchar(50)	+	-	+	-	Attr/FullText
Page_Number	varchar(50)	+	-	+	-	Attr/FullText
Upper_X	integer	-	-	-	-	Attr
Upper_Y	integer	-	-	-	-	Attr
Lower_X	integer	-	-	-	-	Attr
Lower_Y	integer	-	-	-	-	Attr
Threshold_1	integer	-	-	-	-	Attr
Threshold_2	integer	-	-	-	-	Attr
Threshold_3	integer	-	-	-	-	Attr
Rotation	real	-	-	-	-	Attr
Staves_Interspace	integer	-	-	-	-	Attr
Staves_Thickness	integer	-	-	-	-	Attr
ROI_Comment	varchar(255)	-	-	-	-	Attr/FullText
ROI_Auto_Man_Flag	varchar	-	-	-	-	Attr

Tabelle 3.52: Page_Image_ROI

3.4.2 Tabelle Staff_Lines

In der Tabelle werden die Notenlinien mit ihren genauen Koordinaten gespeichert, in dem die Y-Koordinate des Anfangs und des Endes der Notenlinie erfasst werden.

Attributname	Beschreibung	Beispielwert
Staff_Lines_ID	Primärschlüssel, ID der Notenlinie	
ROI_ID	Identifikator des zugehörigen Bereichs, Fremdschlüssel auf die Tabelle IPFV.Page_Image_ROI	
Begin_Staves_Y	Angabe der Y-Koordinate in Pixel, wo die Notenlinie beginnt	
End_Staves_Y	Angabe der Y-Koordinate in Pixel, wo die Notenlinie endet	

Tabelle 3.53: Beschreibungen von Staff_Lines

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Staff_Lines_ID	integer	+	+	-	+	Attr
ROI_ID	integer	+	-	+	-	Attr
Begin_Staves_Y	integer	-	-	-	-	Attr
End_Staves_Y	integer	-	-	-	-	Attr

Tabelle 3.54: Staff_Lines

3.4.3 Tabelle Note_Head

Die Tabelle beinhaltet die Notenköpfe mit ihren Koordinaten und auch Maße, welche die genaue Form des Notenkopfes repräsentieren. Es wird eine Bounding Box um den Notenkopf berechnet, dessen Daten gespeichert werden.

Attributname	Beschreibung	Beispielwert
Note_Head_ID	Primärschlüssel, ID des Notenkopfes	
ROI_ID	ID des Bereiches, in dem sich der Notenkopf befindet, Fremdschlüssel auf die Tabelle IPFV.Page_Image_ROI	
CenterVertical	vertikaler Abstand in Pixel vom Rand der Bounding-Box	0..100
CenterHorizontal	horizontaler Abstand in Pixel vom Rand der Bounding-Box	
M00	Fläche in Pixel	0..100*100
M10	M10/M00 - X-Koordinaten des Schwerpunktes	
M01	M01/M00 - Y-Koordinaten des Schwerpunktes	
M02		
M20		
M11		
M03		
M30		
Ia	Hauptträgheitsachse in Pixel berechnet aus M20, M02 und M11	> 0.0
Ib	Nebenträgheitsachse in Pixel berechnet aus M20, M02 und M11	> 0.0
Ra	Radius in Pixel aus Ia und Ib berechnet	> 0.0
Rb	Radius in Pixel aus Ib und Ia berechnet	> 0.0
Phi	Winkel in Grad (Orientierung)	-360.0 .. 360.0
BBox_X	Abstand in Pixel vom Rand des Bildes	0..10000
BBox_Y	Abstand in Pixel vom Rand des Bildes	0..10000
BBox_Width	Breite der Bounding-Box	0..100
BBox_Height	Höhe der Bounding-Box	0..100

Tabelle 3.55: Beschreibungen von Note_Head

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Note_Head_ID	integer	+	+	-	+	Attr
ROI_ID	integer	+	-	+	-	Attr
CenterVertical	double	-	-	-	-	Attr
CenterHorizontal	double	-	-	-	-	Attr
M00	integer	-	-	-	-	Attr
M10	integer	-	-	-	-	Attr
M01	integer	-	-	-	-	Attr
M02	double	-	-	-	-	Attr
M20	double	-	-	-	-	Attr
M11	double	-	-	-	-	Attr
M03	double	-	-	-	-	Attr
M30	double	-	-	-	-	Attr
Ia	double	-	-	-	-	Attr
Ib	double	-	-	-	-	Attr
Ra	double	-	-	-	-	Attr
Rb	double	-	-	-	-	Attr
Phi	double	-	-	-	-	Attr
BBox_X	integer	-	-	-	-	Attr
BBox_Y	integer	-	-	-	-	Attr
BBox_Width	integer	-	-	-	-	Attr
BBox_Height	integer	-	-	-	-	Attr

Tabelle 3.56: Note_Head

3.4.4 Tabelle Note_Stem

In der Tabelle werden die Notenhälse gespeichert. Dazu werden ihre Koordinaten erfasst und eine Bounding Box ermittelt, welche die speziellen Merkmale des Notenhalses repräsentiert.

Attributname	Beschreibung	Beispielwert
Note_Stem_ID	Primärschlüssel, ID des Notenhalses	
ROI_ID	ID des Bereichs, in dem sich der Notenhals befindet, Fremdschlüssel auf die Tabelle IPFV.Page_Image_ROI	
CenterVertical	vertikaler Abstand in Pixel vom Rand der Bounding-Box	0..100
CenterHorizontal	horizontaler Abstand in Pixel vom Rand der Bounding-Box	
M00	Fläche in Pixel	0..100*100
M10	M10/M00 - X-Koordinaten des Schwerpunktes	
M01	M01/M00 - Y-Koordinaten des Schwerpunktes	
M02		
M20		
M11		
M03		
M30		
Ia	Hauptträgheitsachse in Pixel berechnet aus M20, M02 und M11	> 0.0
Ib	Nebenträgheitsachse in Pixel berechnet aus M20, M02 und M11	> 0.0
Ra	Radius in Pixel aus Ia und Ib berechnet	> 0.0
Rb	Radius in Pixel aus Ib und Ia berechnet	> 0.0
Phi	Winkel in Grad (Orientierung)	-360.0 .. 360.0
BBox_X	Abstand in Pixel vom Rand des Bildes	0..10000
BBox_Y	Abstand in Pixel vom Rand des Bildes	0..10000
BBox_Width	Breite der Bounding-Box	0..100
BBox_Height	Höhe der Bounding-Box	0..100

Tabelle 3.57: Beschreibungen von Note_Stem

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Note_Stem_ID	integer	+	+	-	+	Attr
ROI_ID	integer	+	-	+	-	Attr
CenterVertical	double	-	-	-	-	Attr
CenterHorizontal	double	-	-	-	-	Attr
M00	integer	-	-	-	-	Attr
M10	integer	-	-	-	-	Attr
M01	integer	-	-	-	-	Attr
M02	double	-	-	-	-	Attr
M20	double	-	-	-	-	Attr
M11	double	-	-	-	-	Attr
M03	double	-	-	-	-	Attr
M30	double	-	-	-	-	Attr
Ia	double	-	-	-	-	Attr
Ib	double	-	-	-	-	Attr
Ra	double	-	-	-	-	Attr
Rb	double	-	-	-	-	Attr
Phi	double	-	-	-	-	Attr
BBox_X	integer	-	-	-	-	Attr
BBox_Y	integer	-	-	-	-	Attr
BBox_Width	integer	-	-	-	-	Attr
BBox_Height	integer	-	-	-	-	Attr

Tabelle 3.58: Note_Stem

3.4.5 Tabelle Bar_Lines

In der Tabelle werden die Taktstriche mit ihren Koordinaten und einer Bounding Box, die zur genauen Merkmalsbestimmung dient, gespeichert. Mit diesen Werten kann man anschließend die Taktstriche mit den Notensystemen zusammensetzen, so dass die einzelnen Takte des Musikstückes berechnet werden können.

