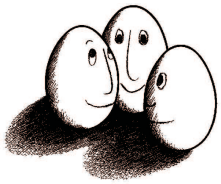
Diplomarbeit

Fallstudie zur multi-relationalen Aufbereitung für die Wissensentdeckung in Datenbanken

Holger Flick



Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

Montag, 14. August 2006

Betreuer:

Prof. Dr. Katharina Morik
Dipl.-Inform. Martin Scholz

'Information is not knowledge.'
- Albert Einstein

Danksagung

Ein herzliches Dankeschön an alle, die mir bei der Erstellung der vorliegenden Arbeit helfend zur Seite gestanden haben. Mein besonderer Dank gilt meinen Betreuern Prof. Dr. Katharina Morik und Dipl.-Inform. Martin Scholz. Beide haben sich immer die Zeit genommen, geduldig meine vielen Fragen zu beantworten. Weiterhin gilt besonderer Dank Dipl.-Inform. Timm Euler, der mich bei Fragen und Vorschlägen zu MiningMart jederzeit unterstützt hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Computer bieten Zugang zu einer riesigen Menge von Informationen. Der lokale Speicherplatz auf Festplatten innerhalb eines Computersystems scheint bereits nahezu beliebig groß und bedingt durch das Internet wächst das zur Verfügung stehende Datenmaterial ins Unendliche. Ein Problem, das bisher immer noch nicht erfolgreich gelöst wurde, ist, verwertbares Wissen aus all diesen Informationen zu extrahieren. Aus diesem Grund werden seit einigen Jahren unter dem Begriff 'Wissensentdeckung in Datenbanken' zahlreiche Methoden und Systeme zusammengefasst, die dazu dienen sollen, neuartiges und unter Umständen verborgenes Wissen in großen Datenbeständen zu entdecken. Zu nennen sind hier vor allem die Data Mining-Verfahren, die Methoden aus den Bereichen Statistik, Maschinelles Lernen, Datenbanksysteme und Visualisierung miteinander verbinden. Durch den geschickten Einsatz von Ansätzen aus all diesen Bereichen soll es möglich sein, neues Wissen in großen Datenbeständen zu finden.

Der Erfolg dieser Verfahren hängt jedoch stark von der Qualität der zur Verfügung stehenden Daten ab und nicht nur von den gewählten Methoden zum Auffinden des Wissens.

In dieser Arbeit werden die Daten der Internet Movie Database zunächst analysiert und aufbereitet. Anschließend wird exemplarisch ein Lernverfahren zur Extraktion von Wissen auf den aufbereiteten Daten durchgeführt. Der Schwerpunkt der Arbeit liegt dabei nicht auf den technischen Aspekten zur effizienten Bearbeitung oder dem Erreichen möglichst interessanter Lernergebnisse, es geht vielmehr darum, den Prozess der Datenvorverarbeitung systematisch und transparent durchzuführen, um daraus eine Modellierung abzuleiten, die die Daten auf einer konzeptuellen Ebene präsentiert. Es soll gezeigt werden, dass eine solche Repräsentation der Daten die Durchführung des angestrebten Lernprozesses erleichtern kann, obwohl zunächst mehr Zeit in die Vorbereitung zu investieren ist.

Als Beispiel wird die Thematik des Fallbasierten Schließens zentral behandelt, die es sich zum Ziel gesetzt hat, einmalig entwickelte Schritte erneut auf ähnliche Probleme anzuwenden.

Bevor jedoch das Fallbeispiel diskutiert werden kann, sind grundlegende Aspekte zu definieren und zu erläutern.

Es wird dazu zuerst in Kapitel 2 der Begriff der Wissensrepräsentation definiert. Anschließend werden Datenbanksysteme eingeführt und ausführlich erläutert, da sie in dieser Arbeit dazu dienen, das repräsentierte Wissen zu speichern. Dazu werden gebräuchliche Datenbankmodelle und die Sprache SQL vorgestellt.

Kapitel 3 beschäftigt sich mit Ontologien. Zunächst wird der Begriff der Ontologie motiviert, definiert und für die Informatik genau abgegrenzt. Neben der Modellierung und Strukturierung von Ontologien werden deskriptive Logiken besprochen. Deskriptive Logiken stellen einen logikbasierten Formalismus der Wissensrepräsentation dar. Es werden auch Entity-Relationship-Diagramme als eine weitere Modellierungstechnik vorgestellt. Abschließend werden weitere Beschreibungssprachen und das semantische Web dargestellt.

Die Wissensentdeckung in Datenbanken wird dann in Kapitel 4 definiert. Darüber hinaus wird das CRISP-DM Modell erläutert, welches ein Data Mining-Projekt in verschiedene Phasen einteilt. Diese Phasen werden im Detail erläutert.

Als ein Beispiel für multi-relationale Lernverfahren wird dann in Kapitel 5 das Lernverfahren RDT eingeführt. Dazu werden die Induktive Logische Programmierung definiert und einige logische Grundlagen vermittelt. Der Entwicklungsprozess des Lernverfahrens RDT zu RDT/DM, welches in dieser Diplomarbeit zur Durchführung der Experimente verwendet wurde, wird ebenfalls geschildert.

Die Software MiningMart, mit der unter anderem die Modellierung in dieser Arbeit durchgeführt wurde, wird in Kapitel 6 vorgestellt. Dazu wird zum einen eine Übersicht über Programmfunktionen gegeben, zum anderen werden aber auch technische Aspekte und die Grundidee zur Erstellung von MiningMart erläutert.

Nachdem alle Grundlagen für das bearbeitete Fallbeispiel in den vorangehenden Kapiteln vorgestellt wurden, wird in Kapitel 7 anhand des CRISP-DM Modells eine Modellierung der Daten einer öffentlichen Datenbank des Internets, der Internet Movie Database, vorgenommen. Das Kapitel ist dabei nach den einzelnen Phasen des CRISP-DM Modells gegliedert.

Für das zuvor entwickelte Modell wird dann in Kapitel 8 gezeigt werden, dass die Anwendung eines multi-relationalen Lernverfahrens auf den aufbereiteten Daten problemlos möglich ist. Weiterhin soll gezeigt werden, dass die Modellierung die Durchführung erleichtert.

Den Abschluss dieser Arbeit bildet eine Zusammenfassung in Kapitel 9 über die vorgestellten Aspekte und neuen Erkenntnisse.

Der Anhang enthält zahlreiche Informationen, um die durchgeführten Versuche nachvollziehen zu können.

2. Wissensrepräsentation

Der Begriff Wissensrepräsentation umfasst die formale Darstellung von Wissen eines wissenbasierten Systems. In dieser Arbeit wird das Wissen in Datenbanken abgelegt. Daher werden im folgenden Abschnitt Datenbanksysteme ausführlich erläutert. Dabei wird nicht nur der Begriff der Datenbank von einem Datenbanksystem abgegrenzt, sondern vielmehr der Aufbau einer Datenbank formal aufgezeigt. Dazu wird das relationale Modell vorgestellt, welches die Basis von vielen aktuellen Datenbanksystemen bildet.

Weiterhin wird darauf eingegangen, wie Wissen durch Einsatz der Sprache SQL in Datenbanksysteme eingefügt bzw. abgerufen werden kann. Im Abschluss werden Kriterien der Repräsentationsgüte diskutiert. So werden z.B. schon für Datenbanksysteme Methoden vorgestellt, die die Effizienz erhöhen und die Komplexität des Systems dabei überschaubar halten. Dies bildet dann eine solide Grundlage für die weiteren Modellierungsschritte in den folgenden Kapiteln.

2.1. Datenbanksysteme

Dieser Abschnitt wird die grundlegenden Begriffe zu Datenbanksystemen definieren. Dabei wird im Unterabschnitt 2.1.3 das relationale Modell vorgestellt, welches die Basis für relationale Datenbanken bildet. Das Modell beschränkt sich dabei nicht nur auf die Beschreibung der Daten, sondern wird auch die Manipulation der Daten ermöglichen. In der Praxis wird jedoch die Datenbeschreibungs- und Datenmanipulationssprache SQL verwendet, die auf diesen Theorien basiert und schrittweise im Detail aus dem relationalen Modell in Abschnitt 2.1.5 hergeleitet wird.

Als Leitfaden bei der Zusammenstellung dieses Kapitels dienten vor allem die Bücher von Date [14] und Biskup [4], die einen sehr guten Überblick über die gesamte Thematik geben.

2.1.1. Begriff der Datenbank

Der Begriff Datenbank wird häufig synonym für ein gesamtes Datenbanksystem verwendet. Abbildung 2.1 zeigt den Aufbau eines Datenbanksystems. Es besteht aus dem Datenbankmanagementsystem (DBMS) und der Datenbank. Dabei ist die Datenbank in diesem Modell einfach nur ein Datenspeicher, der von dem Datenbankmanagementsystem gesteuert wird. Sie ist nach [33] eine Sammlung von nicht-redundanten Daten. Benutzer einer Datenbank greifen immer über das Datenbankmanagementsystem auf eine Datenbank zu, d.h. vom eigentlichen Datenspeicher wird abstrahiert. Der nächste Abschnitt beschreibt den genauen Aufbau des Datenbankmanagementsystems.

2.1.2. Architektur

Es wird im folgenden Unterabschnitt 2.1.2 die NSI/SPARC-Architektur von Datenbanksystemen vorgestellt. Im Anwendungsteil, Kapitel 7, wird deutlich werden, dass die

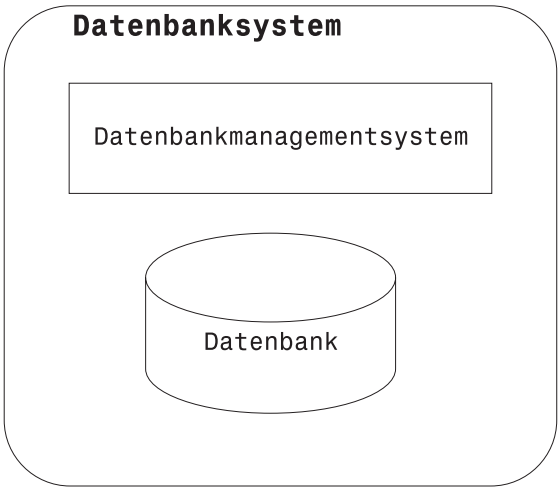


Abbildung 2.1.: Aufbau eines Datenbanksystems (vgl. Abschnitt 2.1.1)

Kenntnis dieses Schemas für die Bearbeitung der Aufgabenstellung und das Verständnis der vorgestellten Lösung unabdingbar ist. Der zweite Teil dieses Abschnittes wird durch die genauere Betrachtung der Komponenten eines Datenbankmanagementsystems gebildet.

Ebenen der Architektur

Nach der ANSI/SPARC Architektur gibt es genau drei Ebenen: die interne, externe und konzeptuelle Ebene. Abbildung 2.2 zeigt die hierarchische Anordnung der Ebenenen. Die interne Ebene befindet sich direkt über der physikalischen Speicherung, d.h. diese Ebene ist dafür verantwortlich, die Daten auf einem Computer abzuspeichern. Der Benutzer kommt mit dieser Ebene nicht in Kontakt, ganz im Gegensatz zu der externen Ebene. Aufgabe dieser Ebene ist es, Möglichkeiten anzubieten, wie einem Benutzer die Daten präsentiert bzw. wie diese gesichtet werden können. Die konzeptuelle Ebene verbindet diese beide Ebenen miteinander. Es gilt zu beachten, dass die externe Ebene individuell für die Benutzer des Systems Betrachtungsmöglichkeiten auf das Systems bietet, wogegen die konzeptuelle Ebene eine allgemeine, durch die Gemeinschaft geprägte Ansicht darstellt. Es gibt also genau eine konzeptuelle Repräsentation und beliebig viele (in Abbildung 2.2 durch gestrichelte Linien dargestellt) externe Repräsentationen. Genauso gibt es genau eine interne Ebene, die repräsentiert, wie die Daten physikalisch im Computersystem gespeichert werden. Es ist hervorzuheben, dass die hier angesprochene 'konzeptuelle' Ebene nicht mit der konzeptuellen Ebene der Datenbankmodellierung zu verwechseln ist. Es wird zwar der selbes Begriff des Konzeptes verwendet, jedoch diskutiert man hier über die Architektur eines *Datenbanksystems* und in den folgenden Abschnitten über die Modellierung einer *Datenbank*.

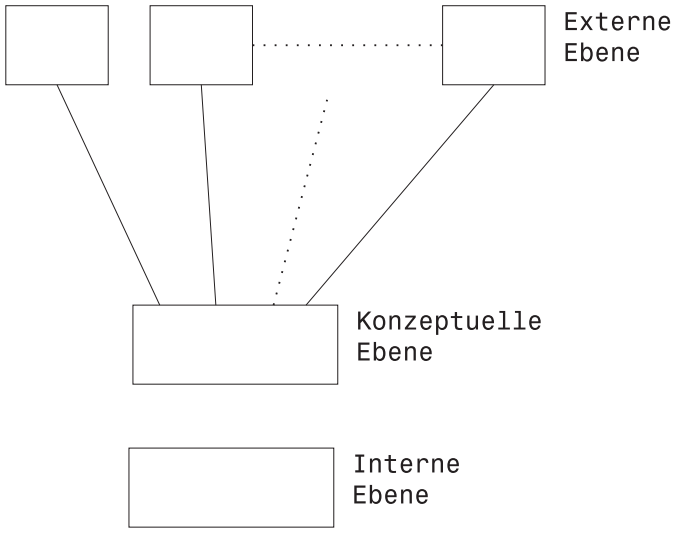


Abbildung 2.2.: Drei Ebenen der Architektur nach ANSI/SPARC (vgl. Abschnitt 2.1.2)

Komponenten des Datenbankmanagementsystems

Datenbankmanagementsysteme werden in drei Bestandteile unterteilt: Die Anwendungsumgebung, das Informationssystem und das Basissystem. Die Pfeile in Abbildung 2.3 zeigen, welche Komponenten miteinander kommunizieren. Die Anordnung der Komponenten in der Abbildung ist nicht willkürlich, sondern visualisiert die Abstraktion. So ist die Anwendungsumgebung der Bestandteil, mit dem der Benutzer in Kontakt tritt. Die Anwendungsumgebung kapselt die Entwurfsumgebung, die haupsächlich zur Administration des Systems verwendet wird, die Programmierumgebung, die eine Anbindung einer höheren Programmiersprache an die Funktionalität des gesamten Systems bereitstellt, und die Arbeitsplatzumgebung, die dem Endbenutzer z.B. eine Benutzeroberfläche zur interaktiven Erstellung von Anfragen bietet. Für den Benutzer ist die nächste Ebene, das Informationssystem, verdeckt. Das Informationssystem ist dafür verantwortlich, die Anfragen der Anwendungsumgebung zu interpretieren und zu verarbeiten und die konzeptuelle Schicht des Systems zu realisieren. In dieser Komponente sind auch die Transaktionsverwaltung und die Meta-Informationen, z.B. die Relationenschemata, zu den in der Datenbank gespeicherten Daten zu finden. Zuletzt ist das Informationssystem in der Lage, die Zugriffe auf das Basissystem abzubilden, d.h. Zugriffe auf die konzeptuelle Ebene können im darunter liegenden Basissystem abgerufen werden. Das Basisystem, meist das Betriebssystem, stellt dem Informationssystem die Funktionalität zum Datenzugriff und zur Datenablage zur Verfügung. Dabei gibt es sowohl flüchtige als auch dauerhafte Speicher. Das Basissystem nutzt z.B. für Anfragen der konzeptuellen Ebene den flüchtigen Speicher, um 'gleichartige' Anfragen schneller beantworten zu können. Im Basissystem werden die Tupel einer Relation nur noch als Daten betrachtet und nicht, wie im Informationssystem und darüber, als Datensätze.

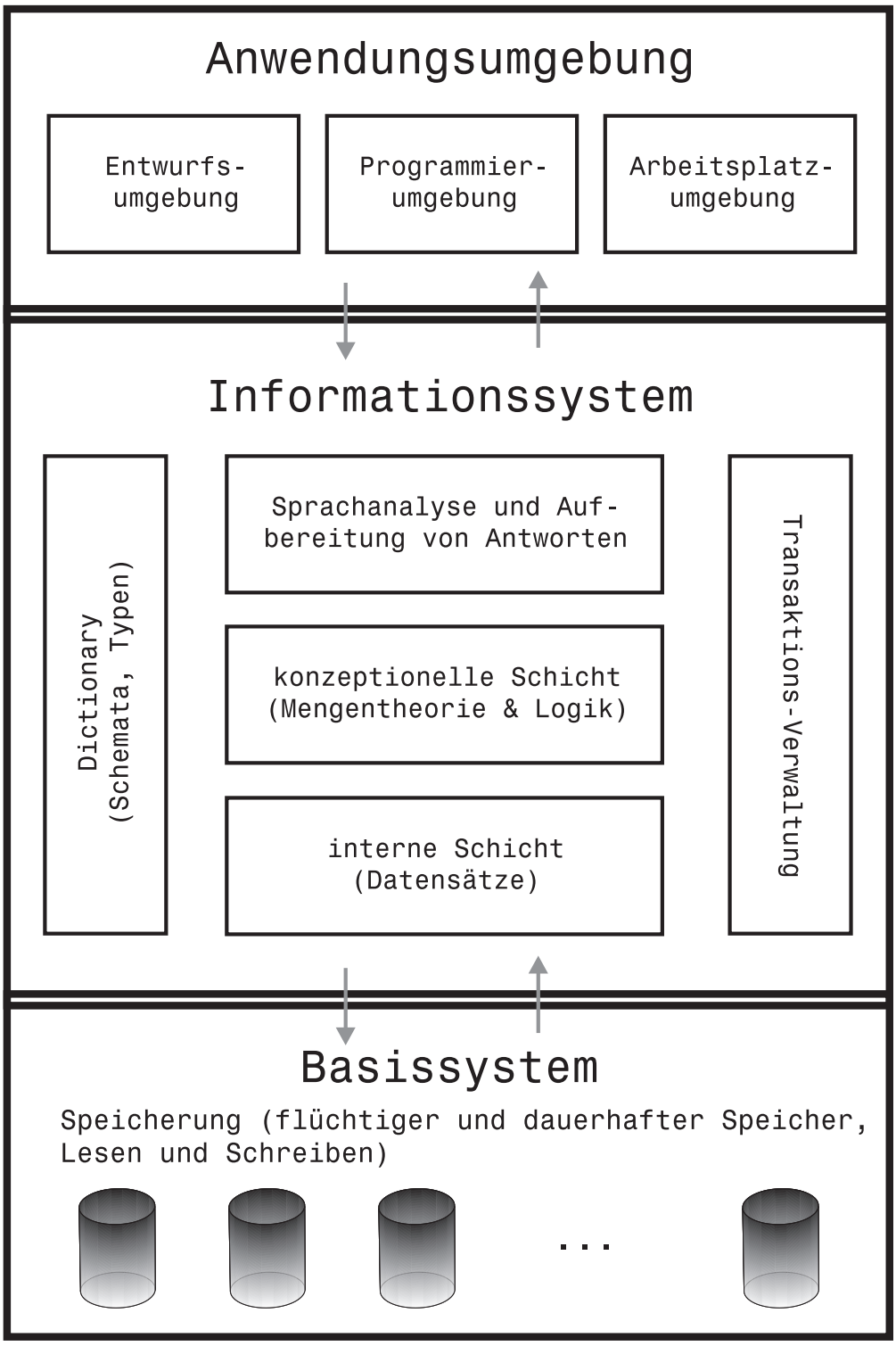


Abbildung 2.3.: Komponenten des Datenbankmanagementsystems (vgl. Abschnitt 2.1.2)

2.1.3. Relationales Modell

Die Arbeit benutzt Datenbanksysteme, die auf dem sogenannten relationalen Modell aufbauen. Es geht auf den Wissenschaftler Edgar F. Codd [64] zurück, der im Jahr 1970 die erste Veröffentlichung zu einem relationalen Modell für Datenbanksysteme machte [9]. Eine Erweiterung zu diesem Modell wurde dann 1979 bekannt [11]. Abschließend schrieb er ein Buch zu diesem Thema [12], welches 1990 erschien.

Der folgende Abschnitt wird die in den Veröffentlichungen genannten Begriffe definieren und erläutern.

Terminologie

Für die weitere Betrachtung ist es wichtig, dass einige Begriffe klar definiert werden, da viele dieser Begriffe mehrdeutig sind. In den folgenden Kapiteln werden die Begriffe genau in dem Zusammenhang wie hier angegeben verwendet.

Definition 2.1 (Attribut)**.** *Ein Attribut bezeichnet eine abstrakte Eigenschaft, bezogen auf eine Relation. Es hat einen Bezeichner, der es innerhalb einer Relation eindeutig identifiziert. Dazu besitzt jedes Attribut genau eine Domäne und Attributwerte.*

Definition 2.2 (Attributwert)**.** *Unter einem Attributwert versteht man eine konkrete Eingenschaft bzw. Belegung eines Attributes.*

Beispiel 2.1 (Attribut, Attributwert)**.** *Das Attribut mit dem Bezeichner 'name' hat den Attributwert 'Schmidt'.*

Definition 2.3 (Domäne)**.** *Eine Domäne wird einem oder mehreren Attributen zugeordnet. Sie bezeichnet die Menge aller möglichen Werte, die das Attribut annehmen kann.*

Beispiel 2.2 (Domäne)**.** *Das Attribut 'name' aus Beispiel 2.1 hat die Domäne 'Nachnamen'. Das Datenbankmanagementsystem bildet die Domänen innerhalb der Datenbank auf Systemebene auf Datentypen ab.*

Definition 2.4 (Relationenschema)**.** *Eine endliche Menge von Attributen mit zugeordneten Domänen heißt Relationenschema.*

Beispiel 2.3.

$$RS := \{(\text{vorname}, \text{ALPHABET}), (\text{nachname}, \text{ALPHABET})\}$$

Das Relationenschema RS beinhaltet zwei Attribute, die als Vektor mit genau zwei Elementen angegeben sind. Die Vektoren bestehen aus dem Bezeichner für das Attribut und der dazugehörigen Domäne. Beide Domänen werden von dem Datenbankmanagementsystem als Zeichenkette auf Systemebene dargestellt.

Definition 2.5 (Relation)**.** *Eine Relation ist eine Menge von Tupeln, die auch als Datensatz bezeichnet werden. Ihr wird ein Relationenschema zugeordnet. Die Ordnung einer Relation ist über die Anzahl der Attribute des zugehörigen Relationenschemas definiert.*

Beispiel 2.4. *Eine Relation mit dem Namen R, basierend auf dem Relationenschema aus Beispiel 2.3, könnte formal z.B. folgendermaßen aussehen:*

$$R := \{(Schmidt, Heinz), (Mueller, Horst), (Becker, Willi)\}$$

R:

nachname	**vorname**
Schmidt	Heinz
Mueller	Horst
Becker	Willi

Tabelle 2.1.: Darstellung der Relation aus Beispiel 2.4 als Tabelle

In der Literatur (z.B. [4]) wird häufig das Relationenschema zusammen mit der Relation in Tupelschreibweise angegeben. Weiterhin werden Relationen meist mit Tabellen gleichgesetzt, da sich Tabellen sehr gut zur Darstellung von Relationen mit den dazugehörigen Attributbezeichnern darstellen lassen. Tabelle 2.1 zeigt die Relation aus Beispiel 2.4 als Tabelle. Die Anzahl der Tupel einer Relation bezeichnet man als Kardinalität und die Anzahl der Attribute bestimmen den Grad der Relation. Alle Begriffe sind in Abbildung 2.4 anschaulich dargestellt.

Definition 2.6 (Schlüssel)**.** *Eine Attributmenge A wird als Schlüssel bezeichnet, wenn gilt:*

- *Alle Attributwerte dieser Attributmenge sind eindeutig.*
- *Mittels A kann jedes Tupel in der Relation eindeutig bestimmt werden.*
- *Wird ein Attribut aus der Attributmenge entfernt, so bildet diese neue Attributmenge keinen Schlüssel mehr.*

Relationen wird immer mindestens ein Schlüssel zugewiesen. Ein spezieller Schlüssel ist hierbei der Primärschlüssel (*foreign key*), der speziell anzugeben ist. In der Tabellendarstellung werden Attribute, die zum Primärschlüssel dazugehören, unterstrichen. Es gibt oftmals mehrere Möglichkeiten den Schlüssel zu wählen. Daher bezeichnet man die Menge aller möglichen Schlüssel mit Schlüsselkandidaten. Einen von diesen Schlüsselkandidaten wählt man als den sogenannten Primärschlüssel aus.

Der Primärschlüssel wird in der Praxis meist mit möglichst kleinen Attributmengen gewählt, um Zugriffszeiten zu optimieren. Einige Datenbanksysteme erfordern nicht die Angabe eines Primärschlüssels, da diese dann intern einen eigenen Schlüssel (z.B. über ein zusätzliches Attribut, welches eine Identifikationsnummer enthält) verwalten. Viele Datenbanksysteme wählen die Anordnung der Tupel innerhalb der Relation standardmäßig gemäß dem Primärschlüssel. Dies spiegelt sich auch bei der Ausgabe von Anfragen wider, sofern der Benutzer explizit keine andere Sortierung wünscht.

Definition 2.7 (Fremdschlüssel)**.** *Man bezeichnet eine Attributmenge einer Relation als Fremdschlüssel (foreign key), wenn eine Attributmenge mit gleicher Domäne existiert, die in einer anderen Relation ein Primärschlüssel ist.*

Beispiel 2.5. *Betrachtet man in Tabelle 2.5 die Relation 'spieler', so ist die Attributmenge des Attributes 'team' ein Fremdschlüssel. Die Relation 'mannschaft' besitzt nämlich eine Attributmenge mit demselben Namen und derselben Domäne als Primärschlüssel. Es hat sich die Sprechweise durchgesetzt zu sagen, dass die Relation 'spieler' den Primärschlüssel der Relation 'mannschaft' referenziert.*

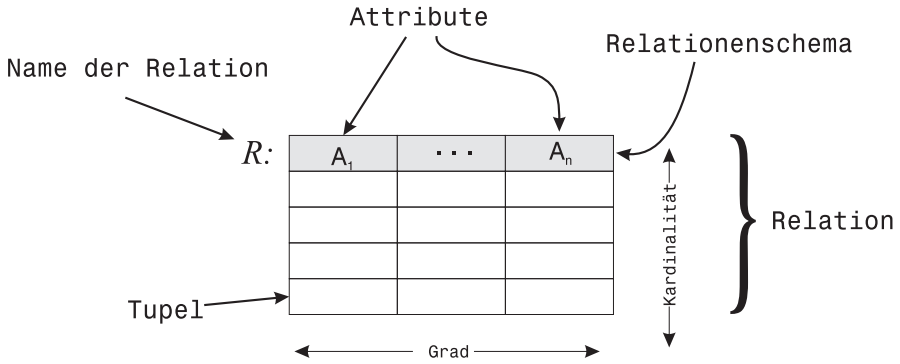


Abbildung 2.4.: Das Schema veranschaulicht ein Relationenschema mit allen assoziierten Begriffen, wie sie in [12, 11, 9] definiert werden.

formaler Begriff	**englischer, formaler Begriff**	**informelles Äquivalent**
Relationenschema	schema	Struktur, Aufbau
Relation	relation	Tabelle
Tupel	tuple	Zeile, Datensatz
Kardinalität	cardinality	Anzahl der Datensätze
Attribut	attribute	Spalte, Feld
Attributwert	attribute value	Wert
Domäne	domain	Pool der legalen Werte
Grad	degree	Anzahl der Spalten
Primärschlüssel	primary key	eindeutiger Bezeichner

Tabelle 2.2.: Relationales Modell: Übersicht der Terminologie

Fremdschlüssel spielen in den Modellen des Praxisbereiches eine besondere Rolle. Man benutzt Fremdschlüssel in der Praxis, um redundante Informationen zu vermeiden bzw. die Daten innerhalb eines Datenbanksystems zu strukturieren. Ein konkretes Beispiel wird in Abschnitt 2.1.3 vorgestellt, wenn die primitiven Operationen der relationalen Algebra bekannt sind.

Informelle Terminologie

Die im letzten Abschnitt eingeführten Begriffe werden oftmals durch bildlichere Begriffe ersetzt. So spricht man bei einer Relation auch von einer Tabelle, die aus Zeilen, den Datensätzen, und Spalten, den Feldern, besteht. Auch für eine Vielzahl anderer Begriffe existieren informelle Begriffe. Eine Übersicht in Anlehnung an [14] ist in Tabelle 2.2 einzusehen.

Relationenalgebra

Die Relationenalgebra ist eine formale Sprache, mit der sich Anfragen über einem Relationenschema ausdrücken lassen. Es ist auch möglich, mehrere Relationen miteinander

Operation	Verknüpfung	Beispiel
Vereinigung	$\cup$	$R \cup S$
Differenz	$-$	$R - S$
Durchschnitt	$\cap$	$R \cap S$
Kartesisches Produkt	$\times$	$R \times S$
Projektion	π	$\pi_{\delta}(R)$
Selektion	σ	$\sigma_{\theta}(R)$
Verbund	$\bowtie$	$R \bowtie_{\theta} S$
Umbenennung	β	$\beta_{\delta \to \gamma}(R)$

Tabelle 2.3.: Notation Operationen der Relationalen Algebra (vgl. [4]). In den Beispielen sind R, S Relationen, θ eine Bedingung und δ eine Menge von Attributen (ausführliche Beispiele in Abschnitt 2.1.3 auf Seite 7).

zu verknüpfen und somit komplexere Informationen herzuleiten.

Die Relationenalgebra ist abgeschlossen, da alle Operationen auf Relationen angewendet werden und auch Relationen als Ergebnis liefern. Weiterhin sind die Ergebnismengen endlich. Um Mengenoperationen anwenden zu können, müssen die Relationen miteinander kompatibel bzw. vereinigungsverträglich sein.

Definition 2.8 (Vereinigungsverträglichkeit von Relationen)**.** *Zwei Relationen A und B bezeichnet man als vereinigungsverträglich genau dann, wenn*

- *A und B den gleichen Grad haben*
- *die Attributbezeichnungen von A und B übereinstimmen*
- *die Domänen der Attribute beider Relationen identisch sind.*

Beispiel 2.6. *Die Relationen R_S und R_T in Abbildung 2.5 sind nicht vereinigungsverträglich, da ihr Grad nicht gleich ist.*

Tabelle 2.3 zeigt eine Liste mit allen Operationen in formaler Schreibweise. Deren Definitionen sind [4] zu entnehmen. In den folgenden Unterabschnitten werden einige Operationen anhand von Beispielen genauer betrachtet und erläutert. Die Mengenoperationen (Vereinigung, Differenz und Durchnitt) sind leicht verständlich und anhand der formalen Schreibweise nachzuvollziehen. Sie werden daher nicht genauer erläutert.

Kartesisches Produkt

Projektion Bei der Projektion wird die Attributmenge auf eine Teilmenge von Attributen beschränkt. Die Anzahl der Tupel einer Relation bleibt unverändert. Tabelle 2.7 enthält ein Beispiel für eine Projektion. Die Relation '$mannschaft$' wird auf ein Attribut eingeschränkt.

Selektion Die Selektion schränkt die Menge der Tupel innerhalb der Relation ein, die Anzahl der Attribute ändert sich nicht. In Tabelle 2.7 ist ein Beispiel zu sehen. Dort wird die Relation '*spieler*' genau auf die Spieler beschränkt, die älter als 32 Jahre alt sind.

$R_S := spieler$:

name	**vorname**	**alter**	**team**
Adams	Russ	24	TOR
Catalanotto	Frank	31	TOR
Cora	Alex	29	BOS
Damon	Johnny	31	BOS
Hill	Aaron	23	TOR
Hinske	Eric	27	TOR
Koskie	Orlando	27	TOR
Millar	Kevin	33	BOS
Mueller	Bill	34	BOS
Nixon	Trot	31	BOS
Ortiz	David	29	BOS
Ramirez	Manny	33	BOS
Renteria	Edgar	29	BOS
Rios	Alexis	24	TOR
Varitek	Jason	33	BOS
Wells	Vernon	26	TOR
Zaun	Gregg	34	TOR

$R_M := mannschaft$:

team	**franchise**	**stadt**
TOR	Blue Jays	Toronto
BOS	Red Sox	Boston
NYY	Yankees	New York
NYM	Mets	New York

Abbildung 2.5.: Relationen *spieler* und $mannschaft$. In den formalen Beispielen werden die Relationen mit R_S bzw. R_M bezeichnet.

R:

a	**b**	**c**	**d**
1	2	3	4
4	5	6	7
7	8	9	0

S:

e	**f**	**g**
1	2	3
7	8	9

$R \times S$:

a	**b**	**c**	**d**	**e**	**f**	**g**
1	2	3	4	1	2	3
4	5	6	7	1	2	3
7	8	9	0	1	2	3
1	2	3	4	7	8	9
4	5	6	7	7	8	9
7	8	9	0	7	8	9

Abbildung 2.6.: Beispiel für das Kartesische Produkt von zwei Relationen. Das Beispiel verdeutlicht unter anderem, dass die Ergebnisrelation dieser Operation eine grosse Anzahl an Tupeln und Attributen erhält, obwohl die Ausgangsrelationen eine recht überschaubare Anzahl besitzen. Diese Problematik wird in Beispiel 2.7 auf der nächsten Seite aufgegriffen.

Verbund Im Einleitungstext zu diesem Abschnitt wurde angesprochen, dass die Struktur einer Datenbank die Güte der Wissensrepräsentation beeinflusst. Der Verbund ist eine wichtige Operation, da man mit ihm zwei Relationen miteinander verknüpfen kann. Dies ist vor allem dann möglich, wenn eine Relation einen Fremdschlüssel enthält. Dann kann durch den Verbund eine Relation gebildet werden, die die Informationen der durch den Fremdschlüssel verknüpften Relation ebenfalls enthält. Das Ergebnis des Verbundes der Relationen aus Tabelle 2.5 ist in Tabelle 2.4 zu sehen. Das Attribut 'team' ist ein Fremdschlüssel in der Relation 'spieler', da es in der Relation 'mannschaft' ein Primärschlüssel

$\pi_{franchise}(R_M)$:

franchise
Blue Jays
Red Sox
Yankees
Mets

$\sigma_{alter \geq 33}(R_S)$:

name	**vorname**	**alter**	**team**
Millar	Kevin	33	BOS
Mueller	Bill	34	BOS
Ramirez	Manny	33	BOS
Varitek	Jason	33	BOS
Zaun	Gregg	34	TOR

Abbildung 2.7.: Projektion (links) und Selektion (rechts) der Relationen Algebra: Die Projektion auf die Relation *mannschaft* beschränkt die Relation auf das Attribut 'franchise', die Anzahl der Tupel bleibt gleich. Die Selektion hingegen verändert die Attributmenge nicht, sondern beschränkt die Tupelmenge auf genau die Tupel, deren Attributwert im Attribut 'alter' größer oder gleich '33' ist.

ist. Duch einen Verbund können wir die zusätzlichen Informationen zu einer Mannschaft erschließen. Es ist nicht mehr notwendig, zu jedem Spieler in der Relation 'spieler' die genauen Informatinen zu einer Mannschaft zu speichern. Dazu spielen mehrere Spieler in der gleichen Mannschaft, d.h. einige Tupel aus der Relation 'spieler' werden mit dem selben Tupel aus der Relation 'mannschaft' verbunden. Es ist offensichtlich, dass die Relationen so nicht nur übersichtlicher, sondern auch wesentlich effizienter im Bezug auf ihre Größe sind.

Einen Spezialfall des Verbunds bildet der **Natürliche Verbund**. Bei dieser Operation werden zwei Relationen über ihre gleichnamigen Attribute verknüpft. Als Operator im logischen Ausdruck θ wird dabei '=' verwendet, d.h. es werden identische Attributwerte verknüpft. Hervorzuheben ist, dass der Natürliche Verbund auch dann definiert ist, wenn es keine gemeinsamen Attribute in den beiden Relationen gibt. In diesem Falle ist das Ergebnis des Natürlichen Verbundes das kartesische Produkt.

Beispiel 2.7. *Sind wir an den Franchise-Bezeichnungen aller Spieler im Alter von 34 Jahren interessiert, so könnten wir formulieren:*

$$def.\ \theta := (R_\sigma.team = R_M.team)$$
$$Anfrage:\ \pi_{franchise}(\sigma_{alter=34} \bowtie \theta R_M)$$

R_σ bezeichne dabei die Relation, die durch die Selektion auf R_S entsteht.

Moderne Datenbanksysteme wie MySQL[1] oder Oracle[2] führen Operationen wie in Beispiel 2.7 in einem 'Schritt' aus und erzeugen genau eine Ausgabe pro Anfrage. Allerdings ist vor allem bei der Arbeit mit KDD-Werkzeugen wie z.B. MiningMart (vgl. Kapitel 6) zu beachten, dass dort Zwischenergebnisse nicht verloren gehen und man daher die Reihenfolge der Operationen überlegt wählen sollte. So ist es zum Beispiel anzuraten, eine Selektion vor einem Verbund durchzuführen, da der Verbund dann auf weniger Tupeln durchzuführen ist und auch die Zwischenergebnisse wesentlich übersichtlicher sind.

[1] URL: `http://www.mysql.com`
[2] URL: `http://www.oracle.com`

$R_S \bowtie_{R_S.team=R_M.team} R_M$:

name	vorname	alter	team	franchise	stadt
Adams	Russ	24	TOR	Blue Jays	Toronto
Catalanotto	Frank	31	TOR	Blue Jays	Toronto
Cora	Alex	29	BOS	Red Sox	Boston
Damon	Johnny	31	BOS	Red Sox	Boston
Hill	Aaron	23	TOR	Blue Jays	Toronto
Hinske	Eric	27	TOR	Blue Jays	Toronto
Koskie	Orlando	27	TOR	Blue Jays	Toronto
Millar	Kevin	33	BOS	Red Sox	Boston
Mueller	Bill	34	BOS	Red Sox	Boston
Nixon	Trot	31	BOS	Red Sox	Boston
Ortiz	David	29	BOS	Red Sox	Boston
Ramirez	Manny	33	BOS	Red Sox	Boston
Renteria	Edgar	29	BOS	Red Sox	Boston
Rios	Alexis	24	TOR	Blue Jays	Toronto
Varitek	Jason	33	BOS	Red Sox	Boston
Wells	Vernon	26	TOR	Blue Jays	Toronto
Zaun	Gregg	34	TOR	Blue Jays	Toronto

Tabelle 2.4.: Verbund der Relationen *spieler* und *mannschaft*. Die Relationen werden über das gemeinsame Attribut *team* verknüpft. Somit erhält die Relation *spieler* die Zusatzinformation über den Namen des Franchises und die Heimstadt der Mannschaft.