Attributname	Beschreibung	Beispielwert
Bar_Lines_ID	Primärschlüssel, ID des Taktstriches	
ROI_ID	ID des Bereichs, in dem sich der Taktstrich befindet, Fremdschlüssel auf die Tabelle IPFV.Page_Image_ROI	
CenterVertical	vertikaler Abstand in Pixel vom Rand der Bounding-Box	0..100
CenterHorizontal	horizontaler Abstand in Pixel vom Rand der Bounding-Box	
M00	Fläche in Pixel	0..100*100
M10	M10/M00 - X-Koordinaten des Schwerpunktes	
M01	M01/M00 - Y-Koordinaten des Schwerpunktes	
M02		
M20		
M11		
M03		
M30		
Ia	Hauptträgheitsachse in Pixel berechnet aus M20, M02 und M11	> 0.0
Ib	Nebenträgheitsachse in Pixel berechnet aus M20, M02 und M11	> 0.0
Ra	Radius in Pixel aus Ia und Ib berechnet	> 0.0
Rb	Radius in Pixel aus Ib und Ia berechnet	> 0.0
Phi	Winkel in Grad (Orientierung)	-360.0 .. 360.0
BBox_X	Abstand in Pixel vom Rand des Bildes	0..10000
BBox_Y	Abstand in Pixel vom Rand des Bildes	0..10000
BBox_Width	Breite der Bounding-Box	0..100
BBox_Height	Höhe der Bounding-Box	0..100

Tabelle 3.59: Beschreibungen von Bar_Lines

Attributname	Format	NOT NULL	Primär-schlüssel	Fremd-schlüssel	Unique	Suche
Bar_Lines_ID	integer	+	+	-	+	Attr
ROI_ID	integer	+	-	+	-	Attr
CenterVertical	double	-	-	-	-	Attr
CenterHorizontal	double	-	-	-	-	Attr
M00	integer	-	-	-	-	Attr
M10	integer	-	-	-	-	Attr
M01	integer	-	-	-	-	Attr
M02	double	-	-	-	-	Attr
M20	double	-	-	-	-	Attr
M11	double	-	-	-	-	Attr
M03	double	-	-	-	-	Attr
M30	double	-	-	-	-	Attr
Ia	double	-	-	-	-	Attr
Ib	double	-	-	-	-	Attr
Ra	double	-	-	-	-	Attr
Rb	double	-	-	-	-	Attr
Phi	double	-	-	-	-	Attr
BBox_X	integer	-	-	-	-	Attr
BBox_Y	integer	-	-	-	-	Attr
BBox_Width	integer	-	-	-	-	Attr
BBox_Height	integer	-	-	-	-	Attr

Tabelle 3.60: Bar_Lines

3.5 Triggers

Zusätzlich zu den Datenbanktabellen wurden zwei Triggers in dem Schema IMAGES definiert:

- **SET_CREATED**: Wenn ein neues Bild bzw. Digitalisat einer eingescannten Notenhandschrift hinzugefügt wird, dann wird es in der Tabelle "Metadata.Page_Images" gespeichert. Bei diesem Einfügen springt der Trigger an, und setzt den Wert des Attributs `created` auf die aktuelle Zeit, dementsprechend auf die Zeit der Speicherung.
- **SET_UPDATED**: Im Fall einer Änderung an einem Digitalisat, setzt der Trigger entsprechend die Zeit des Attributs `updated` in der Tabelle "Metadata.Page_Images".

3.6 UDFs

Es wurden weiterhin vier UDFs im Schema IMAGES definiert:

- **GETLQPAGEIMAGE** erstellt aus dem qualitativ hochwertigen Bild (`LQPageImage`), welches in der Tabelle "Metadata.Page_Images" gespeichert ist, ein Bild von niedriger Qualität. Als Parameter wird der Funktion ein Dateiname übergeben, der auf das qualitativ hochwertige Bild verweist. Das qualitativ niedrige Bild wird zum Anzeigen bzw. zur Ausgabe verwendet, während das qualitativ hochwertige Bild für Bildanalyseprozesse genutzt wird.
- **GETTHMBPAGEIMAGE** erstellt aus dem qualitativ hochwertigen Bild (`LQPageImage`), welches in der Tabelle "Metadata.Page_Images" gespeichert ist, ein Miniaturbild von der Größe eines Daumennagels, welches zu Navigationszwecken verwendet wird.
- **GETHEIGHTPAGEIMAGE** liefert den Wert des Attributs `height` in der Tabelle "Metadata.Page_Images", welches die Höhe des Bildes entspricht.
- **GETWIDTHPAGEIMAGE** liefert den Wert des Attributs `width` in der Tabelle "Metadata.Page_Images", welches die Breite des Bildes entspricht. Mit den Funktionen "GETHEIGHTPAGEIMAGE" und "GETWIDTHPAGEIMAGE" kann man Berechnungen anstellen, um die Pixelmenge eines Bildes zu bestimmen, um ein optimiertes Anzeigen des Bildes zu gewährleisten oder auch Rückschlüsse auf das Format (ob Quer- oder Hochformat) des Bildes zu ziehen.
- **A_FV** bekommt als Parameter eine Menge von Feature-Werten übergeben, und berechnet zu dem daraus resultierenden Feature-Vektor eine Liste mit den ähnlichsten Feature-Vektoren aus der Datenbank. Somit enthält diese Liste die Schreiber mit einer ähnlichen Handschriftcharakteristik als der Schreiber des Ausgangsvektors. Diese

UDF und genaue Beschreibungen zur Distanzberechnung von Feature-Vektoren folgt im Kapitel 4.

Kapitel 4

Distanzfunktion

Wenn eine Menge von Datensätzen vorliegt, möchte man Zusammenhänge zwischen ihnen erkennen. Dafür kommen Algorithmen des Data Minings zum Einsatz. In unserem Beispiel gibt es eine Menge von Feature-Vektoren, zwischen denen man den Zusammenhang in Form von Ähnlichkeiten untersuchen möchte. Ein Feature-Vektor besteht aus n Features (ungefähr 80), die jeweils eine Achse im n-dimensionalen Raum bilden. Wenn man die Werte der Features auf den entsprechenden Achsen abträgt, wird jeder Feature-Vektor durch einen Punkt in diesem Raum repräsentiert. Nun ist es die Aufgabe die Abstände zwischen den Punkten bzw. zwischen den Feature-Vektoren zu bestimmen. Dazu wählt man eine Distanzfunktion, die für jedes Paar von Feature-Vektoren einen skalaren Wert berechnet. Die Wahl der Distanzfunktion ist abhängig von der Art der Eingabedaten, ihren Datentypen und der vorliegenden Problemstellung.

4.1 Beschreibung der Distanzfunktion

In diesem Projekt werden die Distanzen zwischen Feature-Vektoren mit der Hamming-Distanz ermittelt, da sie in dem vorliegenden Fall die besten Resultate erzielt. Zuerst braucht man eine Bewertungsfunktion, welche die Distanz zweier einzelner Merkmale bzw. Features auf einen Wert zwischen 0 und 1 abbildet. Diese Bewertungsfunktion ist von der Art des Features abhängig. Einerseits kommt der einfache Wertevergleich (gleich oder nicht, 0 oder 1) in Frage. Andererseits werden die Distanzen aller Werte des Wertebereichs eines Features in Distanzmatrizen gespeichert, welche heuristisch von den Musikwissenschaftlern erstellt wurden. Wenn der Wertevergleich zweier Features über mehrere Ebenen stattfindet, werden die Distanzen für jede Ebene bestimmt, aufaddiert und die Ebenenverwandtschaft in einer weiteren Distanzmatrix abgelesen. Zwei Werte, deren Pfade sich nur in einer Ebene unterscheiden sind relativ ähnlich, und können einen geringen Distanzwert erhalten. Distanzmatrizen sind effizienter; sie benötigen am Anfang einen erhöhten Berechnungsaufwand, dafür sind keine Berechnungen zur Laufzeit notwendig.

Nachdem man die Distanzwerte für jedes Feature-Paar (d_{f_i}) bestimmt hat, wird der Durchschnitt der Distanzen gebildet. Jedes Feature erhält zusätzlich noch ein Gewicht (ω_i), welches die Relevanz bzw. Wichtigkeit dieses Features widerspiegelt. Die Formel zur Distanzberechnung lautet:

$$d_\Gamma = \frac{\omega_1 d_{f_1} + ... + \omega_n d_{f_n}}{n}$$

Wenn die Distanz zweier Feature-Vektoren klein ist, dann sind die Handschriftcharakteristiken, die sie repräsentieren, ähnlich. Mit Clustering-Methoden aus dem Data-Mining-Bereich kann man eine Menge von Datensätzen, in unserem Bespiel Feature-Vektoren, in Klassen einteilen, die ähnliche Handschriftmerkmale widerspiegeln und im besten Fall genau einem Schreiber entsprechen. Im vorliegenden Fall sind die Klassen bereits bekannt, und die Aufgabe ist es, neue Feature-Vektoren zu klassifizieren. Im enoteHistory-Projekt wurde das instanzbasierte Klassifikationsverfahren k-nearest-neighbor eingesetzt, da es sich am günstigsten erwies. Es klassifiziert existierende Feature-Vektoren mit der Verwendung der Hamming-Distanz und lernt mit jedem neuen Feature-Vektor.

Detaillierte und formelle Beschreibungen zu der Distanzfunktion sind in der Diplomarbeit von Lars Milewski [Mil04] zu finden.