2.1.4. Weitere Modelle

Neben der relationalen Sichtweise gibt es weitere Modellierungsansätze, wie z.B. die Entity-Relationship-Diagramme, die jedoch erst in Abschnitt 3.5 erläutert werden, da sie eher der Modellierung angehören als der Wissensrepräsentation. Dort werden das Netzwerk-Modell und das Entity Set Model kurz mit den E/R-Diagrammen verglichen. Dies wird zeigen, warum in dieser Arbeit das relationale Modell und E/R-Diagramme zur Modellierung benutzt wurden.

2.1.5. Datenabfrage und Datenmanipulation mit SQL

SQL ist eine deklarative Datenabfrage und Datenmanipulationssprache mit der Benutzer mit dem Datenbankmanagementsystem kommunizieren können. In [6] wurden erste Ansätze zu einer Datenabfragesprache namens SEQUEL veröffentlicht, aus der sich dann später SQL entwickelt hat. In den USA ist die Aussprache von SQL als 'SEQUEL' weiterhin gebräuchlich. Die hier angegebene Syntax bezieht sich auf die durch die ISO[3] normierte Syntax, die mit SQL-92 bezeichnet wird. Dieser Abschnitt wird keine formale Herleitung und Definition von SQL enthalten, vielmehr soll gezeigt werden, wo die Elemente der Relationalen Algebra in SQL wiederzufinden sind. Weiterhin enthält jedes Datenbanksystem weitere Spracherweiterungen, die man in der Relationalen Algebra

[3] International Organization for Standardization, kurz: ISO, URL: `http://www.iso.org`

nicht finden wird. So gibt es z.B. Befehle, um Daten in eine Datenbank zu exportieren oder zu importieren, Benutzerkonten zu verwalten oder Nutzungsstatistiken abzurufen. Diese Spracherweiterungen werden nicht betrachtet. Die Syntax von SQL ist stark an die englische Sprache angelehnt und somit sehr leicht nachzuvollziehen. Es sei noch einmal hervorgehoben, dass sowohl Abfragen als auch Manipulationen der Daten mittels SQL durchgeführt werden. Hier soll jedoch nur vorgestellt werden, wie Abfragen formuliert werden.

Abfragen werden grundsätzlich mit dem Schlüsselwort 'SELECT' eingeleitet. Dies führt unmittelbar zu dem Gedanken, dass lediglich die Selektion mittels dieses Schlüsselwortes durchgeführt werden kann. Jedoch übernimmt 'SELECT' die Rolle von allen Operationen der Relationalen Algebra.

Abbildung 2.8 zeigt die vollständige Syntax von 'SELECT'. Direkt hinter dem Schlüsselwort 'SELECT' sind die Attribute anzugeben, die an den Benutzer ausgegeben werden. An dieser Stelle findet also die Projektion statt. Ist keine Projektion gewünscht, so gibt man '*' an. Hinter 'FROM' wählt man die Relation oder mehrere Relationen aus, auf denen die Abfrage durchgeführt wird. Der Natürliche Verbund wird z.B. bei Angabe von zwei Relationen zurückgeliefert. Die sogenannte WHERE-Klausel übernimmt die Selektion der Tupel, indem man eine logische Bedingung hinter dem Schlüsselwort 'WHERE' angibt. Aggregation und Sortierung der Tupel können mittels 'GROUP BY' bzw. 'ORDER BY' angefordert werden. Die Aggregation wird in Abschnitt 6.3 auf Seite 50 genauer erläutert. Die im letzten Abschnitt erläuterten Mengenoperationen werden durch Verknüpfung von mehreren SELECT-Anweisungen formuliert. So dient das Schlüsselwort 'UNION' dazu, die Vereinigung zweier SELECT-Anweisungen zu berechnen.

Dem Schlüsselwort 'DISTINCT' kommt innerhalb von SQL eine besondere Bedeutung zu. In der Relationalen Algebra können niemals zwei Tupel mit dem gleichen Wert zurückgeliefert werden. SQL sieht dies jedoch vor, so dass mit dem Schlüsselwort 'DISTINCT' das gleiche Ergebnis wie in der Relationalen Algebra erzielt wird.

Beispiel 2.8. *Die Anfrage aus Beispiel 2.7 ist als SQL-Anweisung wie folgt zu formulieren:*

```
SELECT DISTINCT franchise
       FROM spieler, mannschaft
       WHERE spieler.alter = 34
```

'DISTINCT' stellt sicher, dass jeder Franchisebezeichner nur einmal zurückgegeben wird. Danach wird mittels 'FROM' der Natürliche Verbund zwischen 'spieler' und 'mannschaft' berechnet. Weiterhin werden mit Hilfe von 'WHERE' genau die Tupel aus der Relation 'spieler' selektiert, deren Attributwert im Attribut 'alter' gleich 34 *ist.*

2.1.6. Funktionale Abhängigkeit

Eine funktionale Abhängigkeit besteht zwischen Attributmengen einer Relation R genau dann, wenn die Werte der einen Attributmenge die Werte der anderen Attributmenge bestimmen.

```
SELECT [DISTINCT] Auswahlliste
FROM Quelle
WHERE Where-Klausel
[GROUP BY (Group-by-Attribut)+
[HAVING Having-Klausel]]
[ORDER BY (Sortierungsattribut)+ [ASC|DESC]]
```

Abbildung 2.8.: Vollständige Syntax der Anweisung 'SELECT' nach SQL-92.

yearid	name	team	rank	g	lgID
2001	Boston Red Sox	BOS	2	161	AL
2002	Boston Red Sox	BOS	2	162	AL
2003	Boston Red Sox	BOS	2	162	AL
2004	Boston Red Sox	BOS	2	162	AL
2005	Boston Red Sox	BOS	2	162	AL
2001	Houston Astros	HOU	1	162	NL
2002	Houston Astros	HOU	2	162	NL
2003	Houston Astros	HOU	2	162	NL
2004	Houston Astros	HOU	2	162	NL
2005	Houston Astros	HOU	2	163	NL

Tabelle 2.5.: Beispiel für funktionale Abhängigkeit. 'team' bestimmt funktional den Wert für 'name'. Somit gilt: $\{team\} \rightarrow \{name\}$

.

Definition 2.9 (Funktionale Abhängigkeit)**.** *Seien X, Y Attributmengen der Relation R. $X \rightarrow Y$ gilt genau dann, wenn jeder Attributwert der Attribute aus X mit genau einem Attributwert aus der Menge Y übereinstimmt.*

Beispiel 2.9. *Tabelle 2.5 enthält eine funktionale Abhängigkeit. Da jeder Wert von 'team' genau einen Wert von 'name' bestimmt, besteht eine funktionale Abhängigkeit, die formal mit $\{team\} \rightarrow \{name\}$ anzugeben ist. Weiterhin gilt z.B.:*

- $\{team\} \rightarrow \{lgID\}$ *und*
- $\{lgID\} \rightarrow \{name, team\}$

2.1.7. Mehrwertige Abhängigkeiten

Fagin führt in [25] die mehrwertigen Abhängigkeiten ein. Seine Motivation entstand dadurch, dass in einer Relation jedes Attribut genau einen Attributwert besitzt, es jedoch in einigen Fällen wünschenswert wäre, mehrere Werte zuweisen zu können. Tabelle 2.6 auf der nächsten Seite zeigt eine Relation mit mehrwertigen Abhängigkeiten. Der Programmierer Smith hat zwei Qualifikationen und kann auch in mehr als einer Sprache programmieren. In Abschnitt 2.1.8 wird gezeigt, wie man die Relation *programmierer* aufbereiten kann, um sie übersichtlicher und weniger missverständlich anzugeben. Dazu muss man jedoch die mehrwertigen Abhängigkeiten (kurz: MVD für *engl. multivalued*

programmierer :

name	**qualifikation**	**sprache**
SMITH	Bachelor	Fortran
SMITH	Bachelor	Cobol
SMITH	Bachelor	Pascal
SMITH	Diplom	Fortran
SMITH	Diplom	Cobol
SMITH	Diplom	Pascal
MILLER	Master	Pascal
MILLER	Master	Cobol

Tabelle 2.6.: Beispiel für eine Tabelle mit mehrwertigen Abhängigkeiten. Kennt man den Namen des Programmierers, so kennt man auch dessen Qualifikation und dessen Sprachkenntnisse. D.h. es gilt z.B. name$\twoheadrightarrow$ qualifikation und name$\twoheadrightarrow$ sprache.

dependencies) erkennen können. Die folgende Definition definiert die MVDs formal mit Hilfe des relationalen Modells:

Definition 2.10. *Seien X, Y, Z Attributmengen der Relation R. Eine mehrwertige Abhängigkeit $X \twoheadrightarrow Y$ liegt für eine Relation $R(X, Y, Z)$ mit einer gegebenen Menge von Werten für die Attributmenge X genau dann vor, wenn eine Menge (d.h. 0 oder mehr) Werte für die Attributmenge Y eindeutig zugeordnet werden kann und diese Werte nur von X, jedoch nicht von der Attributmenge Z abhängen.*

2.1.8. Normalformen und Normalisierung

Normalformen wurden eingeführt, um Redundanzen innerhalb von Relationen zu verhindern und die Modifikation der abgelegten Informationen einfacher und effizienter durchführen zu können. Von dem letzteren Aspekt soll hier abgesehen werden. Der Aspekt der Redundanz in Relationen ist jedoch zu betrachten, da die Güte einer Repräsentation unmittelbar damit zusammenhängt. Der Prozess der Normalisierung überführt also eine Relation in eine oder mehrere normalisierte Relationen. Es gibt mehrere Normalformen, die aufeinander aufbauend definiert sind, d.h. es kann die 2. Normalform nur für Relationen hergestellt werden, die sich bereits in 1. Normalform befinden. Durch den Einsatz von E/R-Diagrammen liegen die abgeleiteten Relationen jedoch bereits automatisch in 3. Normalform vor. Mit Hilfe der Normalisierung kann dann der konzeptuelle Entwurf der abgeleiteten Relationenschemas validiert werden.

Liegt ein Relationenschema nicht einmal in 1. Normalform vor, so spricht man von einer Relation in unnormalisierter Form.

Dieser Abschnitt gibt einen formalen Überblick über die wichtigsten Normalformen von relationalen Datenbanken. Codd stellte bereits bei der ersten Veröffentlichung zum relationalen Modell in [9] einige Ansätze vor, die er dann aber in [10] in Form der ersten drei Normalformen konkretisierte. Später entwickelte er zusammen mit Boyce eine weitere Normalform, die als Boyce/Codd Normalform bekannt ist. Im Zusammenhang mit den mehrwertigen Abhängigkeiten stellte auch Fagin [25] eine weitere, die 4. Normalform,

vor.

[35] diente neben den bereits genannten Referenzen als zusätzliche Quelle, um die Informationen für diesen Abschnitt zusammenzustellen.

Es bleibt dem Entwickler eines Datenbankmodells überlassen, wie weit die Normalisierung vorgenommen wird. Eine Normalisierung bis zur letzten Normalform ist jedoch selten zu finden.

1. Normalform

Definition 2.11. *Eine Relation R ist in 1. Normalform genau dann, wenn alle ihre Attribute nur atomare Domänen besitzen.*

Die 1. Normalform sagt also aus, dass kein Wertebereich eines Attributes weiter aufgespaltet werden kann. Somit sind zusammengesetzte, mengenwertige oder geschachtelte Domänen unzulässig.

Beispiel 2.10. *Ein Attribut mit dem Namen 'adresse' enthält Strasse, Hausnummer, Postleitzahl und Ortsangabe. Die zugehörige Relation ist dann nicht in 1. Normalform. Spaltet man das Attribut jedoch in mehrere Attribute auf, so erhält man eine Relation in 1. Normalform.*

In den folgenden Normalformen ist die funktionale Abhängigkeit (vgl. Abschnitt 2.1.6) das entscheidende Kriterium, ob die Relation verändert werden muss oder nicht.

2. Normalform

Definition 2.12. *Eine Relation R ist in 2. Normalform, wenn sie in 1. Normalform ist und jedes Nichtschlüsselattribut von jedem Schlüsselkandidaten voll funktional abhängig ist.*

3. Normalform

Definition 2.13. *Eine Relation R ist in 3. Normalform, wenn sie in 2. Normalform ist und jedes Nichtschlüsselattribut von R nicht transitiv abhängig ist von jedem Schlüsselkandidaten von R.*

Boyce/Codd Normalform

Definition 2.14. *Sei X eine Menge von Attributen, A genau ein Attribut aus R. Eine Relation R ist in Boyce/Codd Normalform (kurz:BCNF), wenn $X \rightarrow A$ in R gilt und A nicht enthalten in X, dann ist X ein Schlüsselkanditat von R. Jede Relation in Boyce/Codd Normalform ist auch in 3. Normalform.*

Die letzte hier vorgestellte Normalform stellt die korrekte Modellierung von mehrwertigen Abhängigkeiten sicher (vgl. Abschnitt 2.1.7).

4. Normalform

Definition 2.15. *Eine Relation R ist in 4. Normalform, wenn sie in Boyce/Codd Normalform ist und für jede mehrwertige Abhängigkeit einer Attributmenge Y von einer Attributmenge X ($X \twoheadrightarrow Y$) gilt:*

- *die mehrwertige Abhängigkeit ist trivial oder*
- *X ist ein Schlüsselkandidat der Relation R.*

Weniger formal kann man die 4. Normalform so zusammenfassen, dass es in einer Relation nicht mehrere, voneinander unabhängige $1 : n$-Beziehungen geben darf.

3. Ontologien

Dieses Kapitel beschäftigt sich mit dem Begriff der Ontologie in der Wissensrepräsentation. Der Begriff 'Ontologie' stammt aus der Philosophie und beschäftigt sich mit Ordnung und Struktur der Realität [32, 29]. Abschnitt 3.1 grenzt den Begriff für diese Arbeit nach einer kurzen Einleitung in die Thematik genau ab. Die Abschnitte 3.2 und 3.3 beschäftigen sich mit Aspekten zur Modellierung und Strukturierung von Ontologien. Die Thematik der Deskriptiven Logiken wird in Abschnitt 3.4 eingeführt. 3.5 stellt die Entitiy Relationship-Diagramme vor. Ein kurzer Überblick über in der Praxis gebräuchliche Beschreibungssprachen für Ontologien wird in Abschnitt 3.6 gegeben. Abschließend wird das semantische Web als ein Beispiel angeführt, in dem Ontologien eine bedeutende Rolle zukommt.

Weiterhin wird eine Ontologie entwickelt, um eine gemeinsame Sprache für den Austausch und die Wiederverwendung von Wissen eines Phänomens in einem bestimmten Bereich der Realität darzustellen. Eine Ontologie kann dann benutzt werden, Fragen über das Dargestellte zu erstellen, diese zu beantworten, Annahmen zu machen, neue Einsichten zu gewinnen, einzelne Vorgehen zu beschreiben und schließlich auch um das Wissensmanagement innerhalb einer Ontologie zu diskutieren.

Bei der Benutzung von Ontologien ist hervorzuheben, dass zum einen die Ontologie selbst existiert, welche die Konzepte, die innerhalb einer betrachteten Domäne (bzw. Teil der Realität) benutzt werden, spezifiziert. Die Existenz dieser Konzepte und die Beziehungen dieser untereinander sind durch diese Definition innerhalb der Ontologie wahr bzw. ergeben sich durch Vereinbarungen, die während der Erstellung der Ontologie getroffen wurden. Andererseits gibt es empirische Tatsachen über diese Konzepte und Beziehungen, die zwar kein Bestandteil der Ontologie sind, jedoch ihre Struktur beeinflussen. Sie unterliegen einem Kontext, einer Beobachtung, Tests, einer Evaluierung und gegebenenfalls Veränderungen der Ontologie. Diesen Aspekt stellt Beispiel 3.1 klar.

Beispiel 3.1. *In der Domäne der finanziellen Dienstleistungen durch Banken sind in einer Ontologie z.B. die folgenden Teile zu finden:*

- *Währung,*
- *Stammaktien,*
- *Ausführung einer Überweisung,*
- *Bearbeitung von Abfindungen,*
- *...*

Die Tatsache, dass zwischen der Ausführung einer Überweisung und der Bearbeitung einer Abfindung mehrere Tage vergehen, muss in der Ontologie nicht modelliert werden. Es ist durch Testen und Evaluierung zu ermitteln, ob die Ontologie in diesem Punkt modifiziert werden sollte.

Ansatz	Grundsätze für dieses Vorgehen
Inspiration	eine Person entwickelt mit eigener Auffassung eine Ontologie
Induktion	ein spezieller Fall oder Ablauf wird zur Modellierung als Musterbeispiel herangezogen und als Vorlage für die Ontologie genutzt
Herleiten	Allgemeine Prinzipien werden ermittelt und daraus wird eine Modellierung abgeleitet
Synthese	eine Menge bereits existierender Ontologien, die Teilaspekte darstellen, wird benutzt, um eine einzige, genauere Ontologie zu modellieren
Zusammenarbeit	Ansichten und Perspektiven mehrerer Personen ('Expertenwissen') über die Domäne werden erfragt, regelmäßige Absprache während der Entwicklung (Evaluierung und Validierung) findet statt

Tabelle 3.1.: Ansätze zur Entwicklung einer Ontologie

Damit Ontologien in der Praxis eingesetzt werden können, ist es wichtig, dass alle involvierten Parteien diese Ontologie einsetzen. So ist es wenig sinnvoll, wenn eine einzige Bank eine Ontologie einführt, jedoch nicht mit anderen Banken auf diesem Wege Wissen austauschen kann, weil diese Ontologien ablehnen. Es ist daher unabdingbar, dass mit potentiellen Benutzern einer Ontologie während des Erstellungsprozesses kommuniziert wird, um sicherzustellen, dass die Ontologie vollständig, korrekt, verständlich und präzise ist.

Holsapple und Joshi stellen in [32] fünf Ansätze vor, wie die Entwicklung einer Ontologie durchgeführt werden kann. Tabelle 3.1 zeigt diese Ansätze, wobei die Autoren in der Veröffentlichung den Ansatz über 'Zusammenarbeit' im Detail mit Beispielen aus der Praxis vorstellen. Weitere Ansätze zur Erstellung und Handhabung von Ontologien werden in der Praxis stark diskutiert. Einen guten Überblick über Prinzipien, Methoden und Anwendungen bietet [61]. Spezielle Richtlinien zur Erstellung von wiederverwendbaren Ontologien werden in [60, 16] angeführt. [31] beziehet sich auf die Wissensrepräsentation und eine klare, terminologische Spezifikation von Ontologien.

Grüniger und Fox liefern in [30] einige Ansätze zur Entwicklung von Ontologien im Hinblick auf das logische Schließen. Mit der Thematik des Einsatzes von Ontologien in der Wissensentdeckung beschäftigt sich Stumme in [58].

Spyns, Meersman und Jarrar stellen den Begriff der Datenmodellierung dem der Ontologieentwicklung[1] gegenüber [57].

Ontologien im Bezug auf das KDD Werkzeug MiningMart (siehe Kapitel 6) erläutern Euler und Scholz in [24].

3.1. Begriff der Ontologie

Die folgenden zwei Abschnitte werden den Begriff der Ontologie für die Philosophie und die Informatik definieren.

[1] genauer im Englischen mit 'ontology engineering' zu bezeichnen

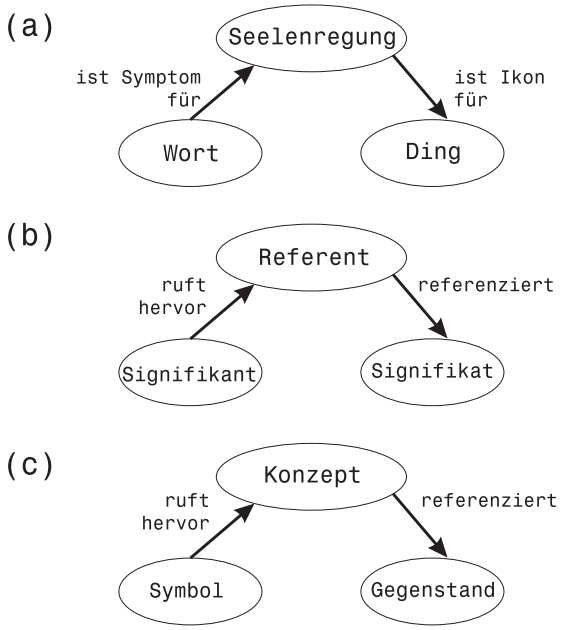


Abbildung 3.1.: Entwicklung des Semiotischen Dreiecks

3.1.1. Philosophie

Der Begriff 'Ontologie' stammt aus dem Griechischen und kann mit 'Lehre vom Sein' übersetzt werden.

Definition 3.1. *Eine Ontologie ist eine philosophische Disziplin, die sich primär mit dem Sein, dem Seienden als solchem und mit den fundamentalen Typen von Entitäten (Gegenstände, Eigenschaften, Prozesse) beschäftigt.*

Die eigentliche Frage ist hier immer 'Wie sind die Dinge als solche?'. Bereits Sokrates, Platon und Aristoteles beschäftigten sich mit dieser Frage. Aristoteles stellte die Theorie auf, dass ein Wort als Symptom für eine Seelenregung zu sehen sei, also für etwas, das der Sprecher sich vorstellt. Diese Vorstellung ist nach Aristoteles dann ein Ikon für ein Ding. Dargestellt als Dreieck, wie in Abbildung 3.1(a), bilden diese Verbindungen die primären Zeichenrelationen. Abgeleitet daraus ergibt sich die Verbindung zwischen Wort und Ding. Spätere Entwicklungen schwächen die Beziehung zwischen dem Wort und dem Bezeichneten ab. Somit besteht in 3.1(b) eine Beziehung zwischen dem Bezeichnenden (Signifikant) und dem Bezeichneten (Signifikat). 3.1(c) führt die Begriffe so auf, wie sie im Rahmen dieser Diplomarbeit verwendet werden. So ist ein Bezug zur Literatur aus dem Bereich der Philosophie gegeben. Zur Vertiefung dieses Gebietes wird [62] als Primärliteratur herangezogen.

3.1.2. Informatik

Allgemein versteht man in der Informatik im Bereich der Wissenrepräsentation unter einer Ontologie ein *formal definiertes System von Konzepten und Beziehungen.*

In [28] nennt Gruber eine erweiternde Definition.

Definition 3.2 (Ontologie [28]). *Eine Ontologie ist eine* SPEZIFIKATION *einer* KONZEPTUALISIERUNG.[2]

Hiermit meint Gruber, dass eine Ontologie eine Beschreibung der Konzepte und Beziehungen, die für einen Agenten oder eine Menge von Agenten existieren können, ist. Dabei kann es sich zum Bespiel um die formale Spezifikation eines Programmes handeln.

Viel wichtiger ist jedoch nach Gruber, welchem Zweck eine Ontologie dient. Wie in der Einleitung dieses Kapitels bereits frei formuliert, bilden der Austausch und die Wiederverwendbarkeit von Wissen den Hauptgrund für die Bildung von Ontologien. Eine Ontologie wird hierbei durch eine Menge von Definitionen in einem formalen Vokabular, den Beschreibungssprachen, angegeben. Abschnitt 3.6 auf Seite 28 gibt einen kurzen Überblick über einige dieser Beschreibungssprachen. Weiterhin ist es erforderlich, dass bei der Benutzung von Ontologien alle Parteien die gleichen Ontologien für identische Abläufe zu Grunde legen und die gleiche formale Sprache einsetzen.

3.2. Modellierung

Bei der Modellierung von Ontologien werden verschiedene Stufen der formalen Beschreibung unterschieden. Auch bei der Wissensrepräsentation wird genau in diesen Kategorien unterschieden, wie die Beschreibung bzw. Präsentation der Daten einzuordnen ist [53]:

informal natürlichsprachliche Beschreibung

semi-informal strukturierte Beschreibung in beschränkter natürlicher Sprache

semi-formal Beschreibung in künstlicher, formal definierter Sprache

formal Beschreibung in sorgfältig definierten Begriffen mit formaler Semantik

3.3. Strukturierung

Es gibt eine Vielzahl von Ansätzen, Ontologien zu kategorisieren bzw. Charakteristika zu finden, um zwischen Ontologietypen zu unterscheiden. Die folgende Auflistung enthält die wichtigsten Ontologietypen, wobei hervorzuheben ist, dass in der Literatur viele andere Einteilungen zu finden sind. Abbildung 3.2 auf der nächsten Seite zeigt verschiedene Ontologietypen, geordnet nach der Ausdrucksstärke. Eine Taxonomie hat die schwächste, die Logische Theorie, in die zum Beispiel die Prädikatenlogik einzuordnen wäre, die größte. Einige Modelle, die in dieser Arbeit vorgestellt werden, sind an der entsprechenden Stelle auf der Achse angeordet. Es folgt eine Auflistung der wichtigsten Unterscheidungs-

[2] engl.: An ontology is a specification of a conceptualization

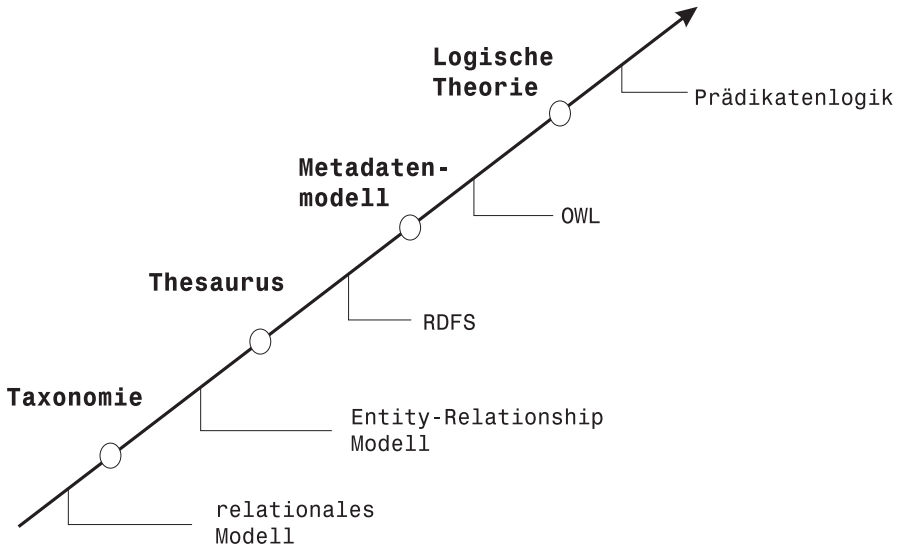


Abbildung 3.2.: Kategorien für Ontologien

kriterien.

Thesaurus Ein Thesaurus besitzt ein kontrolliertes Vokabular, dessen Begriffe durch Beziehungen miteinander verbunden sind. Bei den Beziehungen der Begriffe untereinander handelt es sich z.B. um Synonyme, Ober- und Unterbegriffe, Homonyme und ähnliche Begriffe.

Taxonomiesysteme Taxonomiesyteme dienen der Klassifikation von Dingen in einer geordneten Struktur. Meist handelt es sich um eine hierarchische Struktur, aber auch Netzwerke werden häufig benutzt. Bekannt sind vor allem Taxonomien zur Klassifikation von Lebewesen, die eine hierarchische Form aufweisen, um zum Beispiel Hunde verschiedenen Rassen zuzuordnen.

Top-Level Ontologie Eine Top-Level Ontologie dient zur Beschreibung genereller Konzepte, wie z.B. Zeit, Raum, Vorgang. Dies geschieht dann unabhängig von einer bestimmten Domäne oder Problemstellung.

Domänen Ontologie Die Domänen Ontologie (engl. domain ontology) beschreibt ein grundlegendes Vokabular, welches genau auf eine generische Domäne bezogen wird. Die Konzepte selbst werden dann in einer Top-Level Ontologie spezifiziert.

Im folgenden Abschnitt werden Deskriptive Logiken als Beispiel für einen logikbasierten Formalismus der Wissensrepräsentation vorgestellt.

3.4. Deskriptive Logiken

Dieser Abschnitt orientiert sich an [49]. Nardi und Brachman liefern einen sehr guten Einstieg in die Thematik der Deskriptiven Logiken.

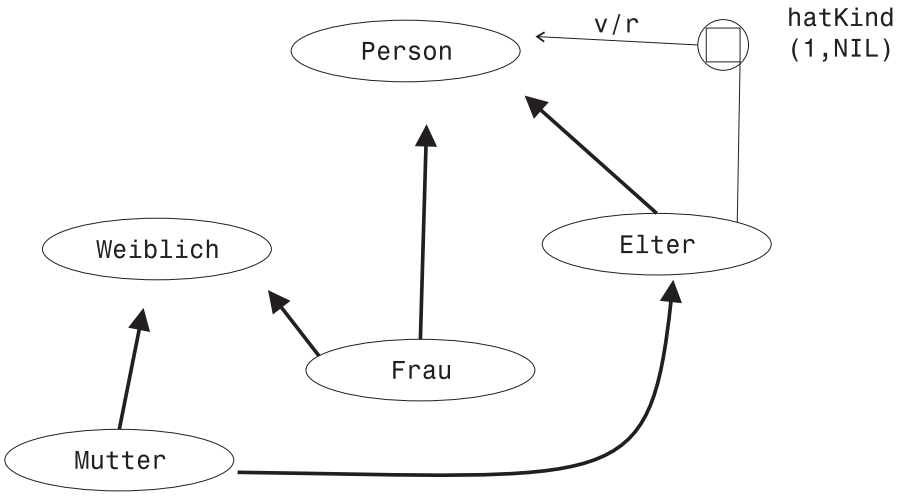


Abbildung 3.3.: Beispiel für ein Netzwerk-Modell. Es dient als Grundlage für die Beispiele zur Einführung der Deskriptiven Logik.

Deskriptive Logiken bilden den Gegenpol zu nicht auf Logik basierenden Ansätzen der Wissensrepräsentation. Beispiele dafür sind Netzwerke. Ein Netzwerk-Modell ist beispielhaft in Abbildung 3.3 zu sehen. Anhand dieses Modelles werden in diesem Abschnitt allgemeine Ansätze einer Deskriptive Logik eingeführt. Die Art der Repräsentation mit Hilfe von Netzwerken oder ähnlichen grafischen Strukturen basiert meist auf Experimenten bzw. Erfahrungswerten. Eine Verallgemeinerung auf viele, gleichartige Problemstellungen fällt bei diesem Vorgehen meist schwer. Daher liegt es nahe, eine Logik erster Ordnung zu benutzen, da diese bereits sehr mächtig ist. Weiterhin sind auf Logik basierende Ansätze direkt allgemeingültig und erlauben die Auswertung von Ausdrücken auf eine einfach zugängliche Weise. Andere Ansätze benötigen meist weitere Strukturen, um das Prinzip des Schließens zu ermöglichen. Netzwerksysteme sind zwar für Menschen einfacher zugänglich, jedoch erreicht die Ausdrucksfähigkeit schnell ihre Grenzen. So entwickelten sich zum Beispiel aus den Netzwerk-Modellen die Entity-Relationship-Diagramme (vergleiche Abschnitt 3.5), die wesentlich mehr Möglichkeiten der Modellierung ermöglichen. Die Deskriptiven Logiken bleiben jedoch weiterhin mächtiger.

Zur Begrifflichkeit bei der Arbeit mit Deskriptiven Logiken ist zu sagen, dass die Ausdrücke bzw. eine Menge von Individuen einer Sprache in einer Deskriptiven Logik mit dem Wort 'Konzept' bezeichnet werden. Dies ist eine unglückliche Wortwahl, da Konzepte in dieser Arbeit auch im Rahmen der konzeptuellen Ebene der Datenbankmodellierung auftreten. In diesem Abschnitt ist jedoch immer der Begriff aus der Thematik der Desktiptiven Logik gemeint.

Es sollen nun einige Aspekte von Deskriptiven Logiken anhand von Beispielen, basierend auf dem Netzwerk-Modell in Abbildung 3.3, aufgezeigt werden.

Die Elemente in einem Netzwerk sind Knoten und Verbindungen zwischen Knoten. Knoten bezeichnen meist Konzepte. Verbindungen zwischen den Konzepten zeichnen Beziehungen aus. In einigen Modellierungen gibt es auch Knoten, die zu einer Verbindng gehören, um eine bestimmte Relation zwischen Konzepten auszudrücken. In Abbildung

3.3 bezeichnet 'hatKind' einen Knoten, der eindeutig zu einer Relation zwischen 'Person' und 'Elter' gehört. Dazu enthält die Abbildung eine sogenannte IS-A-Beziehung, welche eine Hierarchie über Konzepten definiert. So ist zum Beispiel 'Mutter' als 'Elter' in der Abbildung zu erkennen. Als besondere Eigenschaft dieser Beziehung ist zu erwähnen, dass ein Konzept vom generelleren Konzept die Eigenschaften 'erbt'. Das heißt, besitzt eine 'Person' die Eigenschaft 'Alter', so wird auch eine 'Mutter' diese Eigenschaft besitzen.

Weiterhin können Relationen Rollen zugewiesen werden. In Abbildung 3.3 ist 'hatKind' eine Rolle. Dazu hat diese Rolle eine Beschränkung, die aussagt, dass jeder Elter eine Person ist, die mindestens ein Kind hat, wobei es sich bei jedem Kind um eine Person handelt. In diesem Modell hat die Rolle also ein Intervall für die Einschränkung vorgegeben. Ein wichtiger Aspekt ist, dass auch Rollen auf erbende Konzepte übernommen werden.

Aus diesen Erklärungen ist leicht ersichtlich, dass viele weitere Aussagen über das Modell gemacht werden können. Daraus entwickelt sich der Ansatz für die Deskriptiven Logiken. Eine präzise Charakterisierung der Aussage eines Netzwerkes kann dadurch angegeben werden, wenn eine Sprache für die Elemente der Struktur angegeben wird. Das heisst, man erstellt eine Vorschrift für die Belegung der Zeichenketten dieser Sprache. Bei der Syntax gibt es verschiedene Ausprägungen, wobei sich bei der Semantik Tarski-ähnliche Formen durchgesetzt haben.

Die hier vorgestellte Syntax ist eine Art abstrakter Sprache. Gegeben seien hierfür zwei disjunkte Alphabete, die zum einen atomare Konzepte (unäre Prdikatensymbole) und zum anderen atomare Rollen (binäre Prädikatensymbole) ausweisen. Ein Term wird zum Beispiel durch eine Operation zwischen zwei Konzepten gebildet. So bezeichnet

$$C \sqcap D \tag{3.1}$$

den Schnitt zwischen zwei Konzepten. Es werden also die Individuen auf die eingeschränkt, die zu C und zu D gehören. Zum Einstieg kann man zum einfacheren Verständnis Ausdruck 3.1 auch mit

$$C \cap D \tag{3.2}$$

gleichsetzen.

Dazu ist anzumerken, dass Ausdrücke zwischen Konzepten in Deskriptiven Logiken keine Variablen enthalten.

Eine Restriktion, dass Konzepte, die zu einem Konzept R gehören, auch zu Konzept C gehören, wird durch

$$\forall R.C \tag{3.3}$$

ausgedrückt.

Grundsätzlich kann man sagen, dass die Semantik für Konzepte mengentheoretischer Natur ist: Zu einem Konzept gehört eine Menge von Individuen und zu Rollen gehören Paare von Individuen. Die Domäne einer Deskriptiven Logik kann beliebig gewählt werden, d.h. auch unendlich. Gerade die Möglichkeit, über unendliche Bereiche zu argumentieren, zeichnet die Deskriptiven Logiken aus.

Beispiel 3.2. *Einige weitere Beispiele zur Verknüpfung von Konzepten:*
- $Person \sqcap \neg Weiblich$: *Personen, die nicht weiblich sind*
- $Weiblich \sqcup Maennlich$: *alle Individuen, die männlich oder weiblich sind*

$\sqcap$	Konjunktion von Konzepten
$\sqcup$	Disjunktion von Konzepten
$\neg$	Negation eines Konzeptes

Tabelle 3.2.: Mögliche Verknüpfungen zwischen Konzepten und ihre Bedeutung

Zudem können, wie aus der Prädikatenlogik bekannt, die Quantoren $\forall$ und $\exists$ benutzt werden:

Beispiel 3.3. *Die folgenden Ausdrücke benutzen Quantoren:*
- $\exists hatKind.Weiblich$: *Individuen, die ein weibliches Kind haben*
- $\forall hatKind.Weiblich$: *Individuen, die nur weibliche Kinder haben*

Auf diese Art können Rollen mit Beschränkungen auch in der entsprechenden Logik ausgedrückt werden. So kann die Beschränkung, dass Elter mindestens ein Kind haben müssen, ausgedrückt werden mittels

$$\exists hatKind.Person \sqcap \forall hatKind.Person\ . \tag{3.4}$$

Beschränkungen über die Anzahl von Individuen, die einer Rolle zugeordnet werden können, können in Deskriptiven Logiken ebenfalls direkt ausgedrückt werden. Das Konzept

$$(\geqslant 3\, hatKind) \sqcap (\leqslant 2\, hatKind.Weiblich) \tag{3.5}$$

umfasst alle Individuen, die mindestens 3 Kinder haben, von denen jedoch nicht mehr als 2 weiblich sind.

Diese Art der Beschränkungen wird häufig als besondere Ausdrucksmöglichkeit der Deskriptiven Logiken angesehen, jedoch bieten zum Beispiel auch die Entity-Relationship-Diagramme diese Möglichkeit.

Zu Beginn dieses Abschnitts wurde die einfache Möglichkeit erwähnt, mit Deskriptiven Logiken zu folgern. Die Deduktion wird in der Notation üblicherweise mit $\sqsubseteq$ gekennzeichnet und erfolgt über eine Subsumtion, die in Abschnitt 5.2 erläutert und in Defintion 5.7 auf Seite 37 definiert wird. Es ist anzumerken, dass die Subsumtion sehr detailliert theoretisch untersucht wurde, besonders in Bezug auf Berechenbarkeit und Effizienz.

Tabelle 3.2 auf dieser Seite zeigt eine Übersicht über die vorgestellten Verknüpfungen in Ausdrücken.