Vorgehensweise bei einer Anfrage

Wenn der Nutzer den Schreiber einer unbekannten Notenhandschrift bestimmen möchte, dann stellt er eine Anfrage an das System, indem er manuell die Merkmale der vorliegenden Handschrift abliest. Die Merkmale der Handschrift bilden einen Feature-Vektor. Zur Bestimmung des Schreibers wird zuerst mit dem Klassifikationsverfahren k-nearest-neighor die k-nächsten Feature-Vektoren zu dem Anfrage-Vektor berechnet, deren Distanz unter einem gewissen Schwellwert liegt. Diese ermittelten Feature-Vektoren repräsentieren die ähnlichsten Handschriften zu der unbekannten Notenhandschrift. Anschließend wird der Schreiber ermittelt, zu dem die meisten Feature-Vektoren in dieser berechneten Menge gehören. Ist die Anzahl der Feature-Vektoren zweier Schreiber gleich, wird der Schreiber zu dem Feature-Vektor ausgegeben, der den kleinsten Distanzwert zum Anfrage-Vektor hat. Somit erhält der Nutzer eine gerankte Liste von Schreibern. In der Abbildung 4.1 ist der Ablauf dargestellt, indem die Handschriftcharakteristik einer Notenhandschrift extrahiert und klassifiziert wird.

4.2 Evaluierung der Distanzfunktion

Um die besten Resultate bei der Schreibererkennung zu erzielen, ist es notwendig die Distanzfunktion und ihre Parameter zu evaluieren. Dazu bedient man sich einer Scoring-Funktion für instanzbasierte Clustering-Methoden. Die Scoring-Funktion ist ein Maß zur

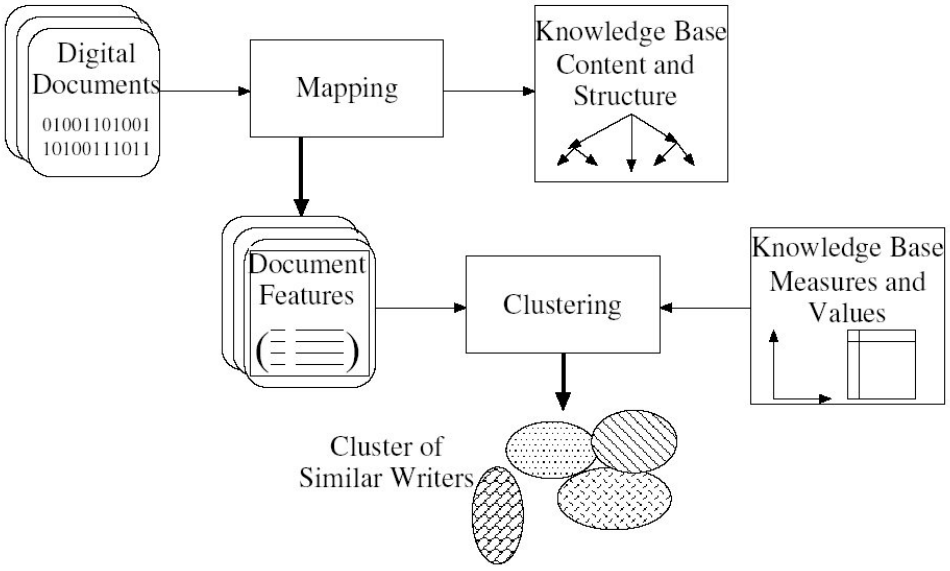


Abbildung 4.1: Extraktion

Ähnlichkeit von Clustern. Die Distanzen zwischen Instanzen in einem Cluster sollen so klein wie möglich sein; dagegen versucht man die Distanzen zwischen Instanzen unterschiedlicher Cluster zu maximieren. Das K-Maß bewertet das System basierend auf Klassifikationsaspekte. Die berechnete Ergebnismenge zur Identifikation des Schreibers wird daran bewertet, ob es den richtigen Schreiber enthält oder nicht. K ist somit der Anteil aller Anfragen, in denen der Schreiber richtig erkannt wurde.

Wahl der Distanzfunktion

Es wurden verschiedene Distanzfunktionen getestet und bewertet. Im Schreibererkennungssystem sind keine Cluster-Verfahren zur Berechnung der Klassen notwendig, da die Schreiberklassen bereits bekannt sind. Um die Qualität der Distanzfunktion und ihre Parameter zu bewerten, wird ein Clustering-Algorithmus auf die bereits existierenden Feature-Vektoren angewendet. Die Ausgabe soll genau die bereits bekannten Klassen berechnen. Dabei erzielte die Hamming-Distanz, gemessen an der Scoring-Funktion und dem K-Maß, die besten Resultate.

Optimierung des Schwellwertes

Der Umfang der Ergebnismenge von Feature-Vektoren hängt vom Schwellwert ab. Zur Ermittlung des optimalen Schwellwerts wurden Precision und Recall verwendet. Precision ist ein Maß für die Qualität des Ergebnisses und repräsentiert den Anteil der Feature-Vektoren, die wirklich relevant sind. Dagegen ist Recall ein Maß für die Quantität des

Ergebnisses und gibt den Anteil der erwarteten Feature-Vektoren an, die auch wirklich im Ergebnis enthalten sind. Als optimalen Schwellwert ergab sich 0,13, mit dem ein Precision-Wert von 85% und ein Recall-Wert von 70% erreicht wurde.

Gewichte der Features

In der Distanzfunktion erhält jedes Feature ein Gewicht, welches die Relevanz des Features in der Distanzberechnung repräsentiert. Diese Gewichte beeinflussen somit den Distanzwert zwischen zwei Feature-Vektoren. Eine Optimierung eines einzelnen Gewichts kann man erreichen, indem man die Distanz ohne das entsprechende Feature berechnet und analysiert, ob sich das Ergebnis verbessert oder verschlechtert. Wenn sich das Ergebnis verbessert, dann hat das Feature keine so große Bedeutung und sein Gewicht wird verringert. Wenn sich dagegen das Ergebnis verschlechtert, ist das Feature wichtiger und sein Gewicht muss erhöht werden.

Weiterhin muss die Verteilung der Gewichte betrachtet werden. Dabei ergab sich als Kompromiss zwischen einer guten Scoring-Funktion und einem guten K-Maß eine lineare Verteilung der Gewichte, bei der alle Features unterschiedliche, aber linear ansteigende Gewichte haben. Im Prototyp wurde eine Mischung aus den Gewichten gewählt, die auf dem musikwissenschaftlichen Fachwissen basiert und aus den Tests ermittelt wurden.

Distanzmatrizen

Die Werte in den Distanzmatrizen bestimmen die Ähnlichkeit bzw. den Unterschied zwischen zwei Werten eines Features. Diese Werte wurden von den Musikwissenschaftlern festgelegt. Um auch die Distanzmatrizen zu optimieren, wurden die Werte etwas variiert. Die Ergebnisse konnten nur minimal verbessert werden, indem die ursprünglichen Werte etwas herabgesetzt wurden. Im Prototyp wurden die Werte in den meisten Fällen beibehalten.

Nullwerte

In dem Schreibererkennungssystem unterscheidet man zwischen zwei Arten von Nullwerten für Features:

- **Non-Information-Null** bedeutet, dass das entsprechende Feature nicht verwendet wurde, und man nicht weiß, wie der Schreiber dieses Feature geschrieben hätte. Diese Nullwerte gehen nicht in die Distanzberechnung mit ein, und haben somit auch keinen Einfluss auf das Klassifikationsergebnis.
- **No-Applicable-Null** signalisiert, dass der Schreiber ein solches Zeichen nicht schreibt. Für den Schreiber wird nie ein Wert dafür existieren, und deswegen erhält

dieser Wert die maximale Distanz zu allen anderen Werten des Wertebereichs. Dieser Nullwert repräsentiert auch einen Wert im Wertebereich des Features, und befindet sich direkt unter dem Feature-Knoten in der Feature Base.

Die Entscheidung, um welche Art des Nullwertes es sich handelt, wird automatisch getroffen, da der Nutzer den Unterschied zwischen ihnen nicht kennt. Auf jeden Fall erhält man bessere Ergebnisse, wenn man so wenige Nullwerte wie möglich im Anfrage-Vektor hat.

ND-Wert

Bei der manuellen Schreiberanalyse wird neben der Distanz auch der ND-Wert angegeben, der den prozentualen Anteil der Merkmale angibt, die zur Berechnung der Distanz herangezogen wurden. Da Nullwerte Einfluss auf das Ergebnis haben, ist somit der Distanzwert verlässlicher, je größer der ND-Wert ist.

Anhand die durchgeführten Tests konnte man eine optimale Konfiguration der Distanzfunktion und ihren Parametern, sowie des Systems finden, welches zur Steigerung der Erkennungsrate von Schreibern beiträgt. In den Tests lag die Erkennungsrate bei 90% bis 95%.

Genauere Informationen zu den Testdurchführungen und deren Ergebnissen sind in der Diplomarbeit von Lars Milewski [Mil04] zu finden.

4.3 Implementierung der UDF

Für dieses Projekt war das Verwenden von bereits existierenden Data-Mining-Tools zu aufwendig. Darum wurden Data-Mining-Techniken zur Schreibererkennung mit Hilfe einer UDF umgesetzt, welche in das DB2 Datenbanksystem integriert wurde. UDFs werden anhand ihres Rückgabewertes unterschieden, welche entweder eine skalare Funktion, eine Spaltenfunktion oder eine Tabellenfunktion sein können. Es wurde eine UDF `A_FV` als Tabellenfunktion geschrieben. `A_FV` berechnet zu einem Anfrage-Vektor die nächsten Feature-Vektoren, deren Distanz unter einem gewissen Schwellwert liegt, und gibt sie als Liste mit den dazugehörigen Schreibern zurück.

Die Klassen `UDFs` und `UDFapp`

Dabei wird die Klasse `UDFs` von der Datenbank verwendet, während die Klasse `UDFapp` von der Kommandozeile aus gestartet (instanziiert) wird. Beide Klassen haben die UDF-Methode `A_FV`, welche die gleiche Funktionalität besitzt.

Zur Implementierung der Algorithmen und Strukturen wurde Java 1.4 verwendet. Die Java-Klassen befinden sich in der JAR-Datei `enh` im Package de.enotehistory. Zur Anbindung an

die Datenbank wurde die JDBC-Schnittstelle verwendet. Im Package de.enotehistory befinden sich die weiteren Packages `Datatypes` und `Tools`, die von der UDF genutzt werden. Abschnitt 4.3.2 erklärt die genauen Programmschritte.

4.3.1 Package Datatypes

Interfaces

FeatureValue ist ein Interface, welches Methoden zum Speichern und allgemeines Arbeiten mit Feature-Werten definiert. Ein Feature-Wert basiert auf eine Punktnotation, welche bereits im Kapitel 1.3 erwähnt wurde. Dieses Interface definiert folgende Methoden:

- **getValue()**: Liefert in einem String-Array die Werte eines Features zurück. Normalerweise wird nur ein Wert zurückgeliefert.
- **getFeature()**: Liefert das Feature, welches zu diesem Feature-Wert gehört.
- **distance(FeatureValue otherValue)**: Berechnet die Distanz zu einem gegebenen Feature-Wert.
- **isNullT()**: Gibt an, ob der Wert ein Non-Applicable-Nullwert ist.

Feature Das Interface "Feature" definiert Methoden zur Beschreibung eines Features. Es wurden get- und set-Methoden deklariert, die zur Abfrage und zur Speicherung des Feature-Codes, der Beschreibung des Features, der Position in der Prioritätenliste und des Gewichtes in der Distanzfunktion dienen.

FeatureVector ist ein Interface, welches Methoden zur Verwaltung eines Feature-Vektors definiert. Es existieren get- und set-Methoden zur Speicherung und Rückgabe des Feature-Vektors, der Bewertungsmaße, der Signatur, des zugehörigen Schreibers und seiner Schreibperiode. Die Methode `distance()` berechnet die Distanz zu einem anderen Feature-Vektor. Die Methode `getVector()` gibt die Werte des Feature-Vektors als Array von Objekten des Typs FeatureValue zurück.

Implementierungen der Interface

DBFeatureValue implementiert das Interface `FeatureValue`.

MemFeature implementiert des Interface `MemFeature`. Die Informationen werden einmalig aus der Datenbank geholt.

MemFeatureVector implementiert das Interface `FeatureVector`. Die Methode `distance()` testet zuerst ob die Arrays der Feature-Werte der beiden Feature-Vektoren korrekt sind. Um die Distanz zu einem Feature-Vektor zu berechnen, wird die Distanzfunktion zu jedem Feature-Wertepaar angewandt. Wenn ein Feature-Wert ein Nullwert ist, dann liefert die Feature-Wert-Distanzfunktion -1 und dieses Feature wird für die weitere Berechnung ignoriert. Zur endgültigen Distanzberechnung zwischen zwei Feature-Vektoren wird dann die Summe über die Distanzen und den Gewichten gebildet.

Zusätzliche Klassen

FeatureManager Diese Klasse verwaltet eine Liste von Features. Es sind folgende Methoden definiert:

- **getFeatures()** liefert ein Array von Features zurück.
- **getFeature(String code)** liefert ein Feature zurück. Dabei wird ein effizienteren Zugriff auf das Feature mittels einer Hashtabelle realisiert.
- **addFeature(Feature feature)** fügt ein Feature zu dem Array hinzu.

Distance speichert das Resultat der Distanzfunktion zweier Feature-Vektoren. Es wird der Distanzwert wie auch das Gewicht gespeichert. Die Funktion `distance(FeatureVector othVector)` der Klasse `MemFeatureVector` liefert ein Objekt der Klasse `Distance` als Ergebnis zurück.

DistanceMatrix speichert eine Distanzmatrix und stellt folgende Funktionen darauf zur Verfügung:

- **getDistance(String value1, String value2)** liefert den Wert in der Matrix, der mit `Spalte=value1` und `Reihe=value2` gespeichert wurde.
- **recalcHash()** erstellt ein Label-indexierte Hashtabelle.
- **mergePDMhorizontal(PartialDistanceMatrix[] pdm)** verbindet die als Parameter im Array übergebenen Teilmatrizen mit der Distanzmatrix.

HierNode Die Klasse repräsentiert einen Knoten in der Feature Base. Der Knoten hat einen Knoten-Code, eine Knotenbeschreibung, einen Dateinamen des Piktogramms und einen Knotentyp als Attribute. Als Methoden sind definiert:

- **childPath(String parent)** liefert den Pfad in der Feature Base, mit dem sich der Vaterknoten und der Kindknoten unterscheiden. Beispielsweise würde "_.1." für den Vaterknoten "1.1.1." und seinem Kindknoten "1.1.1._.1" zurück geliefert werden.

- **int numberOfLevels(String code)** berechnet die Anzahl der Levels (Tiefe und Länge) des Knotenpfades.
- **path(String code)** liefert einen Knotenpfad als String-Array zurück. Jeder String repräsentiert ein Level des Pfades.

NodeScanInfo enthält Informationen über einen Knoten in der Feature Base. Diese Daten werden bei Scannen der HTML Struktur der Feature Base benötigt.

PartialDistanceMatrix repräsentiert einen Teil der Distanzmatrix, welcher nicht quadratisch ist. Sie entspricht einer Teilmenge von Zeilen der Distanzmatrix. Die Distanzmatrix wurde geteilt, weil Excel nicht mehr als 255 Zeilen verwalten kann.

TreeNode repräsentiert einen Knoten in der Feature Base durch seinen Pfad. Diese Klasse wird von den Tools zum Import der Distanzmatrizen benötigt. Es sind Methoden definiert, die den Wert eines Features in Punktnotation ändern bzw. zurückgeben, oder in Form eines Pfadausdruckes liefern.

EnhConnection

Im Package `Tools` befindet sich die Klasse `EnhConnection`, welche Methoden für die Datenbankanfragen bereitstellt. Diese Datenbankanfragen werden in der Klasse `UDFapp` und `UDFs` benötigt, um die Feature-Vektoren aus der Datenbank, inklusive mit ihren Feature-Values, zu bekommen.