Es werden nun noch zwei Komponenten der Deskriptiven Logiken genannt, die den Einsatz als Wissensbasis gestalten. Bisher ist noch nicht erläutert worden, wie eine Sprache in Deskriptiver Logik in einem System zur Wissensrepräsentation eingesetzt werden kann. Vielmehr muss eine Sprache zur Definition der Wissensbasis angegeben und dazu eine Möglichkeit der Deduktion in ihr ermöglicht werden.

Um eine Terminologie basierend auf Konzepten zu bauen, wird eine sogenannte TBox eingeführt. Eine TBox ist eine Definition eines Konzeptes. So kann zum Beispiel das Konzept 'Frau' wie folgt als TBox definiert werden:

$$Frau \equiv Person \sqcap Weiblich \tag{3.6}$$

In Wissensbasen, die auf Deskriptiven Logiken basieren, kann also eine Terminologie durch eine Menge von Konzeptdefinitionen der obigen Form erstellt werden. Es gibt

entscheidende Einschränkungen bei der Definition von TBox-en, die hier kurz aufgeführt werden sollen:

- jedes Konzept darf nur einmal definiert werden
- Definitionen müssen azyklisch sein, d.h. Konzepte werden nicht in Abhängikeit von sich selbst oder in Abhängigkeit von anderen zu dem zu definierenden Konzept definiert.

Diese Einschränkungen stellen sicher, dass jedes Konzept eindeutig expandiert werden kann.

Als zweite Komponente ist die sogenannte ABox einzuführen, die erweiterndes Wissen zu einer Domäne liefert. Dies besteht aus Behauptungen über Individuen.

So wird das Individuum 'PEGGY' durch Ausdruck 3.7 als weibliche Person definiert.

$$Weiblich \sqcap Person(PEGGY) \tag{3.7}$$

Weiterhin können auch eine ABox über Relationen gebildet werden:

$$hatKind(PEGGY, KELLY) \tag{3.8}$$

Ausdruck 3.8 sagt aus, dass 'KELLY' ein Kind von 'PEGGY' ist. Ausdruck 3.7 ist eine 'konzeptuelle Behauptung'[3] und Ausdruck 3.8 eine 'Behauptung über eine Rolle'[4].

Sowohl die TBox als auch die ABox haben spezielle Schlussmechanismen, die in [49] ausführlich beschrieben und mit Beispielen belegt werden.

Nardi legt in der Einleitung von [49] auch Wert darauf, die Herkunft der Deskriptiven Logiken anzugeben. Vorgänger von Deskriptiven Logiken ist KL-ONE. KL-ONE leitete den Übergang von semantischen Netzwerken zu wohl fundierten Logiken ein.

Der Einfluss von KL-ONE ist tiefgründig und daher wird KL-ONE als Wurzel aller Sprachen der Deskriptiven Logiken angesehen. So führte KL-ONE viele der Ideen ein, die bei der Arbeit mit Deskriptiven Logiken untersucht wurden. Bereits KL-ONE enthielt Konzepte, Rollen und die mögliche Beziehung zwischen diesen Komponenten.

Weiterhin sind auch die Restriktionen und die Deduktionsmöglichkeiten dort zu finden. Grundaspekte für TBox und ABox wurden ebenfalls im Rahmen von Arbeiten mit KL-ONE erarbeitet. Außerdem kann man sagen, dass KL-ONE als das erste Beispiel für eine Sprache gilt, die ein erhebliches Zusammenspiel von Theorie und Praxis zulässt. Dies ist auch ein grundlegendes Charakteristikum von Deskriptiven Logiken.

3.5. Entity-Relationship-Diagramme

Zur Modellierung von Datenbanksystemen werden in der Praxis bisher keine der hier vorgestellten Ansätze verwendet. Vielmehr sollen nun die Entity-Relationship-Diagramme eingeführt werden.

Diese Diagramme sind eine Konsequenz aus Ansätzen, die zuvor verwendet wurden und zahlreiche Nachteile aufwiesen. Die Entity-Relationship-Diagramme (kurz ERDs) basieren auf den Netzwerk-Modellen (vgl. [2, 56]), dem Relationenmodell (vgl. [9, 11, 12])

[3] engl. concept assertion
[4] engl. role assertion

und dem Entity Set Model, welches in [56] vorgestellt wird. Das Netzwerk-Modell ist zwar sehr gut dazu geeignet, eine natürliche Sicht auf die Daten durch die Trennung von Entitäten und Beziehungen zu gewinnen, jedoch ist die Unabhängigkeit von den Daten problematisch.

Das bereits in Abschnitt 2.1.3 vorgestellte relationale Modell weist diese Abhängkeit von den Daten nicht auf, stellt aber semantische Aspekte der Domäne aus der Praxis nur eingeschränkt dar. Das Entity Set Model ist ebenfalls datenunabhängig, allerdings ist die Darstellung der Werte im Modell nicht unmittelbar zugänglich. Basierend auf diesen Kernaussagen entwickelte Chen dann die ERDs (vgl. [8]).

In Abschnitt 3.4 wurde ein Netzwerk-Modell vorgestellt, um Deskriprive Logiken einzuführen. Dabei wurde bereits mehrmals auf die ERDs verwiesen.

Zwei Aspekte sollten direkt hervorgehoben werden: Zum einen können ERDs sehr einfach in das Relationenmodell umgewandelt werden, was den Ansatz nahelegt, zunächst Diagramme zu entwickeln und diese dann formal in Relationenalgebra zu formulieren. Zum anderen können ERDs dazu benutzt werden, um ein im Relationenmodell erstelltes System zu validieren.

In Abschnitt 3.4 wurden weitere Aspekte hervorgehoben, die in Deskriptiven Logiken ausgedrückt werden können, jedoch nicht mit Netzwerk-Modellen. ERDs können - genau wie die Deskriptiven Logiken - eine 'IS-A'-Beziehung ausdrücken. Dies ist bei der Modellierung von Datenbanksystemen eine unerlässliche Methode, um Generalisierung und Spezialisierung abzubilden. Weiterhin können auch in ERDs die Anzahl begrenzende Restriktionen mittels Multiplizitäten auf den Relationen angegeben werden. Diese Eigenschaft wird in der Literatur zwar als eine besondere Eigenschaft der Deskriptiven Logiken ausgezeichnet, ist aber auch in ERDs und anderen Modellierungsansätzen zu finden.

In ERDs können auch Rollen modelliert werden. Diese Rollen können zudem mit gewissen Restriktionen versehen werden.

Es ist also klar zu erkennen, dass die ERDs einen Gegensatz zu den Deskriptiven Lokigen bilden, da sie keinen logischen Ansatz besitzen.

In der Praxis erstellt man häufig direkt ein Modell aus den Tabellen und Relationen einer Datenbank. Viele Datenbanksysteme bieten Unterstützung für eine grafische Darstellung der Relationen und deren Beziehungen zueinander, so dass eine grafische Überprüfung des Datenmodels leicht möglich ist. Leider führen die meisten Datenbanksysteme eigene Darstellungsformen ein und nutzen nicht die von Chen gewählten grafischen Ansätze. Zum Beispiel werden die Entitäten, angelehnt an UML, meist mit rechteckigen, und nicht mit ovalen Formen dargestellt. Aus diesem Grund werden die ERDs in dieser Arbeit auch nur den Deskriptiven Logiken gegenübergestellt, von einer Einführung der Modellierungsaspekte wird abgesehen.

Die Modelle in dieser Arbeit sind in ihrer Darstellung an MiningMart angelehnt und werden nicht als ERDs dargestellt.

3.6. Beschreibungssprachen

In den letzten Abschnitten wurde für den erfolgreichen Einsatz von Ontologien vorausgesetzt, dass eine gemeinsame Sprache aller Beteiligten erforderlich ist. Es gibt in der Literatur zahlreiche Ansätze, wobei sich der in der folgenden Liste als erster vorgestellte Ansatz zur Zeit mehr und mehr manifestiert. Es ist jedoch nicht anzunehmen, dass

die hier vorgestellten Sprachen irgendwann die zur Zeit in der Praxis verwendeten Sprachen vollständig ersetzen werden. Bei den Literaturverweisen handelt es sich meist um Spezifikationen oder Vorschläge des World Wide Web Consortium[5](W3C).

Extensible Markup Language (XML) bezeichnet einen Standard des World Wide Web Consortium für von Maschinen und von Menschen lesbaren Dokumenten in Form einer Baumstruktur. XML definiert die Regeln des Aufbaus der Dokumente, kann jedoch auch die Daten, die zunächst beschrieben werden, enthalten (vgl. [52, 3])

Die Elemente innerhalb eines XML-Dokumentes lassen sich frei wählen. Dokumente können beliebige Daten enthalten und beschreiben. Der Grundgedanke von XML ist es, die Daten von ihrer Repräsentation zu trennen. Weiterhin können XML-Dokumente auf allen Computersystemen ausgetauscht werden. XML besitzt so die Fähigkeit, die wichtigsten Datenstrukturen der Informatik, wie Listen, Bäume und Datensätze, erfassen zu können. Weiterhin ist durch die Festlegung eines Schemas, welches sogar mit Datentypen (ähnlich einer Datenbank) versehen werden kann, eine strikte Syntax gegeben, die die maschinelle Auswertung eines XML-Dokumentes z.B. mit Hilfe eines Parsers erleichtert. Zwei häufig genannte Nachteile sind zum einen die Wortfülle, da das Dokument durch die vielen Meta-Markierungen aufgebläht wird. Weiterhin setzt XML auf einer hierarchischen Datenstruktur auf, die z.B. gegenüber dem relationalen Modell (siehe Abschnitt 2.1.3) entscheidende Nachteile aufweist. XML ist eine echte Teilmenge des SGML-Standards, da die Definition der Sprache durch Einschränkung der SGML-Sprachdefinition erfolgt.

Im Gegensatz zu den folgenden beiden Sprachen ist XML, aufgrund ihres Einsatzes durch Firmen wie Microsoft und Google in ihren Produkten, stark verbreitet.

Resource Description Framework (RDF) ist eine formale Sprache zur Bereitstellung von Metadaten im World Wide Web (vgl. [3]). Grundidee ist, das Web in einer maschinell lesbaren Form zu beschreiben.

Schemata in RDF werden meist durch Graphen dargestellt, jedoch ist auch die Formalisierung mit Hilfe von XML zu finden. Weiterhin gibt es eine Anfragesprache namens SPARQL, die in [50] definiert wird. RDF bildet mit seiner Anfragesprache SPARQL und mit XML einen Grundbaustein des semantischen Webs. Die Sprache wurde zeitgleich zur Sprache OWL entwickelt.

Web Ontology Language (OWL) ist eine Spezifikation des Worldwide Web Consortiums, die es ermöglicht, Ontologien mittels einer formalen Beschreibungssprache zu erstellen (vgl. [39]). Es werden mit OWL Teile einer Domäne, wie in der Einleitung zu Kapitel 3 gefordert, so formal beschrieben, dass auch Software-Agenten ihre Bedeutung verarbeiten, d.h. im korrekten Kontext verstehen und auswerten können.

Die Sprache basiert auf RDF-Syntax, hat allerdings eine höhere Ausdrucksstärke, da weitere Sprachelemente eingeführt werden. Neue Ausdrücke werden gebildet, um sich der Prädikatenlogik anzunähern. OWL bildet einen weiteren Grundbaustein des semantischen Webs.

[5]URL: `http://www.w3.org`

Web 1.0 (veröffentlichen)	Web 2.0 (teilnehmen)	Anwendung
persönliche Webseite	Weblog	persönlicher Webauftritt
BritannicaOnline	Wikipedia	Enzyklopädien
Ofoto	Flickr	Fotoalben
DoubleClick	GoogleAdSense	Werbung
Screen Scraping	Web Services	Inhalte verbreiten
Content-Management-System	Wikis	Verwaltung der Inhalte
Taxonomie	Folksonomy	Klassifizierung von Inhalten
einzelner Artikel	Verbreitung von Artikeln	Interopeabilität

Tabelle 3.3.: Gegenüberstellung von Web 1.0 und Web 2.0 (semantisches Web)

3.7. Semantisches Web

Das semantische Web [17] ist eine Weiterentwicklung des World Wide Web in dem Sinne, dass die Informationen mit Metadaten erweitert werden, die das Web durch Maschinen lesbar machen sollen. Der Entwurf stammt vom Begründer des World Wide Web, Tim Berners-Lee.

Bisher bestehen Inhalte im Web aus HTML-Quellcode, der von einem Browser interpretiert wird. HTML vermischt den eigentlichen Inhalt mit den Festlegungen von Layout, eine Extraktion der eigentlichen Informationen fällt schwer. Es gibt in der Literatur viele Ansätze zum Thema 'Screen Scraping', das sich mit der maschinellen Untersuchung von Webseiten beschäftigt. Auch der Versuch der Einführung von zusätzlichen Marken in HTML, die z.B. von Suchmaschinen zur Kategorisierung benutzt werden sollten, führte zu keiner Verbesserung, da die einzelnen Begriffe z.B. nicht durch die Software-Agenten einem Kontext zugeordnet werden können.

Somit stellt das semantische Web den Ansatz der Annotierung vor, d.h. jedes Dokument wird mit einer Ontologie z.B. in Form eines RDF- oder OWL-Schemas versehen. Dadurch ist nicht nur eine Kategorisierung der Inhalte möglich, sondern auch das Ziehen von Schlussfolgerungen. Anfragen können über Anfragesprachen wie SPARQL direkt an eine Seite gerichtet werden.

Der in dieser Arbeit untersuchte Datensatz wird ebenfalls im Web publiziert. Das Projekt wurde jedoch bereits vor Entstehung des Web ins Leben gerufen, so dass die Anbindung an das Web nachträglich entwickelt wurde. Diese Schnittstelle liefert jedoch keinerlei Metainformationen, so dass die Rohdaten Ausgangspunkt dieser Arbeit sein mussten, um die Informationen komplett durch Einsatz des KDD-Prozesses aufzubereiten (vgl. Abschnitt 4).

Ein weiterer Hauptaspekt des semantischen Web im Vergleich zu der ersten Web-Variante ist die Einbeziehung des Benutzers. So steht im 'alten' Web das Prinzip der Publizierung im Vordergrund, d.h. der Benutzer erstellt einen Inhalt und überträgt ihn ins Internet. Änderungen können nur durch Ersetzen dieses Dokumentes vorgenommen werden. Das semantische Web bzw. Web 2.0 hingegen setzt auf Partizipation und Interaktion der Benutzer. Dokumente werden zwar immer noch von einem Benutzer in das Internet gestellt, jedoch haben andere Benutzer z.B. Möglichkeiten der Kommentierung oder sogar der Ergänzung. Tabelle 3.3 auf dieser Seite stellt einige bekannte Technologien vor, die durch das semantische Web ins Leben gerufen wurden, und stellt sie den bisher bekannten gegenüber.

4. Wissensentdeckung in Datenbanken (KDD) anhand des Prozessmodells CRISP-DM

Dieses Kapitel definiert im nächsten Abschnitt den Begriff der Wissensentdeckung in Datenbanken und erläutert dann in Abschnitt 4.2 das Prozessmodel CRISP-DM, welches den Lebenszyklus eines Data Mining-Projektes in einzelne Phasen zerlegt. Diese Phasen werden in Abschnitt 4.2 in Unterabschnitten ausführlich erläutert.

4.1. Wissensentdeckung in Datenbanken

Um den Begriff der Wissensentdeckung in Datenbanken bzw. der Knowledge Discovery in Databases zu definieren wird meist folgendes Zitat aus [26] angegeben:

Definition 4.1 (Wissensentdeckung in Datenbanken)**.** *Wissensentdeckung in Datenbanken ist der nichttriviale Prozess der Identifikation gültiger, neuer, potentiell nützlicher und schlussendlich verständlicher Muster in (großen) Datenbeständen.*

Der folgende Abschnitt erläutert das CRIP-DM-Prozessmodell, welches die schrittweise Vorgehensweise zur Bearbeitung des Fallbeispieles in Kapitel 7 bestimmt hat.

4.2. Prozessmodell CRISP-DM

Das hier vorgestellte Prozessmodell mit dem Namen CRISP-DM 1.0 vermittelt einen ausführlichen Überblick über die einzelnen Abschnitte eines Data Mining-Projektes. Einige Kernaspekte, die ausführlich in [7] dargelegt werden, werden in den folgenden Abschnitten erläutert. Es werden vor allem die Aspekte angesprochen, die bei der Bearbeitung des Beispieles in Kapitel 7 eine Rolle gespielt haben.

Abbildung 4.1 auf der nächsten Seite zeigt die sechs Schritte, die ein Data Mining-Projekt durchläuft, einige werden unter Umständen - wie die Pfeile andeuten - mehrmals durchlaufen. Das heißt insbesondere auch, dass jeder beliebige der vorangegangenen Schritte überarbeitet wird, wenn in einem der nachfolgenden Schritte festgestellt wird, dass Anpassungen notwendig sind.

Die folgenden Unterabschnitte beschreiben jeden der sechs Schritte, die in Abbildung 4.1 auf der nächsten Seite dargestellt sind. Jeder der Schritte sollte abschließend bei der Durchführung dokumentiert werden, damit bei erneuter Iteration eines Schrittes bereits ermittelte Informationen zur Verfügung stehen und man auch zu einem späteren Zeitpunkt immer auf zuvor gefundene Ergebnisse zurückgreifen kann. Es ist anzumerken, dass [7] zu jedem der Schritte nicht nur detailliertere Informationen, sondern auch feinere Unterteilungen vornimmt, die hier nicht im Vordergrund stehen sollen. Die Schritte zum Datenverständnis und zur Datenvorbereitung sind exemplarisch in Abbildung 4.2 auf Seite 33 dargestellt.

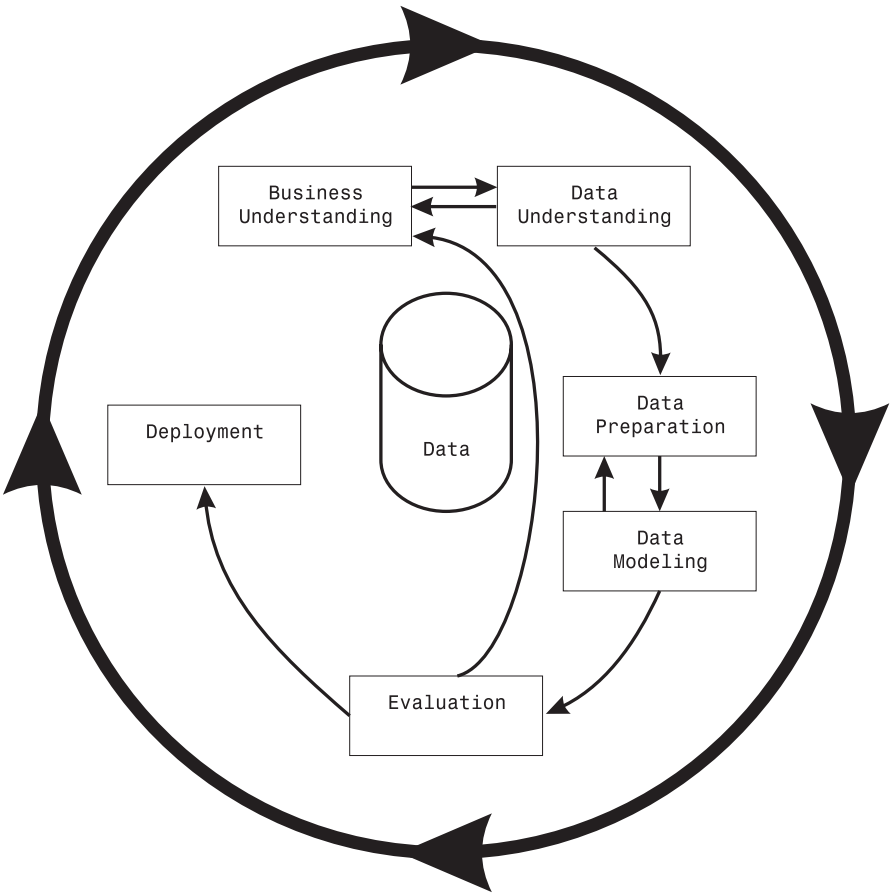


Abbildung 4.1.: Phasen des CRISP-DM, entnommen aus [7]

4.2.1. Verstehen des Sachbereiches

Bevor man sich mit den Daten eines Projektes auseinandersetzt, muss zunächst geklärt werden, um welchen Sachbereich es geht und welche Ziele verfolgt bzw. welche Probleme gelöst werden sollen. Außerdem sollten Experten befragt werden, um Schlüsselaspekte aufzudecken. Daraus können dann Problemdefinitionen für das Data Mining erarbeitet werden. Auch die Erstellung eines vorläufigen Planes, um diese Probleme zu lösen, gehört zum ersten Schritt.