4.3.2 Ablauf der UDF-Funktion

Beim Aufruf der UDF-Methode `A_FV` werden 79 Feature-Werte übergeben, die anschließend in einen Feature-Vektor umgeformt werden. Zu diesem werden nun die Feature-Vektoren aus der Datenbank gesucht, zu denen die Distanz kleiner als 0.13 ist. Das bedeutet, dass die ähnlichsten Handschriften ermittelt werden. In der Methode wird als erstes eine Verbindung zu der Datenbank aufgebaut. Dann wird eine Liste mit allen Features und allen Feature-Vektoren geholt, die sich in der Datenbank befinden. In einer Schleife wird zu jedem Feature-Vektor aus der Datenbank die Distanz zum Eingabe-Vektor berechnet. Die Distanz, bestehend aus dem Distanzwert und dem Gewicht der Features ω, wird als Objekt der Klasse `Distance` gespeichert. Wenn die Distanz kleiner als 0.13 ist, dann wird dieser Feature-Vektor in das Ergebnis inklusive dem Schreiber dieser Notenhandschrift und dem Distanzwert aufgenommen. Aus der Datenbank wird die Summe aller Features (maxWeightSum) abgefragt. Zu jedem gefundenen Feature-Vektor im Ergebnis wird der ND-Wert berechnet, in dem das Gewicht der Features ω durch maxWeightSum dividiert

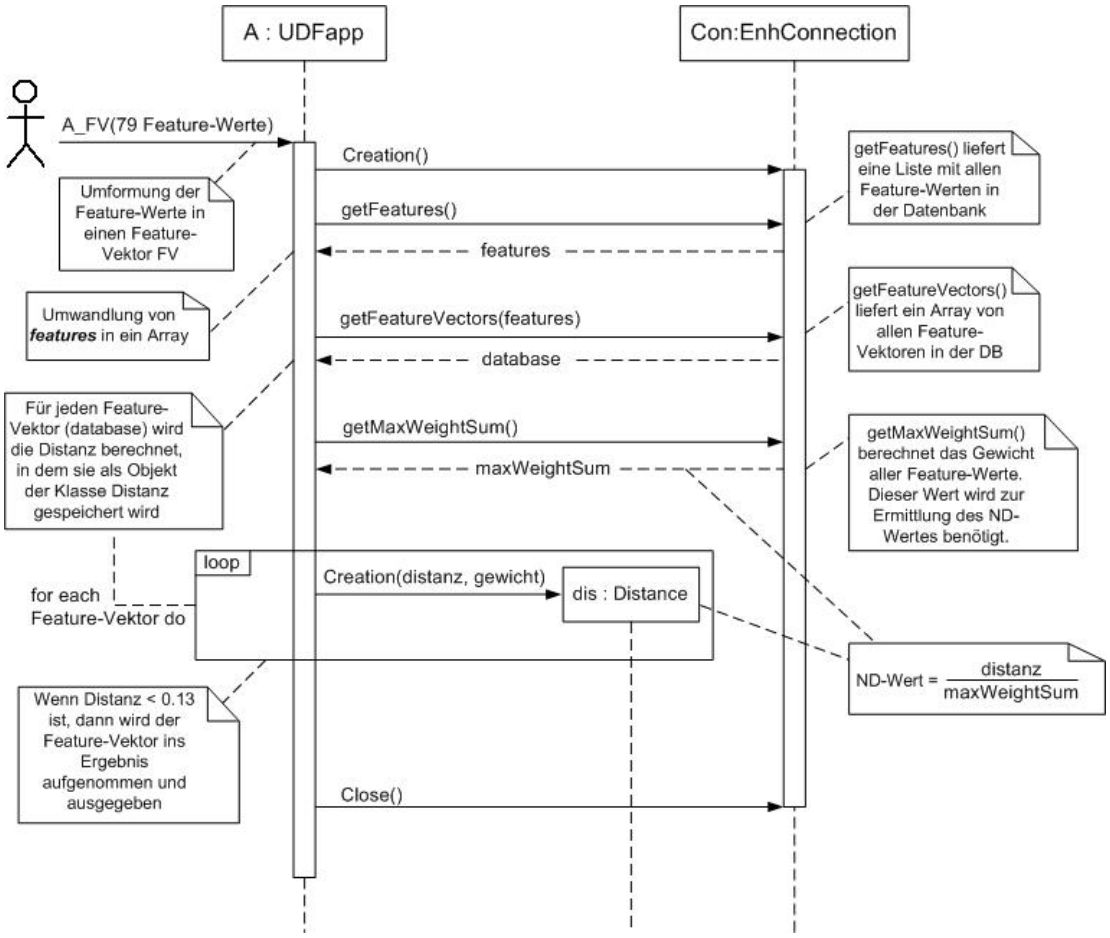


Abbildung 4.2: UML-Sequenzdiagramm zur UDF-Funktion

wird. Anschließend wird das Ergebnis ausgegeben. Diese Abarbeitungsschritte werden auch im UML-Diagramm 4.2 dargestellt.

Kapitel 5

eNotehistory Webpräsenz

Um das System der Öffentlichkeit zugänglich zu machen, wurde eine Internetpräsenz unter *www.enotehistory.de* erstellt. Dort erhält man Informationen über das Projekt und deren Projektpartnern, und man kann das System in Aktion sehen bzw. selber nutzen. Die Webschnittstelle kann für die Metadatensuche, die Navigation im Dokumentenbestand und die manuelle Schreiberidentifikation. Wie man diese Funktionalität nutzen kann, erklärt der folgende Abschnitt 5.1. Bei dieser webbasierten Nutzungsschnittstelle kommen Java Servlets zum Einsatz, die dynamische Webseiten erzeugen. Was Java Servlets sind und wie sie in dem System zum Einsatz kommen, wird anschließend im Abschnitt 5.2 erläutert.

5.1 Webseiten des eNoteHistory-Projektes

Unter der Adresse *www.enotehistory.de* gelangt man zu den Webseiten des Projektes, auf denen eine Navigation und Suche in den Metadaten und in den digitalisierten Notenhandschriften möglich ist.

Abbildung 5.1: eNoteHistory-Webseite

Die Webseiten sind in den folgenden vier Bereiche unterteilt, wie es auch in der Abbildung 5.1 zu sehen ist:

- Startseite: allgemeine Informationen zum Projekt und zu den drei Projektpartnern
- Suche: Suche nach Signaturen, Werktitel, Komponisten und Schreiber
- Navigation: Navigation durch die vorhandenen Notenhandschriften
- Analyse: Manuelle und automatische Schreiberanalyse

Durch die Auswahl eines Bereiches wird man entsprechend weitergeleitet; siehe folgende Abschnitte.

5.1.1 Startseite

Auf der Startseite befinden sich Links, die auf die Seiten der einzelnen Projektpartner verweisen. Dort findet man genauere Informationen, die Publikationen und zum Projekt zugehörige Diplomarbeiten. Weiterhin sind alle anstehenden Ereignisse und Projektpräsentationen auf der Webseite veröffentlicht.

5.1.2 Suche

Die Suche ermöglicht es, nach speziellen Notenhandschriften zu suchen (siehe Abbildung 5.2), indem man den Werktitel, den Namen des Komponisten oder des Schreibers angibt. Die Suche mit Hilfe der Signatur ist auch möglich. Als Ergebnis der Suche bekommt man eine Liste mit allen zutreffenden Notenhandschriften.

Abbildung 5.2: Suche nach Notenhandschriften

5.1.3 Navigation

Die Webseite zur Navigation, wie in der Abbildung 5.3 zu sehen, ist eine Auflistung aller Notenhandschriften, die sich in unserer Datenbasis befinden. Man kann die Liste seitenweise durchgehen, und man wird weitere Informationen finden, wenn man den Links zu den Werk-Titeln oder Komponisten folgt.

Bibliothek	Signatur	Werk-Titel	Komponist	Bilder
D-B	Sammlung Thulemeier Nr. 88	Ouverture D	Johann Gottlieb Graun	N/A
D-B	Sammlung Thulemeier Nr. 14	Concerto g	Carl Philipp Emanuel Bach	N/A
D-B	Sammlung Thulemeier Nr. 12	Concerto d	Carl Philipp Emanuel Bach	N/A
D-ROu	Musica Saec. XVIII.-9.^9.10	Trios	Giuseppe Antonio Brescianello	N/A
D-ROu	Musica Saec. XVIII.-9.^8	Cantata Core amante di perché libertá	Giuseppe Antonio Brescianello	Bild
D-ROu	Musica Saec. XVIII.-9.^7	Cantata Sequir fera che fugge	Giuseppe Antonio Brescianello	N/A
D-ROu	Musica Saec. XVIII.-9.^6	Ouverture A	Giuseppe Antonio Brescianello	N/A
D-ROu	Musica Saec. XVIII.-9.^5	Ouverture D	Giuseppe Antonio Brescianello	N/A
D-ROu	Musica Saec. XVIII.-9.^4	Ouverture g	Giuseppe Antonio Brescianello	N/A
D-ROu	Musica Saec. XVIII.-9.^3	Ouverture F	Giuseppe Antonio Brescianello	N/A

Abbildung 5.3: Navigation in den Notenhandschriften

5.1.4 Analyse

Bei der Analyse entscheidet man sich zuerst zwischen der manuellen und automatischen Schreiberanalyse. Es öffnet sich entsprechend ein Popup-Fenster.

Manuelle Schreiberanalyse

In dem neuen Fenster findet man links eine Liste von Merkmalen, die das erste Level in der Feature Base (siehe Abschnitt 1.3) repräsentieren. Man kann die Merkmalsliste "Merkmal für Merkmal" aufklappen, und am Ende einen Wert für das entsprechende Feature auswählen. Diese Struktur entspricht der Feature Base, in dessen Baumstruktur sich die Werte der Features am Ende befinden.