4.2.2. Datenverständnis

Der zweite Schritt beginnt damit, zunächst alle verfügbaren Daten (Rohdaten) zusammenzutragen und verfügbar zu machen. Dann kann man sich mit den Daten vertraut machen, um Qualitätsaspekte zu analysieren und erste Eindrücke zu bekommen. Eine sprachliche Beschreibung der vorliegenden Daten mit Aspekten wie Format, Anzahl der Datensätze, Anzahl der Fehler usw., ist ratsam. In [7] wird angeraten, sowohl einen Bericht über die zusammengetragenen Daten als auch über eine Beschreibung der Daten anzufertigen. Im Vordergrund soll die Analyse der Qualität der vorliegenden Daten ste-

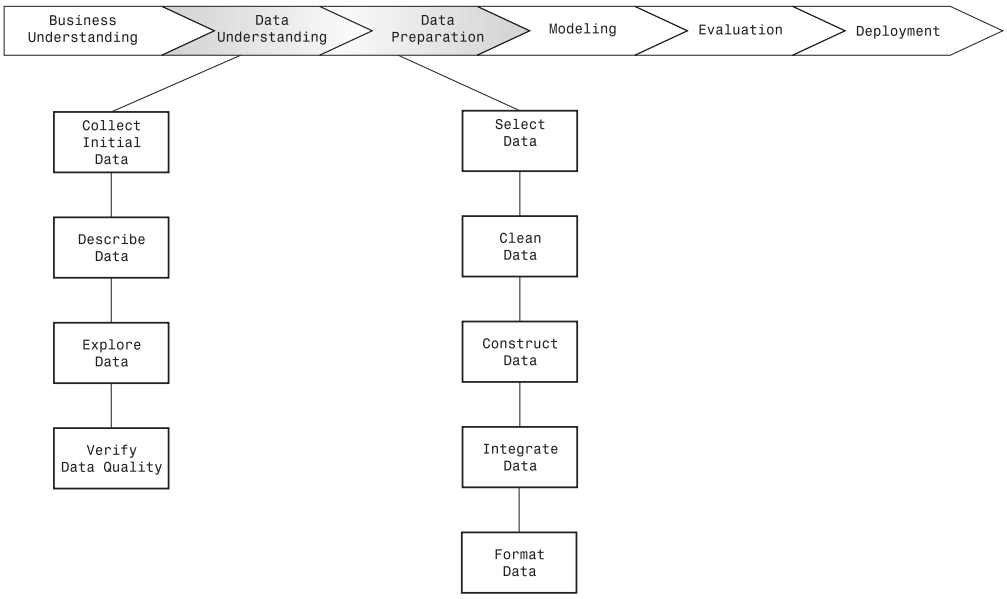


Abbildung 4.2.: Das Schema zeigt die zwei für diese Arbeit zentralen Phasen des CRISP-DM Modells im Detail.

hen, um diese an den Anforderungen des durchzuführenden Mining-Prozesses zu messen. So können interessante Teilmengen in den Daten gefunden werden, um verborgene Informationen mit Hilfe der Datenvorbereitung im nächsten Schritt zugänglich zu machen. Abschließend können Tests durchgeführt werden, um die unmittelbar festgestellte Qualität zu verifizieren. Mögliche Tests sind zum Beispiel die Betrachtung der Vollständigkeit und die Verifikation der Daten. Falls (sachliche) Fehler in den Daten auftreten, ist eine tiefergehende Analyse erforderlich, wie häufig diese sind. Hier wird klar, wie wichtig es ist, sich zunächst mit dem Sachbereich vertraut zu machen, da sonst einige der Fehler nicht erkannt werden können.

4.2.3. Datenvorbereitung

In diesem Schritt wird aus den Rohdaten der endgültige Datensatz erstellt, der dann von den Modellierungswerkzeugen im nächsten Schritt verwendet wird. Dieser Schritt wird meist mehrmals ausgeführt, da z.B. in einem späteren Schritt festgestellt wird, dass einige Informationen auf eine andere Weise aufbereitet werden müssen. Den Hauptteil der durchzuführenden Aufgaben in diesem Schritt bildet zunächst die Auswahl der aufzubereitenden Daten, die eine häufige Fehlerquelle darstellt. Dies wird meist erst in späteren Schritten erkannt, wenn zum Beispiel erforderliche Daten nicht berücksichtigt wurden, die zunächst als unwichtig eingestuft worden sind. Die gewählten Daten werden unter Umständen gesäubert. Dazu werden eventuell neue Merkmale errechnet, fehlende Werte ersetzt oder Werte zu einem Wert zusammengefasst.

Beispiel 4.1. *Das Ergänzen der Daten könnte zum Beispiel bei vorhandenen Feldern für Breite und Höhe die zusätzliche Einführung der Fläche in den Datensatz umfassen.*

Abschließend sollte die Datenvorbereitung das Formatieren der Daten umfassen, so dass sie einfach weiterverarbeitet werden können. Eine einheitliche Speicherung der Werte und Textinformationen und eine klare Trennung von Feldern und Datensätzen soll hergestellt werden. Auch eine Änderung der Reihenfolge ist oftmals sinnvoll. Dies muss für jeden Fall individuell entschieden werden, da das einzuhaltende Format vom gewählten Werkzeug, welches die Daten weiterverarbeiten soll, abhängig ist.

4.2.4. Modellierung

Hierunter ist die Modellierung der Daten als Vorbereitung zur Eingabe in eine Lernumgebung zu verstehen. Die Modellierung ist von der verwendeten Lernmethode abhängig. Eine Modellierung für neuronale Netzwerke stellt andere Anforderungen als die für Entscheidungsbäume. Dieser Schritt im KDD-Prozess umfasst auch die Einschätzung der Ergebnisse, die mit den erstellten Modellen erzielt wurden. Eine häufige Wiederholung dieses Schrittes ist in der Praxis erforderlich, da man meist nicht sofort das beste Modell für die Daten finden wird. Dies ist jedoch ein klarer Vorteil des CRISP-DM Modells, da man jederzeit zu vorhergehenden Schritten zurückspringen und die notwendigen Änderungen vornehmen kann.

Es ist hierbei auf gar keinen Fall untypisch, dass man während der Modellierung feststellt, dass weitere Informationen, die zum Beispiel bei der Selektion der Daten zu schnell verworfen wurden, benötigt werden und man somit vorangegangene Schritte erneut abarbeiten muss.

Diese Arbeit legt den Hauptaspekt auf die Modellierung, und obwohl dies nicht explizit erwähnt wird, mussten die vorangehenden Schritte häufig erneut durchgeführt werden. Die hier vorgestellten Modelle sind das Ergebnis des dynamischen Einsatzes des CRISP-DM Modells.

4.2.5. Evaluierung

Der vorangegangene Schritt sollte ein oder mehrere Modelle für eine gute Lösung der Data Mining-Problemstellung geliefert haben. Diese Phase soll nun dazu dienen, die erzielten Ergebnisse genauestens zu evaluieren, vor allem im Hinblick auf die ursprüngliche Zielsetzung. Es muss sichergestellt werden, dass kein wichtiger Aspekt bei der Modellierung vergessen wurde.

4.2.6. Anwendungsphase

Die Anwendungsphase kann nicht einheitlich charakterisiert werden. Die hier durchgeführten Aufgaben sind davon abhängig, welches Ergebnis durch das Data Mining erzielt werden soll. Es kann sein, dass lediglich ein Bericht über die durchgeführte Modellierung erstellt wird. Häufig wird aber ein zuvor exemplarisch durchgeführter Prozess durch speziell in diesem Schritt zu entwickelnde Software für weitere Daten der gleichen Form automatisiert. Dies kann sehr zeitaufwendig sein und erhöht die Bedeutung der vorangegangenen Schritte enorm, da die entwickelten Modelle als Mustervorlage für ein Softwareprodukt dienen.

Denkbar ist zum Beispiel eine monatliche Analyse und Vorhersage von Umfrageergebnissen, die dann durch Software gesteuert durchgeführt werden kann.

5. Lernverfahren

Dieses Kapitel stellt das Lernverfahren vor, welches zur Lösung der gestellten Lernaufgabe benutzt wurde. Dies wird zunächst über die Induktive Logische Programmierung motiviert, die in Abschnitt 5.1 informell definiert wird.

Abschnitt 5.4 führt den Begriff des Regellernens ein und 5.2 gibt eine Übersicht über logische Aspekte, die für das Verständnis der folgenden Lernverfahren benötigt werden. Der letzte Abschnitt dieses Kapitels (5.3) betrachtet das Lernverfahren RDT stellvertretend für relationale Lernverfahren im Detail.

5.1. Induktive Logische Programmierung (ILP)

Die Induktive Logische Programmierung (kurz: ILP)[1] ist ein Ansatz des maschinellen Lernens, der Techniken der logischen Programmierung anwendet [45, 47, 46]. Ziel dieser Theorie ist das Erlernen logischer Programme.

Ein großer Vorteil von ILP liegt darin, dass relationale Begrifflichkeiten sehr einfach ausgedrückt werden können, da eine eingeschränkte Prädikatenlogik verwendet wird. Weiterhin besteht die Möglichkeit, Hintergrundwissen für den Lernprozess anzugeben. Die Funktionslernaufgabe von ILP kann wie folgt angegeben werden [27]:

gegeben sind:

- eine Menge E der positiven und negativen Beispiele in einer Sprache L_E,
- Hintergrundwissen B in einer Sprache L_B, wobei $B \cup H \not\models \bot$

gesucht wird:

- eine Hypothese H in einer Sprache L_H, so dass die folgenden Bedingungen erfüllt sind:
  - Konsistenz: $B, H, E\neg \models \bot$
  - Vollständigkeit: $B, H \models E^+$
  - Korrektheit: $\forall e \in E^- : B, H\neg \models e$

Die Mengen B und E bestehen meist aus variablenfreien Fakten.

Die Induktive Logische Programmierung kann als Suche aufgefasst werden. Voraussetzung dafür ist, dass die Generalisierungsbeziehung so ausgedrückt werden kann, dass es für zwei Klauseln genau eine Generalisierung bzw. Spezialisierung gibt. Der Lernprozess besteht dann genau darin,

- Beispiele so lange zu generalisieren, bis alle positiven Beispiele bezüglich des Hintergrundwissens abgedeckt sind bzw.
- allgemeine Hypothesen so lange zu spezialisieren, dass keine negativen Beispiele mehr bezüglich des Hintergrundwissens abgedeckt sind.

[1] engl. inductive logic programming

Es ist also eine Generalisierungsbeziehung zu finden, die den Hypothesenraum so anordnet, dass der induktive Schluss als eine durch von Beispielen gesteuerte Suche in den Hypothesen, die einen Verband bilden, ausgedrückt werden kann. Weiterhin kann der Hypothesenraum in jedem Schritt beschnitten werden, d.h. Hypothesen werden aussortiert.

Die hier mit der Logik in Verbindung stehenden Begriffe und der notwendige Generalisierungsbegriff werden im folgenden Abschnit informell definiert.

5.2. Logische Grundlagen

In diesem Abschnitt werden einige Begriffe der Logik informell eingeführt. Für eine formale Definition wird hier auf [55] verwiesen.

Die Induktive Logische Programmierung ist über eine eingeschränkte Prädikatenlogik definiert. Daher sind die folgenden Definitionen erforderlich.

Definition 5.1 (Literal, Term, Variable). *Ein Literal $l = p(t_1, \ldots, t_n)$ besteht aus dem Prädikatensymbol p und den Termen $t_1, \ldots, t_n$. Die Variablensymbole der Terme werden, wie in [48], als Termvariablen bezeichnet, um sie von den Prädikatenvariablen zu unterscheiden. Variablen werden durch Großbuchstaben gekennzeichnet, Prädikatensymbole mit Kleinbuchstaben.*

Aus mehreren Literalen kann eine Klausel bzw. eine Hornklausel gebildet werden:

Definition 5.2 (Klausel). *Eine Klausel ist eine endliche Menge von Literalen. Üblich ist die Darstellung als Menge $\{\neg l_1, \ldots, \neg l_i, l_{i+1}, \ldots, l_n\}$ oder als Regel $l_1, \ldots, l_i \rightarrow l_{i+1}, \ldots, l_n$. Diese Umformung ist möglich, da $\neg A \vee B \Leftrightarrow A \rightarrow B$.*

Definition 5.3 (Hornklausel). *Eine Klausel heißt Hornklausel, wenn sie höchstens ein positives Literal enthält.*

Ein Spezialfall einer Hornklausel ist eine Regel:

Definition 5.4 (Regel). *Eine Klausel heißt Regel, wenn sie genau ein positives und mindestens ein negatives Literal enthält. Dies führt zu Aussagen der Form $l_1, \ldots, l_{n-1} \rightarrow l_n$.*

$l_1, \ldots, l_{n-1}$ bezeichnet man als Prämissen, l_n als Konklusion der Regel.

Insbesondere bezeichnet man Klauseln, die nur genau ein positives Literal enthalten, als Fakt. Sie bilden in der Induktiven Logischen Programmierung die positiven Beispiele der Wissensbasis.

Um ILP als Suche auffassen zu können, wird eine Generalisierungsbeziehung über den Klauseln benötigt. Oftmals wird in der Literatur die Implikation als Ordnung von Klauseln verwendet. Diese ist jedoch im allgemeinen Fall nicht entscheidbar (vgl. [34], Seite 140), so dass die sogenannte Subsumtionsbeziehung definiert wird. Dazu werden die Begriffe der Substitution und Unifikation benötigt.

Definition 5.5 (Substitution). *Eine Substitution $\theta = \{X_1|t_1, \ldots, X_n|t_n\}$ ist eine eindeutige Abbildung aus der Menge der Variablen in die Menge der Terme. Es wird nicht unifiziert, d.h. die Abbildung darf weder verschiedene quantifizierte Variablen noch quantifizierte und freie Variablen unifizieren.*

Eine Substitution ersetzt also gemäß Definition 5.5 innerhalb einer Klausel Termvariablen X_i durch die entsprechenden Terme t_i. Die Anwendung einer Substitution θ auf eine Klausel K schreibt man kurz $K\theta$.

Eine spezielle Substitution ist die Unifikation.

Definition 5.6 (Unifikation). *Wird eine Menge von Klauseln $\{A_1, \ldots, A_n\}$ durch eine Substitution σ zu einem äquivalenten Ausdruck substituiert, so bezeichnet man σ als Unifikator dieser Menge. Es gilt also*

$$A_1\sigma \equiv A_2\sigma \equiv \cdots \equiv A_n\sigma$$

Die Anwendung des Unifikators auf diese Menge bezeichnet man als Unifikation.

Beispiel 5.1 (Unifikator). *Gegeben seien die Klauseln*

$$A_1 = \{X, Y, f(Y)\}, \qquad A_2 = \{a, b, Z\}$$

Die Substitution

$$\sigma = \{X|a, Y|b, Z|f(b)\}$$

ist dann ein Unifikator dieser Klauseln.

Es ist anzumerken, dass nicht für jede Ausdrucksmenge ein Unifikator existiert.

Definition 5.7 (θ-Subsumtion). *Seien C und D Klauseln und θ eine Substitution. C subsumiert D ($C \vdash_\theta D$) genau dann, wenn $C\theta \subseteq D$ und $|C| \leq |D|$ gilt.*

Die Subsumtion liefert also die notwendige Generalisierungsbeziehung, um ILP als eine Suche auffassen zu können.

Die Subsumtion kann als korrekte Ableitungsrelation hergeleitet werden. Weitere Eigenschaften der Subsumtion mit ausführlichen Beispielen sind zu finden in [41] und [27].

Leider gibt es in der Terminologie der Logik auch den Begriff des 'Modells'. Dies ist im Rahmen dieser Arbeit besonders störend, kann jedoch nicht vermieden werden, da der Begriff des logischen Modells zur Definition des Regellernens unabdingbar ist. Eine formale Definition des Begriffes ist in [34] zu finden.

In dieser Arbeit ist jedoch die Definition des minimalen Modells wichtig und wird daher explizit hier aufgeführt:

Definition 5.8 (minimales Modell). *Eine Belegung I heißt ein minimales Modell von A, wenn I ein Modell von A ist und es keine Belegung I' gibt, die auch ein Modell von A ist und $I' \subset I$. Das minimale Modell von A wird formal aufgeschrieben als $\mathcal{M}^+(A)$.*

5.3. Regellernverfahren

Dieser Abschnitt erläutert das Regelentdeckungswerkzeug RDT (Rule Discovery Tool), welches als Lernverfahren im Anwendungsteil dieser Arbeit benutzt wird. Hierzu wird zunächst das Grundverfahren RDT vorgestellt, gefolgt von den beiden Weiterentwicklungen RDT/DB und RDT/DM.

Der folgende Abschnitt führt den Begriff des Regellernens und Lernverfahren ein, die unter dem Begriff 'Regellernverfahren' in der Literatur angeführt werden. Regellernverfahren sind Lernverfahren der Induktiven Logischen Programmierung.

5.4. Regellernen

Kietz definiert in [36] den Begriff des Regellernens. Definition 5.9 ist von dort übernommen.

Definition 5.9 (Regellernen)**.** *Es sei eine Menge von Beobachtungen E in einer Sprache $\mathcal{L}_E$ gegeben, Hintergrundwissen B in einer Sprache $\mathcal{L}_B$ sowie eine Hypothesensprache $\mathcal{L}_H$. $\mathcal{M}(H)$ bezeichne die Menge aller Modelle von H, wobei hier Modelle in einer Logik gemeint sind.*

Gesucht wird eine Hypothesenmenge H (Regeln) in $\mathcal{L}_H$ mit den folgenden Eigenschaften:

Gültigkeit: *$\mathcal{M}^+(B \cup E) \subseteq \mathcal{M}(H)$, d.h. H ist in allen minimalen Modellen von B und E wahr.*

Notwendigkeit: *$\forall h \in H : \exists e \in E : B, E\backslash\{e\}$ und $B, E\backslash\{e\}, h \models e$, d.h. alle Hypothesen enthalten neue Informationen über die Beobachtungen.*

Vollständigkeit: *$\forall h \in H$, die gültig und notwendig sind: $H \models h$, d.h. alle in B und E wahren Hypothesen folgen aus H.*

Minimalität: *$\nexists G \subset H : G$ gültig und vollständig.*

Weiterhin legte Kietz auch die Lernaufgabe für das in den folgenden Abschnitten betrachtete Lernverfahren RDT fest. Er definierte das Lernverfahren über funktionsfreien, k-literalen Hornklauseln für die Sprache der Hypothesen. Aus diesem Grund gibt der folgende Abschnitt einen kurzen Abriss über logische Grundlagen, insbesondere über die hier genannten Begrifflichkeiten. Trotzdem sind weitere Bücher zu diesem Thema zur Vertiefung der Aspekte zu Rate zu ziehen (zum Beispiel [34, 55]).

5.4.1. RDT

RDT ist die Abkürzung für 'Rule Discovery Tool' (Regelentdeckungswerkzeug), welches das effiziente Lernen von Regeln auf der Basis von Fakten ermöglicht. Die erste Fassung wurde in die Werkbank MOBAL [40] integriert.

Die Hypothesensprache $\mathcal{L}_H$ wird dabei explizit in ihrer Syntax beschränkt, wobei diese Beschränkung durch den Anwender deklariert und angepasst werden kann. Dies geschieht durch die Angabe von Wissen, welches aus einer Menge von Hypothesen besteht. Man spricht hierbei auch von Lernen mit deklarativem Bias. Dies wird als Charakteristikum der Induktiven Logischen Programmierung angesehen.

Die syntaktische Form der Hypothesen wird durch Regelschemata eingeschränkt. Zusätzlich wird die Semantik der Hypothesen über eine Prädikatentopologie und eine Sortentaxonomie festgelegt.

Die folgenden Unterabschnitte geben einen kurzen Einstieg in diese Begrifflichkeit. Auf [48] wird für einen ausführlichen Überblick und alle Definitionen verwiesen. Die hier abgedruckten Definitionen sind teilweise von dort übernommen.

Weitere Literaturangaben zu den einzelnen Begriffen sind in den Unterabschnitten angegeben.

Regelschemata

Ein Regelschema bestimmt die syntaktische Form einer Regel, die durch das Lernverfahren gefunden werden kann.

Definition 5.10 (Regelschema)**.** *Ein Regelschema $\mathcal{RS}$ hat den Aufbau einer Regel. Mindestens ein Prädikatensymbol wurde durch eine Prädikatenvariable ersetzt. Ein Regelschema führt neben dem eigentlichen Schema eine Bezeichnung und Angaben zu allen benutzten Prädikatenvariablen und Konstanten mit sich.*

Beispiel 5.2.

$$schema1(C, P1, P2, Q) : P1(X, Y)\&P2(X, C) \rightarrow Q(Y)$$

Das obige Regelschema hat die Bezeichnung 'schema1' und enthält, neben den Prädikatenvariablen P1, P2 und Q, die Konstante C. Der Bezeichner und die verwendeten Variablen werden vom eigentlichen Regelschema durch einen Doppelpunkt getrennt.

Nun ist es erforderlich, Relationsketten und die Tiefe solcher Ketten zu definieren. Die Definitionen für Relationskette, Tiefe und Prämissenordnung sind in [48], Seite 21 ff. zu finden und müssen hier nicht erneut aufgeführt werden. Für diese Arbeit ist der Aspekt wichtig, dass der RDT-Algorithmus die Regelschemata schrittweise instanziiert und dies nur mit Hilfe einer Ordnung der Regelschemata (Prämissenordnung) geschehen kann, die mit Hilfe der Relationsketten und der dazugehörigen Tiefe definiert wird.

Prädikatentopologie

Bei der Prädikatentopologie handelt es sich um eine semantische Einschränkung des Hypothesenraumes. Hierbei wird in Hinblick auf den Verwendungszweck des Lernergebnisses die Menge der Prädikate $\mathcal{P}$ in ggf. nicht-disjunkte Mengen T_i einer Gruppierung $\mathcal{T} = \{T_1, \ldots, T_m\}$ zerlegt. T_i bezeichnet man dabei als Topologieknoten. Diese können vom Benutzer mit Bezeichnungen versehen und mittels gerichteten Kanten zu einer Hierarchie verbunden werden. Somit kann der Bereich der Suche für einen bestimmten Zweck angepasst werden.

Sortentaxonomie

Sorten sind eine weitere semantische Einschränkung und unterscheiden sich hier nicht vom Begriff der 'Sorte' in der Logik (vgl. [55]). Sie werden den Termvariablen eines Prädikats zugewiesen. Diese Zuweisung erfolgt während der Instanziierung der Regelschemata, wobei das Binden eines Prädikates die Sorten der Termvariablen bestimmt. Auf diese Weise können nur weitere Prädikate mit Termvariablen der gleichen Sorte für dieses an eine Sorte gebundene Prädikat ausgewählt werden.

Deklarativer Bias

RDT ist wie bereits erwähnt ein Lernverfahren mit deklarativem Bias. Dieser kann nun mit Hilfe der Begriffe aus den vorangegangenen Abschnitten präzise für das Lernverfahren RDT definiert werden.

Definition 5.11 (Deklarativer Bias)**.** *Der deklarative Bias wird unterteilt in den syntaktischen und semantischen Bias.*

syntaktischer Bias: *Er bestimmt die syntaktische Form der Hypothesen. Bei RDT übernehmen die Regelschemata diese Rolle.*

semantischer Bias: *Er bestimmt den logischen Aufbau der Hypothesen. RDT nutzt hierfür die Prädikatentopologie und die Sortentaxonomie.*

Hypothesengenerierung

Die Erstellung einer Regel aus einem Regelschema nennt man Instanziierung des Regelschemas. Jede so erstellte Regel ist zunächst eine Hypothese, die getestet werden muss. Der Test einer Hypothese wird im nächsten Abschnitt beschrieben. Zunächst wird die Instanziierung definiert.

Definition 5.12. *Sei P eine Prädikatenvariable, p ein Prädikatensymbol und $\mathcal{RS}$ ein Regelschema. Eine Instanziierung Σ ist eine endliche Menge von Paaren $P|p$, wobei P und p genau die gleiche Anzahl von Termvariablen besitzen. $\mathcal{RS}\Sigma$ bezeichnet die Anwendung der Instanziierung Σ auf das Regelschema $\mathcal{RS}$. Bei der Instanziierung werden alle Prädikatenvariablen durch die zugeordneten Prädikatensymbole ersetzt. Konstanten werden erst nach der Instanziierung der Prädikatenvariablen betrachtet, indem alle möglichen Belegungen für die Konstanten in den Prädikaten eingesetzt werden.*

Es ist hervorzuheben, dass der RDT-Algorithmus eine schrittweise Instanziierung gemäß der Prämissenordnung durchführt. Es wird dadurch immer ein Prädikat instanziiert, welches über bereits instanziierte Prädikate mit der Konklusion verbunden ist. Daraus resultieren drei Vorteile:

- Es werden nur notwendige Instanziierungen durchgeführt.
- Die Auswahl der möglichen Instanziierungen ist durch die verbindenden Prädikate eingeschränkt.
- Durch die inkrementelle Instanziierung kann der Hypothesentest bereits auf teilweise instanziierten Hypothesen durchgeführt werden. Falls dieser instanziierte Teil bereits den Test nicht besteht, so ist eine weitere Instanziierung der Hypothese hinfällig, da auch dadurch die Hypothese den Test nicht bestehen kann.

Beispiele zur Instanziierung sind in der Arbeit von Münstermann [48] zu finden.
Als letzten wichtigen Teilaspekt zum Verständnis des RDT-Algorithmus wird im nächsten Abschnitt der Hypothesentest vorgestellt.

Hypothesentest

RDT testet sowohl vollständig als auch teilweise instanziierte Hypothesen auf Akzeptanz. Vollständig instanziierte Hypothesen müssen ein Akzeptanzkriterum erfüllen, für teilweise instanziierte Hypothesen wird ein Beschneidungskriterium definiert, welches zu spezielle Hypothesen verwirft.

RDT verwendet spezielle Bewertungsmaße, die in [48], Seite 24 ff., vollständig aufgelistet sind. Für die Bewertungsmaße werden Mengen mit Hypothesen gebildet, die bestimmte Kriterien erfüllen. Ein Akzeptanzkriterium benutzt dann die Kardinalitäten dieser Mengen und verbindet diese mit Hilfe von Vergleichs- $(=, >, <, \dots)$und Veränderungsoperatoren $(+, -, \dots)$. Konjunktion und Disjunktion von Ausdrücken sind ebenfalls möglich. Das Beschneidungskriterium kann meist durch Negation des Akzeptanzkriteriums angegeben werden. Beispiele für Akzeptanz- und Beschneidungskriterium sind in [48] auf Seite 25 ff. zu finden.

Ablaufbeschreibung des Algortihmus

Der vollständige Algorithmus in Pseudo-Code ist zu finden in [36]. Diese Arbeit beschränkt sich auf die Kernaspekte des Ablaufes, d.h. Implementierungsaspekte (wie z.B. Datenstrukturen) werden nicht beschrieben.

Man kann den Algorithmus in drei Hauptschritte unterteilen:

1. Bildung der Hierarchie der Regelschemata
2. Generierung der Hypothesen (Instanziierung)
3. Hypothesentest

Der Begriff der Hierarchie von Regelschemata ist noch zu klären. Dazu definieren wir eine Ordnung über die Regelschemata in Form einer Generalisierungsbeziehung.

Definition 5.13 (Generalisierungsbeziehung)**.** *R und R' seien Regelschemata, σ eine Substitution bezüglich Termvariablen und Σ bezüglich Prädikatenvariablen. Dabei darf Σ verschiedene Prädikatenvariablen nicht gleichsetzen. R ist genau dann genereller als R', wenn gilt:*

$$\exists\, \sigma,\ \Sigma\ :\ R\,\sigma\,\Sigma \subseteq R'$$

Wir schreiben dann $R \geq_{\mathcal{RS}} R'$.

Ein Beispiel für eine Hierarchie ist in [41] zu finden.

Der genaue Ablauf soll nun beschrieben werden: Der Benutzer gibt ein Zielprädikat Q vor. Es werden dann alle Regelschemata ausgewählt, deren Konklusion durch das Zielprädikat substituierbar ist. Die Hierarchie wird nur für diese ausgewählten Regelschemata gebildet. Es erfolgt dann eine Top-Down-Breitensuche, ausgehend von einem generellsten Regelschema. Ist dieses abgearbeitet, so wird das nächste Regelschema im Algorithmus gemäß der Ordnung ausgewählt. Auf diese Weise werden nur die als zu generell spezifizierten Hypothesen weiter untersucht.

Die Hypothesengenerierung erfolgt insofern schrittweise, da während der Instanziierung der einzelnen Prädikate immer direkt gegen das Beschneidungskriterium überprüft wird. Zu spezielle Hypothesen werden direkt aussortiert. Ist ein Regelschema vollständig instanziiert, so wird der Akzeptanztest durchgeführt.

Modellbasierter Ansatz

Durch die freie Gestaltungsmöglichkeit der Hypothesensprache kann man auch von einer Modellierung im Bereich des maschinellen Lernens sprechen. Aus diesem Grund bietet

sich RDT im Rahmen dieser Arbeit als Lernverfahren geradezu an, da die Modellierung des begrifflichen Wissens im Vordergrund steht und diese Modellierung sogar in die Lernphase übernommen werden kann. Die Modellierung ist in der Software MiningMart erfolgt. Alle Informationen werden bei MiningMart in einer Datenbank abgelegt. Somit muss eine Brücke von der Datenbank zur Lerneingabe für das Lernverfahren RDT geschaffen werden. Dies wird durch eine spezielle Implementierung von RDT, namens RDT/DB, möglich.

5.4.2. RDT/DB

RDT/DB ist eine Weiterentwicklung von RDT, die die Analyse einer Datenbank ermöglicht. Dabei wird das Modellwissen aus Tabellen einer Datenbank bezogen, d.h. sämtliche Prädikate des Hypothesenraums sind aus den Informationen in der Datenbank zu bilden und auf dieser zu testen. Hypothesengenerierung und Hypothesentest finden also vollständig auf der Datenbank statt.

Grundlegend hierfür ist die Abbildung aus Definition 5.14, die in den folgenden Abschnitten als Mapping bezeichnet wird.

Definition 5.14 (Mapping)**.** *Sei $\mathcal{T}$ die Menge aller Tabellen der Datenbank und P die Menge aller Prädikate. Dann wird eine Abbildung der folgenden Form als Mapping bezeichnet:*

$$map : \mathcal{T} \rightarrow \mathcal{P}$$

RDT/DB führt insgesamt vier verschiedene Mappings ein, um Tabellen auf Prädikate abzubilden. Weiterhin sind für den Hypothesentest die Hypothesen in SQL-Anweisungen bzw. Anfragen umzuwandeln, um sie auf der Datenbank auswerten zu können. Die Mappings als auch das Vorgehen zur Erstellung der SQL-Anweisungen sind in [5] ausführlich beschrieben.

5.4.3. RDT/DM

In [48] entwickelt Dirk Münstermann das Lernverfahren RDT/DB weiter. Münstermann versucht die Laufzeit von RDT/DB durch den Einsatz von sog. Sichten in der Datenbank und eine dynamische Anpassung des Hypothesentests zu verbessern. Weiterhin werden die Metainformationen der Tabellen in der Datenbank benutzt, um Fremdschlüsselbeziehungen auszulesen und für den Lernvorgang zu verwenden. Dies bedeutet, dass die in RDT/DB eingeführten Mappings an diese neuen Ansätze angepasst werden müssen.

Grundsätzlich werden bei einer Tabelle, die keinerlei Fremdschlüssel besitzt, die gleichen Mappings benutzt wie bei RDT/DB. Für Tabellen, die Fremdschlüsselbeziehungen aufweisen, führt Münstermann erweiterte Abbildungen ein, die ausführlich in [48] mit Beispielen erläutert werden. Änderungen an den SQL-Anweisungen, die für den Hypothesentest notwendig sind, ergeben sich somit ebenfalls und werden auch dort aufgeführt.

Weiterhin führt er eine vollständige Neuimplementierung in Java mit Anbindung an beliebige Datenbanken mit Hilfe der JDBC-Schnittstelle durch. An dem Kernalgorithmus des Regelentdeckungswerkzeugs RDT wurden jedoch gegenüber RDT/DB keinerlei Veränderungen vorgenommen.

Es ist anzumerken, dass mit diesem Lernverfahren alle Experimente dieser Arbeit durchgeführt werden, wobei einige Anpassungen vorgenommen werden mussten, die in Abschnitt 8.1 im Anhang dieser Arbeit nachzulesen sind.

6. MiningMart

Dieses Kapitel gibt einen Überblick über das MiningMart-Projekt [42, 43, 44]. Dazu wird zunächst in Abschnitt 6.1 die Grundidee zur Entwicklung von MiningMart geschildert. Einzelne Bestandteile der daraufhin entwickelten und zur Bearbeitung des Fallbeispiels eingesetzten Software werden in Abschnitt 6.2 eingeführt.

Den Abschluss des Kapitels bildet ein detaillierter Einblick in die in MiningMart zur Verfügung stehenden Data Mining-Operationen (Operatoren) in Abschnitt 6.3.

Es ist anzumerken, dass eine neue Version der MiningMart-Software verwendet wurde, die während der Erstellung dieser Arbeit zeitgleich am Informatik-Lehrstuhl 8 der Universität Dortmund entwickelt wurde. Der Grundaufbau der Software und die zugrunde liegenden Modelle haben sich aber nicht geändert. Es wurden einige technische Erweiterungen vorgenommen, die für diese Arbeit jedoch keinerlei Bedeutung haben.

Zum Erstellungszeitpunkt dieser Arbeit wurde auch [22] ausgearbeitet. Alle Modellierungen wurden dort ebenfalls mit dieser MiningMart-Version durchgeführt, so dass diese Quelle weitere Aspekte zur aktuellen Implementierung von MiningMart liefert.

Die Ideen und Konzepte aus den angeführten Quellen aus früheren Jahren sind aber weiterhin relevant und bieten zudem einen guten Einstieg in die MiningMart-Thematik. Dazu vermitteln sie einen generellen Überlick zum MiningMart-Projekt und eine Einordnung in andere Aspekte der Wissensentdeckung in Datenbanken.

6.1. Grundidee

In dieser Arbeit wurde bereits das CRISP-DM Modell in Kapitel 4 eingeführt und ausführlich erläutert.

Pyle sagt in [51], dass die Datenvorverarbeitung $50-80\%$ des gesamten KDD-Prozesses benötigt.

Vor allem benötigen die folgenden Schritte die meiste Zeit bei der Datenvorverarbeitung:

- Auswahl der Lernaufgabe
- Sampling der Daten
- Merkmalsgenerierung und Merkmalsauswahl
- Datenbereinigung
- Definition von geeigneten Kriterien für die Evaluation.

Weiterhin führt Pyle als Grund an, dass viele dieser Vorverarbeitungsschritte immer wieder neu, für jeden Fall individuell, implementiert werden müssen. Jeder Benutzer, der eine Wissensentdeckung durchführen möchte, beginnt somit nahezu jedes Projekt, ohne Vorwissen anderer Benutzer direkt auf die vorliegenden Daten anwenden zu können.

Daher hat das MiningMart-Projekt das Ziel, die Datenvorverarbeitung und die Auswahl der Techniken zu vereinfachen. Weiterhin soll auch die Qualität der Datenvorverarbeitung verbessert werden.

MiningMart basiert daher auf dem Prinzip des Fallbasierten Schließens [37]. In [1] wird neben einem Überblick über die Entwicklungsgeschichte des Fallbasierten Schließens (kurz: CBR[1]) ein CBR-Zyklus vorgestellt. Die vier Kernaspekte mit den englischen Schlagwörtern dieses Zyklusses sind:

-RETRIEVE:	Ermitteln des ähnlichsten Falles aus anderen bekannten Fällen
-REUSE:	Benutzen der Informationen und des Wissens dieses Falles, um das neue Problem zu lösen
-REVISE:	Überprüfen der gefundenen Lösung
-RETAIN:	Aufbewahren der Teile dieser Problemlösung, die wahrscheinlich auch für die Zukunft für andere Problemlösungen interessant sein können.

Dieser Ansatz hat sich als erfolgreich in vielen Bereichen der Künstlichen Intelligenz erwiesen, so z.B. in der Erkennung von Betrugsversuchen bei der Gewährung von Krediten [63].

Ein weiteres Beispiel ist der Einsatz bei Support-Hotlines. Kunden schildern ihr Problem und mit Hilfe von CBR wird auf ein ähliches bzw. genau dasselbe Problem geschlossen.

6.2. Aufbau und Bestandteile

MiningMart bietet eine grafische Benutzerführung an, die die einfache Datenvorverarbeitung erlaubt. Dazu werden sogenannte Vorverarbeitungsketten, im Folgenden nur noch kurz mit Ketten bezeichnet, modelliert. Alle Ketten werden dabei innerhalb eines Falles[2] modelliert. Die Modellierung der Fälle erfolgt, genau wie die der Daten, in einem Meta-Modell. Dadurch können Fälle (bzw. einzelne Ketten aus ihnen) unabhängig von den modellierten Daten auf andere, ähnliche Fälle übertragen werden. Dazu bietet MiningMart eine Schnittstelle zum World Wide Web, so dass Fälle publiziert und frei zugänglich gemacht werden können [23]. Die Universität Dortmund bietet zur Zeit eine Falldatenbank auf ihrer Internetseite an [59]. Dort sind zahlreiche Fälle von vorangegangenen Projekten und Arbeiten zu finden, wie zum Beispiel [38, 20].