Um eine unbekannte Notenhandschrift zu analysieren, sollte man so viele Werte wie möglich auswählen. Für jedes Feature muss man den Pfad in der Baumstruktur folgen, bis man am Ende eine Menge von Beispielen in Form von Bildern repräsentiert bekommt. Zwischen denen wählt man das Ähnlichste aus, was dem Feature auf dem Notenblatt entspricht. Es wird einige Zeit dauern, um alle Features auszuwählen; aber man wird ein besseres Ergebnis erzielen.

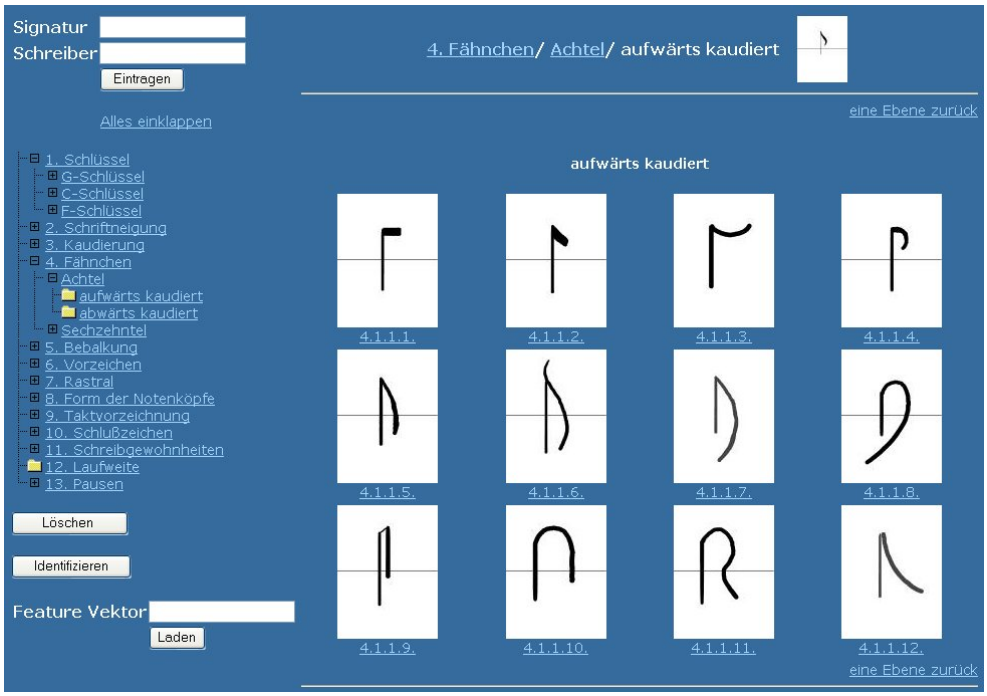
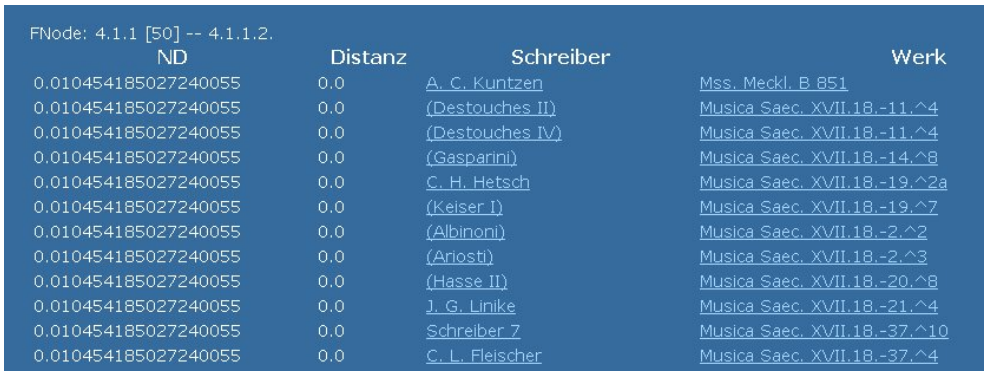


Abbildung 5.4: Auswahl des Notenfähnchen einer aufwärts kaudierten Achtelnote

Die Abbildung 5.4 zeigt Bilder von Notenfähnchen einer aufwärts kaudierten Achtelnote. Nachdem man sich für einen Wert entschieden hat, wird dieses Feature in der Baumstruktur mit einem Häkchen markiert. Man kann für ein Feature auch mehrere Werte auswählen. Durch `Identifizieren` wird die Menge der ausgewählten Features, was einen Feature-Vektor entspricht, zum Ermitteln des Schreibers an den Server geschickt. Als Ergebnis erhält man eine Liste von möglichen Schreibern, wie in Abbildung 5.5 zu sehen ist.



FNode: 4.1.1 [50] -- 4.1.1.2.

ND	Distanz	Schreiber	Werk
0.010454185027240055	0.0	A. C. Kuntzen	Mss. Meckl. B 851
0.010454185027240055	0.0	(Destouches II)	Musica Saec. XVII.18.-11.^4
0.010454185027240055	0.0	(Destouches IV)	Musica Saec. XVII.18.-11.^4
0.010454185027240055	0.0	(Gasparini)	Musica Saec. XVII.18.-14.^8
0.010454185027240055	0.0	C. H. Hetsch	Musica Saec. XVII.18.-19.^2a
0.010454185027240055	0.0	(Keiser I)	Musica Saec. XVII.18.-19.^7
0.010454185027240055	0.0	(Albinoni)	Musica Saec. XVII.18.-2.^2
0.010454185027240055	0.0	(Ariosti)	Musica Saec. XVII.18.-2.^3
0.010454185027240055	0.0	(Hasse II)	Musica Saec. XVII.18.-20.^8
0.010454185027240055	0.0	J. G. Linike	Musica Saec. XVII.18.-21.^4
0.010454185027240055	0.0	Schreiber 7	Musica Saec. XVII.18.-37.^10
0.010454185027240055	0.0	C. L. Fleischer	Musica Saec. XVII.18.-37.^4

Abbildung 5.5: Ergebnisliste der Notenhandschriftenanalyse

Ergebnisliste

In dieser Liste werden zu jedem Schreiber ein ND-Wert und ein Distanzwert angegeben. Der ND-Wert gibt den prozentualen Anteil der Merkmale an, die zur Berechnung verwendet wurden. Die restlichen Merkmale wurden als Nullwerte und somit als unbekannt gewertet. Da Nullwerte Einfluss auf das Ergebnis haben, ist das berechnete Ergebnis umso aussagekräftiger je größer der ND-Wert ist. Der Distanzwert spiegelt die Ähnlichkeit der Handschriften wider. Je kleiner die Distanz ist, desto ähnlicher sind sich die unbekannte Notenhandschrift und die Handschrift von dem entsprechenden Schreiber.

Löschen von Feature-Werten

Ein Häkchen erscheint in der Baumstruktur, wenn man einen Wert für das Feature ausgewählt hat. Falls man sich geirrt hat und den Wert wieder löschen möchte, geht man mit dem Mauszeiger auf das Häkchen. Es wird ein kleines Fenster geöffnet, in dem man auf dem Link `Löschen` klicken kann, um diesen Feature-Wert wieder zu entfernen.

Möchte man alle Werte seiner Analyse löschen, dann kann man den Button `Löschen` drücken.

Laden eines Feature-Vektors

Das Eingabefeld, zum Laden eines Feature-Vektors ist nur für Experten gedacht. Jeder Wert eines Features wird durch einen Zahlencode repräsentiert, den man dort eingeben kann. In einer Menge von Feature-Werten werden die Werte durch Komma getrennt.

Neue Notenhandschrift hinzufügen

Wie in Abbildung 5.6 zu sehen ist, kann man auch eine neue Notenhandschrift zur Datenbasis hinzufügen. Dies ist aber nur Experten vorbehalten. Nachdem man die Werte für die Features der neuen Notenhandschrift ausgewählt hat, trägt man in den Eingabefeldern die Signatur und den Schreiber ein, und schickt es an den Server, der es in die Datenbank einträgt.

Abbildung 5.6: Eintragen einer neuen Notenhandschrift

Automatische Schreiberanalyse

Das Fraunhofer Institut hat ein Programm zur Bildanalyse von Notenhandschriften entwickelt, welches noch nicht in die Webseiten integriert wurde.

5.2 Servlets-Grundlagen

Die Webseiten des eNoteHistory-Projektes wurden mit Java Servlets erstellt. Java Servlets erweitern die Funktionalität eines Webservers, da sie als serverseitige Java-Komponente dynamische Webinhalte erzeugen. Sie werden in der Programmiersprache Java geschrieben, und ihnen steht eine mächtige Klassenbibliothek für den Datenbankzugriff zur Verfügung. Somit wird dem Client ermöglicht, mit der Datenbank zu kommunizieren.

Mit eventuellen Eingaben aus einem Formular, anderen Daten aus dem Browser bzw. Cookies werden entsprechend der Implementierung, eine Webseite als Antwort generiert und zum Client geschickt. Im Vergleich zu anderen Alternativen sind Servlets nicht nur effizienter, sondern auch leistungsfähiger und sicherer. Die Java Virtual Machine läuft ständig und für die einzelnen Anfragen braucht man lediglich Threads starten, anstatt ganze Prozesse.

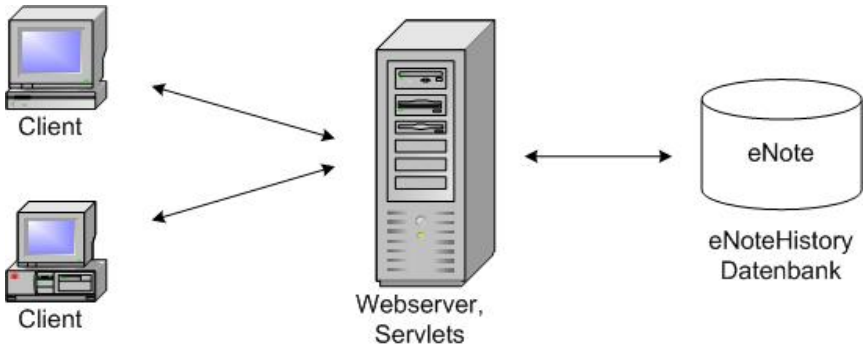


Abbildung 5.7: Einsatzgebiet von Servlets

5.3 eNoteHistory-Servlets

Die Webseiten wurden mit Servlets entwickelt, die den HTML-Code der Seiten generieren. In der Datei web.xml sind alle Servlets definiert. Es folgen kurze Beschreibungen der einzelnen Servlets.

Analysis Dieses Servlet generiert den HTML-Code für die Analyse. Auf der Seite kann man die manuelle oder die automatische Schreiberanalyse auswählen. Der Aufruf des Servlets geschieht mit *www.enotehistory.de/analysis*.

AutoAnalysis Dieses Servlet generiert den HTML-Code für die automatische Schreiberanalyse. Nachdem man sich für die automatische Schreiberanalyse entschieden hat, wird dieses Servlet mit *www.enotehistory.de/autoanalysis* aufgerufen.

Build Dieses Servlet erstellt alle Bilddateien, verändert die Größe wenn notwendig und schreibt sie an entsprechender Stelle auf die Festplatte. Zukünftige Implementierungen werden mit einem Parameter `force` erweitert, um sicher zu stellen, dass alle Bilder überschrieben werden.

Composer Dieses Servlet generiert den HTML-Code für eine Seite, die Informationen über den Komponisten enthält. Bei der Suche und bei der Navigation auf den eNoteHistory-Webseiten erhält man eine Liste mit Notenhandschriften. Der Komponist eines Musikstückes ist auch angegeben, und der Name ist mit einer Informationsseite des Komponisten verlinkt, die von diesem Servlet generiert wird. Bei dem Aufruf wird eine ID mitgeliefert, die eindeutig den Komponisten identifiziert.

Index Dieses Servlet generiert den HTML-Code für die Startseite der Webseiten. Wenn man die Seiten *www.enotehistory.de* aufruft, wird man automatisch zu *www.enotehistory.de/index* weitergeleitet.

Log Das Servlet gibt die geloggten Informationen über den laufenden Server wieder. Diese Informationen sind nur für den Administrator zugänglich. Das bedeutet, dass man als Administrator eingeloggt sein muss.

ManAnalysis Dieses Servlet generiert den HTML-Code für das manuelle Analyse-Fenster. Ein Popup-Fenster wird erscheinen, wo der Nutzer die spezifischen Attribute der Metadaten für ein Notenblatt definieren kann. Nachdem man die manuelle Schreiberanalyse gewählt hat, wird dieses Servlet mit *www.enotehistory.de/mananalysis* aufgerufen, und das Popup-Fenster öffnet sich.

Metadata Dieses Servlet generiert den HTML-Code für eine Seite, die Informationen zu einem Musikstück, wie Titel, Komponist, Konkordanzen und einer Quellenbeschreibung gibt. Bei der Suche und bei der Navigation auf den eNoteHistory-Webseiten erhält man eine Liste mit Notenhandschriften. Die Signatur des Musikstücks leitet zu dieser Informationsseite weiter, die von diesem Servlet generiert wird. Bei dem Aufruf wird eine ID mitgeliefert, die eindeutig die Signatur identifiziert.

Navigation Dieses Servlet zeigt das Navigationsfenster, um in der eNoteHistory-Datenbank zu browsen. Dies geschieht durch den Aufruf *www.enotehistory.de/navigation*.

Pages Dieses Servlet generiert den HTML-Code, welcher das Originalbild des eingescannten Notenblattes zeigt. Bei der Suche und bei der Navigation auf den eNoteHistory-Webseiten erhält man eine Liste mit Notenhandschriften. Bei einigen Notenhandschriften kann man die eingescannten Notenblätter anschauen, welche in einer HTML-Seite eingebettet sind, die dieses Servlet erzeugt. Bei dem Aufruf wird die Signatur der Notenhandschrift als Parameter übertragen.

ScribeIdent Dieses Servlet kodiert das Menü bei der manuellen Schreiberidentifikation, das Auf- und Zusammenklappen von Menüpunkten.

Search Dieses Servlet generiert den HTML-Code für die Webseite zum Suchen in den bibliothekarischen Daten. Es wird auch verwendet, um die Suchanfrage an die eNoteHistory-Datenbank zu stellen, und die Ergebnisliste in HTML zu generieren. Dabei werden zwei Parameter, das Suchfeld und den Such-String, übergeben.

Updater Dieses Servlet generiert den HTML-Code, um Informationen in der Datenbank zu aktualisieren. StdUpdater ist eine Supportklasse dieses Servlets. Das Servlet "Updater" kann unter www.enotehistory.de/updater aufgerufen werden. Dazu muss man sich einloggen. Somit ist der Zugriff bzw. Änderungen an den Datenbestand nur für Mitarbeiter erlaubt.

Work Dieses Servlet generiert den HTML-Code, um Informationen zu einem Musikstück zu präsentieren. Bei der Suche und bei der Navigation auf den eNoteHistory-Webseiten erhält man eine Liste mit Notenhandschriften. Der Name des Musikstücks leitet zu dieser Informationsseite weiter, die von diesem Servlet generiert wird. Bei dem Aufruf wird eine ID mitgeliefert, die eindeutig das Musikstück identifiziert.

Kapitel 6

Migration in eine Content Manager Umgebung

Als anstehende Aufgabe in dem eNoteHistory-Projekt ist die Migration des entwickelten eNote-Archivs in eine digitale Bibliothek, und zwar in den IBM Content Manager. Dazu müssen Speicher- und Zugriffsrechte für die Nutzung entwickelt werden. Es wird eine Webpräsentation basierend auf den Daten des Content Managers erstellt, die das Anzeigen von Metadaten und komprimierten Bildern gestattet. Dem Experten ist der Zugriff auf spezielle Metadaten gestattet, sowie das Ändern von Daten und das Hinzufügen von neuen Notenhandschriften.

6.1 Migration in den Content Manager

In einer Diplomarbeit des Lehrstuhl Datenbank- und Informationssysteme wurde die Migration in eine Content Manager Umgebung bearbeitet. Dabei kann man die Migration in zwei Schritte unterteilen, welche auch in der Abbildung 6.1 dargestellt sind:

1. **Migration der Datenstruktur**: Falls das konzeptionelle Datenmodell nicht in der Archivdokumentation enthalten ist, wurde das konzeptionelle Datenmodell der vorhandenen Daten durch Reverse Engineering erstellt. Danach wurde das Modell in ein ICM-Datenmodell (IBM-Content-Manager-Datenmodell) konvertiert. Der genaue Algorithmus ist in der Diplomarbeit von Sebastian Dolke [Dol04] nachzulesen.

2. **Datenmigration**

   - **Datenübertragung**: Eine Java-Anwendung liest über eine JDBC-Verbindung den Inhalt der Daten aus, und überträgt sie über einen IIC-Connector in den Content Manager. Bei der Abbildung des Datenmodells wurden Documents Item Types zur Speicherung der Dokumente verwendet. Die Zuordnung der einzelnen Tabellen zu einem Item Type erfolgt mit Hilfe einer XML-Datei. Für jede Zeile einer Datenbanktabelle wird ein Item im Content Manager erzeugt.
   - **Rekonstruktion der Beziehungen**: Nachdem die Daten von der Datenbank in den Content Manager übertragen wurden, folgt die Rekonstruktion der Beziehungen zwischen den Daten. Die Beziehungen in der Datenbank wie Primärschlüssel und Fremdschlüssel wurden durch pid's eines Items im Content Manager dargestellt.

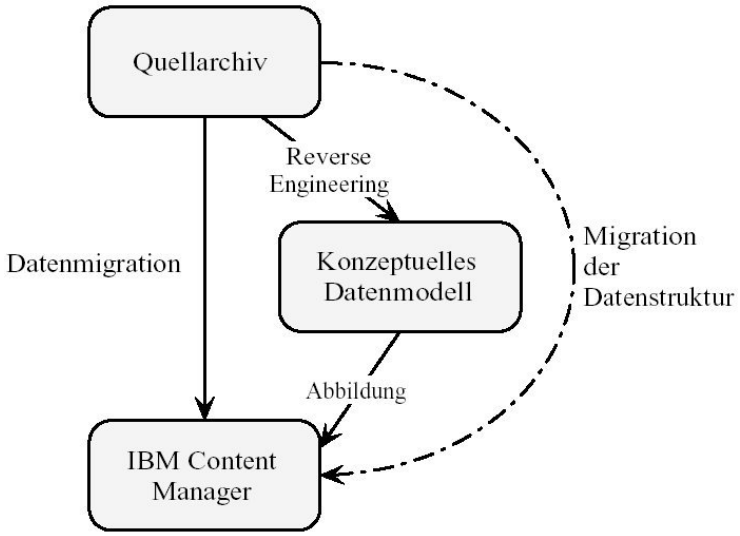


Abbildung 6.1: Migration eines Archivs in den IBM Content Manager

Migration der UDF

Anschließend erfolgt noch die Migration der Abstandsfunktion in den Library Server des IBM Content Managers. Die UDF wurde in die Library-Server-Datenbank integriert. In der UDF, welche eine Liste mit den ähnlichsten Feature-Vektoren berechnet, wird auf die Datenbanktabellen zugegriffen, um zur Berechnung notwendige Daten zu erhalten. Da durch die Migration die Datenbanktabellen in Item Types konvertiert wurden, ist dies nicht mehr möglich. Deswegen wurden alle SQL-Aufrufe in der UDF in die Anfragesprache des IBM Content Managers umgewandelt.

6.2 Architektur

In der Abbildung 6.2 sind die Architektur des Systems und der Datenzugriff bildlich dargestellt. Der Nutzer greift weiterhin über die Webseiten des eNoteHistory-Projektes zu. Diese Webseiten werden mit Java Servlets erstellt, welche auch Anfragen an die Datenbank stellen. Die Daten des eNote-Archivs befinden sind nach der Migration im IBM Content Manager. Mit Hilfe eines Administrator-Servlets kann man einstellen, an welches System die Servlets ihre Anfragen künftig stellen sollen. Die vom Servlet benötigten Anfragen sind in einer abstrakten Backend-Schnittstelle (abstract backend interface) definiert, dessen konkrete Implementierungen in den APIs der entsprechende Systeme erfolgt.

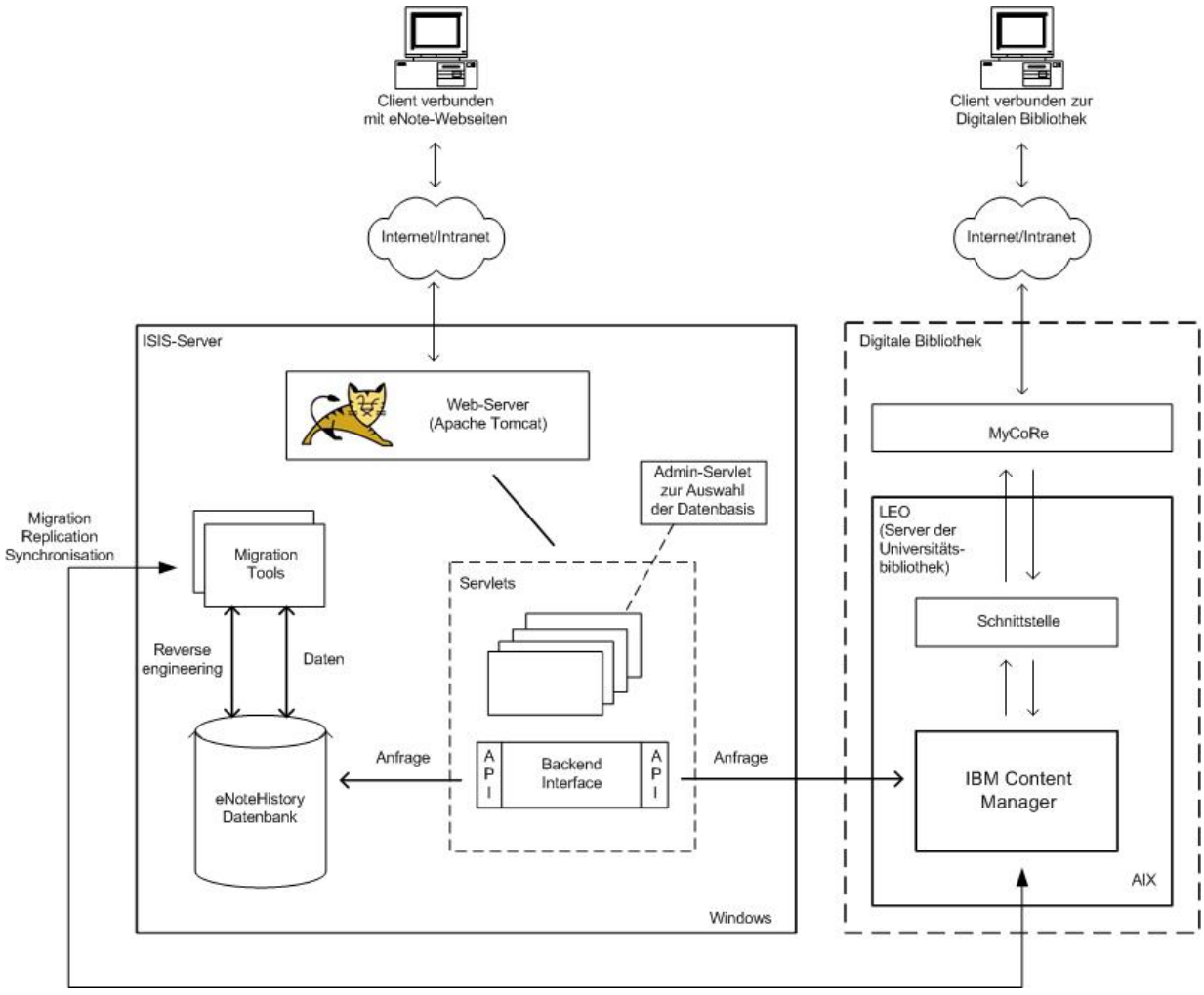


Abbildung 6.2: Migration in eine Content Manager Umgebung

Der Server der Universitätsbibliothek stellt eine Digitale Bibliothek zur Verfügung; weitere Informationen stehen unter *http://www.uni-rostock.de/ub/UB_DIGB.HTM*. Der Nutzer, der auf digitale Dokumente zugreifen möchte, wird zur MyCoRe-Schnittstelle geleitet, wel-

ches eine allgemeine Schnittstelle für Digitale Bibliotheken darstellt. Die weitere Anpassung der MyCoRe-Schnittstelle bzw. deren Zugriff auf das eNote-Archiv im Content Manager ist noch in der Entwicklung.

Das MyCoRe-Projekt

Die Universität Rostock beteiligt sich an dem MyCoRe-Projekt, dessen Ziel es ist, eine einheitliche Infrastruktur einer Digitalen Bibliothek zu bieten. MyCoRe ist ein Open Source Projekt (GNU General Public License) zur Entwicklung eines Systems für Digitale Bibliotheken und Archivlösungen. In diesem Projekt arbeiten Universitäten zusammen, um für derartige Anwendungen einen gemeinsamen Software-Kern zu erstellen. Dieser Kern kann entsprechend den eigenen Bedürfnisse angepasst bzw. erweitert werden. Die Basis bilden Java-Klassenbibliotheken, XML-Techniken und unterschiedliche Datenbank-Backends, sowie IBM DB2 und IBM Content Manager, als auch MySQL und XMLDB-Datenbanken. Weitere Informationen zum MyCoRe-Projekt findet man unter *http://www.mycore.de*.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Dol04] Sebastian Dolke. Umsetzung von Datenmodellen und Methoden bei der Integration spezieller Dokumentenserver in eine IBM-Content-Manager-Umgebung. Master's thesis, Universität Rostock, Institut für Informatik, 2004.

[Mil04] Lars Milewski. Integration von Clustering-/Classification-Techniken in eine objektrelationale Datenbankumgebung. Master's thesis, Universität Rostock, Institut für Informatik, 2004.