Fälle werden in MiningMart mittels zwei Editoren erstellt. Dabei werden Konzepte im Konzepteditor[3], Ketten im Fall-Editor[4] modelliert.

[1] engl. case-based reasoning
[2] engl. case
[3] engl. concept editor
[4] engl. case editor

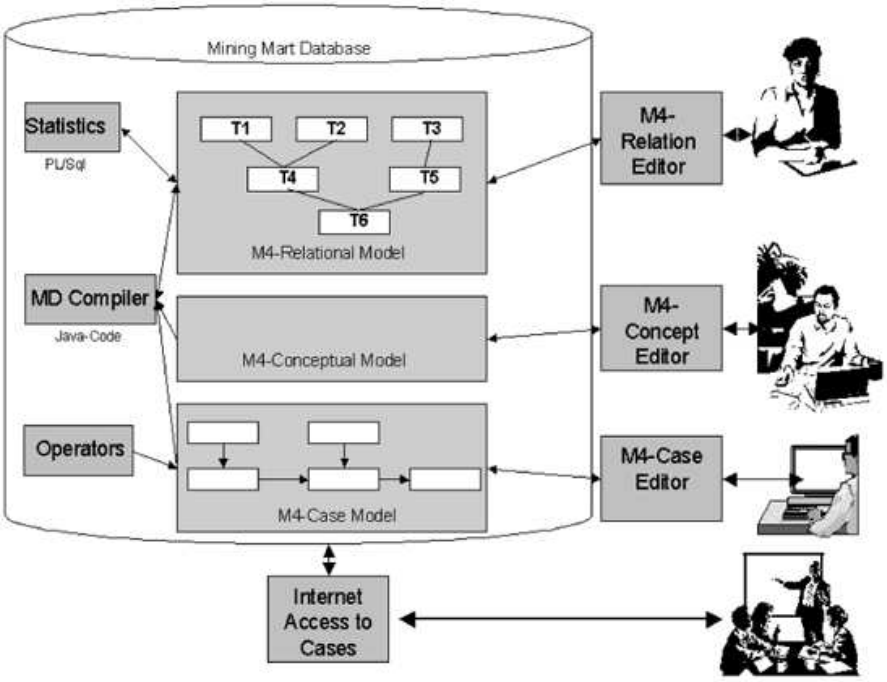


Abbildung 6.1.: Architektur von MiningMart. Den Kern von MiningMart bildet das M4 Meta-Datenmodell.

Der Kern von MiningMart besteht aus dem M4 Meta-Datenmodell. Es handelt sich um ein Meta-Modell der Meta-Daten. Es werden sowohl Informationen über die zu analysierenden Daten als auch Informationen über den eigentlichen Prozess der Wissensentdeckung selbst in ihm abgelegt. Aus diesem Grund muss zwischen zwei verschiedenen Arten von Daten unterschieden werden:

- Unter dem Begriff 'Geschäftsdaten'[5] werden die zu analysierenden Daten zusammengefasst.
- Die Meta-Daten, die den Prozess der Wissensentdeckung auf den Geschäftsdaten beschreiben. Dieser Teil der Daten wird im Folgenden als der 'Fall' oder eine Sammlung von Ketten bezeichnet. Diese Daten können weiter unterteilt werden:
  - Daten zur konzeptuellen Ebene: Diese beschreiben die Konzepte und deren Beziehungen zueinander, die in MiningMart modelliert wurden.
  - Daten zur relationalen Ebene: Einige Konzepte müssen auf Informationen aus Datenbanktabellen auf der relationalen Ebene zugreifen können, um den Zugriff der Konzepte auf diese Ebene zu ermöglichen. Daher müssen auch Daten zu der relationalen Ebene im M4-Modell abgelegt werden.

Diese Trennung ist nicht nur theoretischer Natur, sondern wird vielmehr in MiningMart konsequent umgesetzt, indem zwei unterschiedliche Bereiche im Datenbanksystem für die Fall- und Geschäftsdaten eingerichtet werden. Somit können die Ketten eines Falles oder

[5] engl. business data

sogar ein gesamter Fall auf beliebige Geschäftsdaten angewendet werden. Eine Kapselung einzelner Ketten in Schablonen[6] ist bereits angedacht, steht jedoch zur Zeit noch nicht vollständig zur Verfügung.

Beispiel 6.1. *Gegeben seien die Daten einer Versicherungsgesellschaft. Diese bilden in MiningMart die Geschäftsdaten. Bei der Datenvorverarbeitung wird eine Kette erstellt, die Aggregationen auf einigen numerischen Werten durchführt, um zum Beispiel die gesamte Versicherungssumme einer Person oder Ähnliches zu berechnen.*

Diese Kette kann nun von einer anderen Firma auf eigene Geschäftsdaten angewendet werden, um z.B. Zahlungsforderungen auf die gleiche Weise zu aggregieren.

Nutzen nun beide Firmen die MiningMart-Casebase, so müsste nur eine Firma die Kette modellieren und publizieren.

Selbstverständlich kann eine einzige Firma bereits durch geschickte Strukturierung von Ketten Schritte bei der Modellierung verkürzen, da einige Ketten auf mehrere Konzepte angewendet werden können.

Die Kernaspekte des Fallbasierenden Schließen sind hier unmittelbar zu erkennen.

Genauer gesagt basiert das konzeptuelle Datenmodell von MiningMart auf einer Ontologie (vergleiche Kapitel 3 auf Seite 19 und [24]).

Eine schematische Darstellung der Architektur ist in Abbildung 6.1 auf der vorherigen Seite zu sehen. Dazu ist in Abbildung 6.2 auf der nächsten Seite eine vereinfachte Darstellung des M4-Modells als UML-Modell angegeben. Dort ist zu erkennen, welche Informationen erfasst werden und welche Beziehungen diese zueinander besitzen.

6.2.1. Konzepteditor

Der Konzepteditor ermöglicht das Modellieren der Konzepte eines Falles [13]. Dabei können Beziehungen zwischen den Konzepten modelliert werden.

Konzepte können auf Basis von Tabellen auf der relationalen Ebene erzeugt werden. Dazu gibt es Konzepte, die durch einen Operator in einer Kette erstellt worden sind. Diese werden durch zwei gekreuzte Hämmer dargestellt. Weiterhin kann man zum Beispiel durch Projektionen sehen, welches Konzept aus welchem anderen Konzept hervorgegangen ist. Abbildung 6.3 auf Seite 49 zeigt einen Ausschnitt eines im Rahmen dieser Arbeit erstellen Falles.

Ebenso können genaue Informationen zum Aufbau eines Konzeptes eingesehen werden. Dies gilt auch für Informationen aus der relationalen Ebene, falls ein Konzept dort den Ursprung hat. Dazu kommt die Möglichkeit, alle Daten eines Konzeptes durchblättern zu können (vergleiche Abbildung 6.4 auf Seite 50). Auch übliche Statistiken können direkt zu einem Konzept abgerufen werden (vergleiche Abbildung 6.5 auf Seite 51). Zudem können Konzepte beliebig angeordnet werden, so dass die Erstellung übersichtlicher und verständlicher Modelle leicht fällt.

6.2.2. Fall-Editor

Im Fall-Editor, der in Abbildung 6.6 auf Seite 52 zu sehen ist, werden die Ketten modelliert. Ein Fall ist dabei eine Abfolge von Schritten, die aus Operatoren bestehen, die bestimmte Operationen auf den Konzepten durchführen.

[6] engl. template

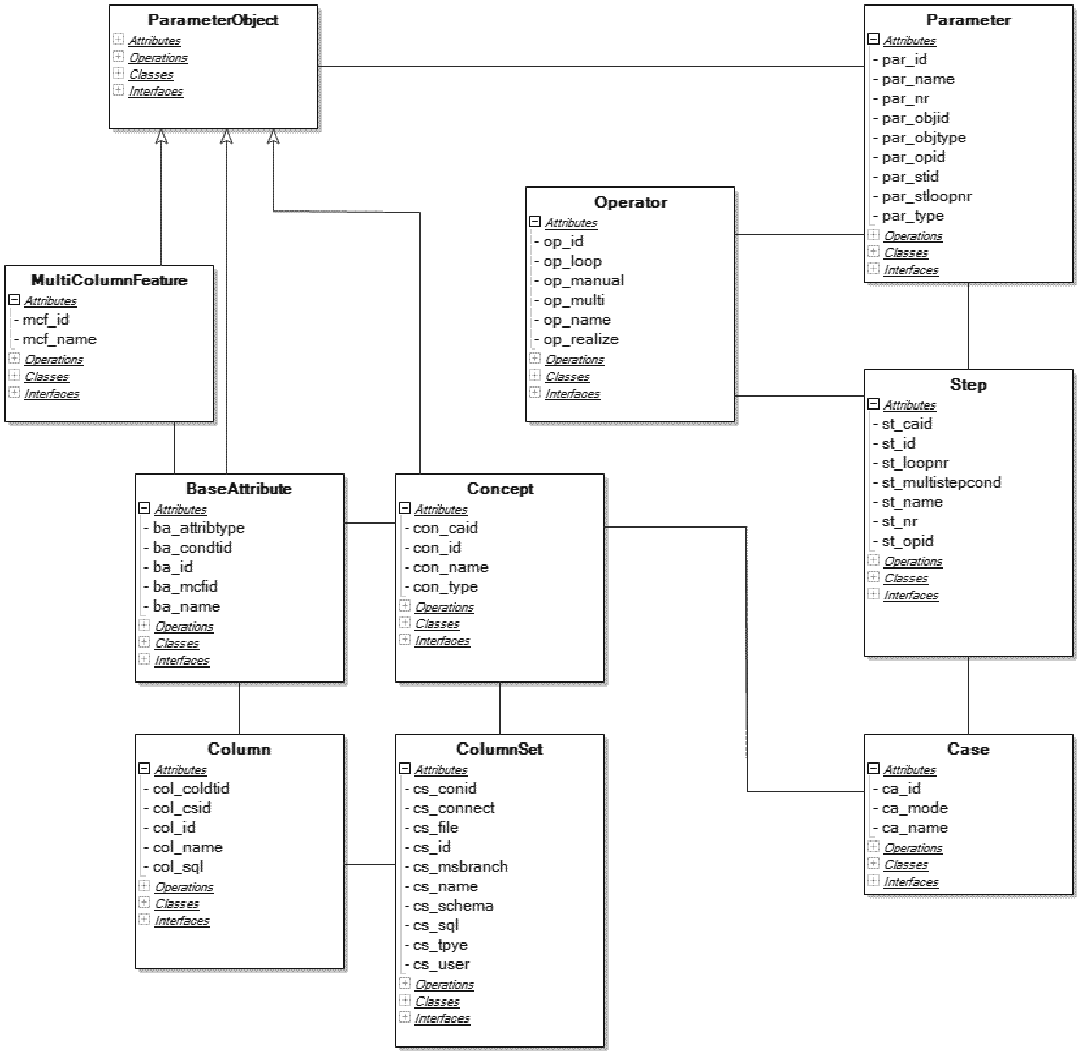


Abbildung 6.2.: Vereinfachtes UML-Modell des Meta-Modells von MiningMart, welches kurz mit M4 bezeichnet wird

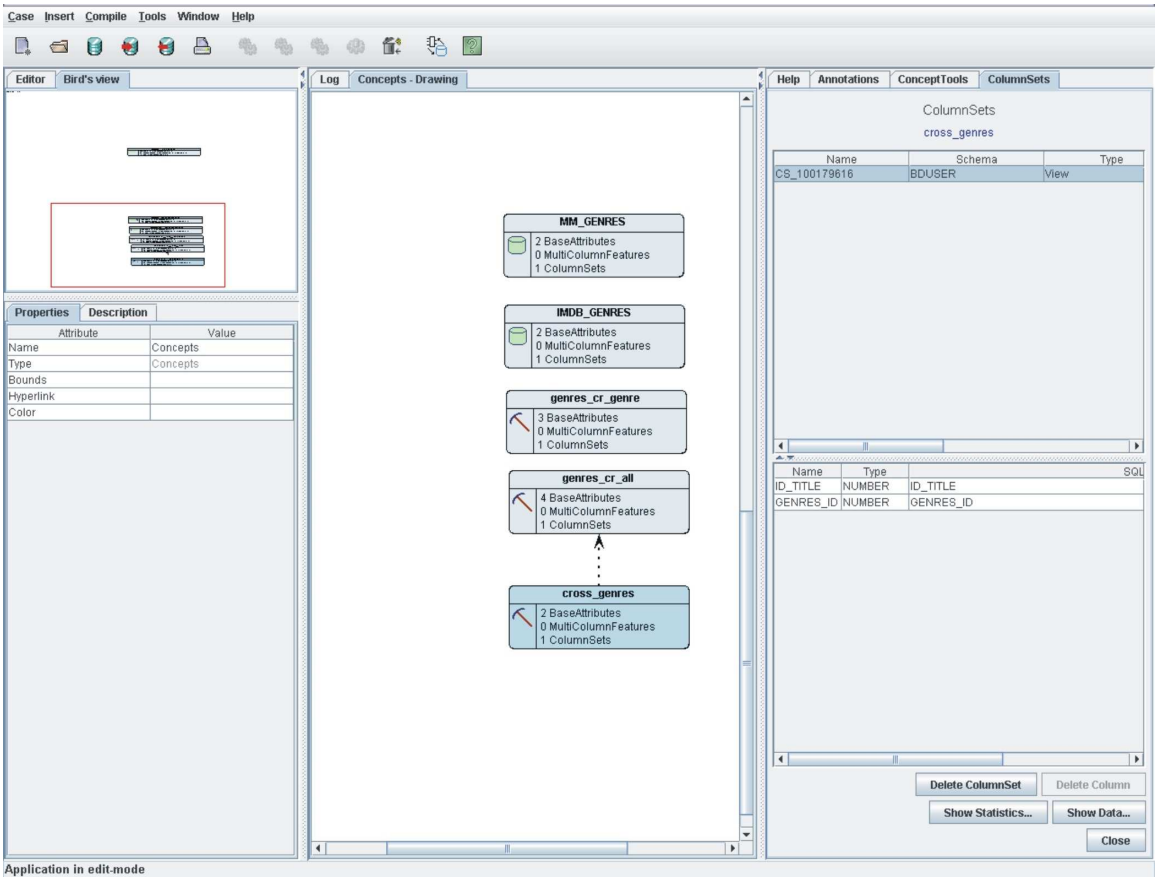


Abbildung 6.3.: Der Konzepteditor in MiningMart: Der Konzepteditor zeigt alle Konzepte eines Falles oder einer Kette an. Beziehungen zwischen den Konzepten werden durch Verbindungslinien dargestellt. Zu jedem Konzept können die Relationen und Attribute angezeigt bzw. bearbeitet werden.

Operatoren werden ausgewählt und mit gerichteten Kanten miteinander verbunden. So können beliebige Wissensentdeckungsprozesse modelliert werden. Zusätzliche Struktur kann in der Modellierung durch Schachtelung der Ketten erzielt werden. So kann eine Kette beliebig viele untergeordnete Ketten haben. Hervorzuheben ist, dass die Konzepte, die in untergeordneten Ketten erstellt werden, in den übergeordneten Ketten in die Modellierung mit einbezogen werden können. Somit kann zusätzliche Abstraktion erzielt werden, da die oberen Ketten nur die Hauptmodellierungsschritte enthalten und die untergeordneten Ketten die Details behandeln. Abbildung 6.7 auf Seite 53 zeigt ein Schema, wie Ketten geschachtelt im Fall-Editor modelliert werden können.

Es sei davor gewarnt, eine zu hohe Schachtelungstiefe bei der Modellierung zu benutzen. Die Beispielketten dieser Arbeit haben eine maximale Schachtelungstiefe von 3. Grund hierfür ist, dass bei vielen verschiedenen geschachtelten Ketten zwar die Ketten kurz und schnell verständlich sind, Fehler in der Modellierung aber schwer nachvollziehbar und Änderungen ebenfalls mühselig durchzuführen sind, da es schwierig ist, einen bestimmten Operator in mehreren Ebenen zu finden.

Neben der eigentlichen Modellierung können Operatoren mit aussagekräftigen Namen

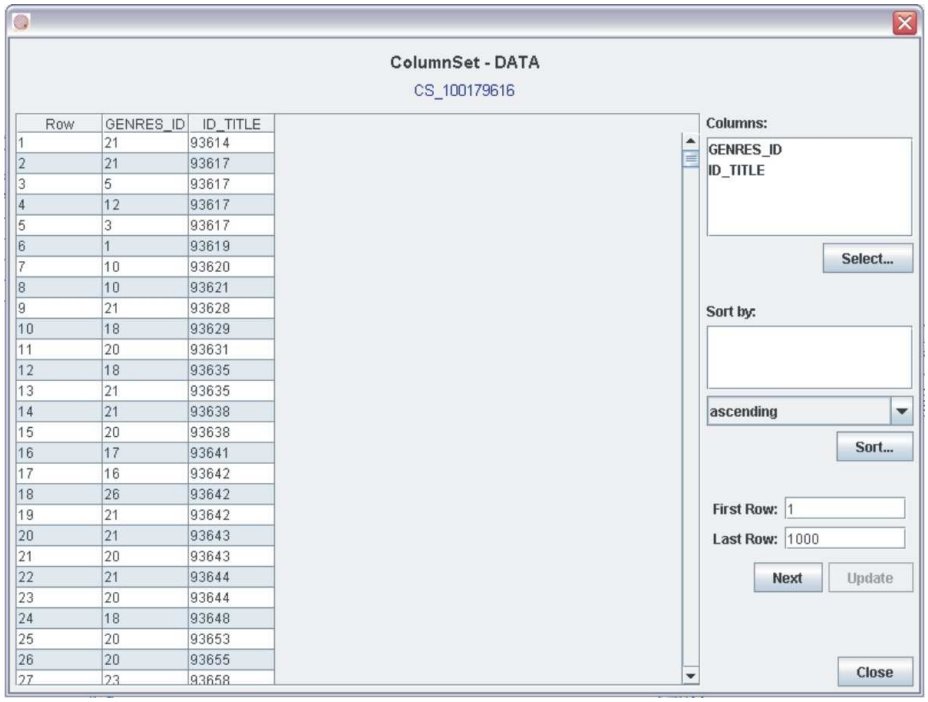


Abbildung 6.4.: Anzeige der Daten eines Konzeptes: Daten aller Attribute eines Konzeptes können eingesehen werden. Dabei können der anzuzeigende Bereich und die Sortierung frei festgelegt werden. Bei Selektionen und Merkmalsgenerierungen ist diese Ansicht besonders hilfreich, da Zwischenschritte in Ketten einfach daraufhin überprüft werden können, ob die Konzepte die gewünschten Attribute aufweisen.

versehen werden. Dazu können alle Elemente im Editor mit beliebig langen Kommentaren versehen werden.

Auch die Operatoren, die im nächsten Abschnitt ausführlich beschrieben werden, werden im Fall-Editor konfiguriert, d.h. die Parameter mit entsprechenden Werten belegt.

Der Editor bietet zudem eine komfortable Validierung an, so dass fehlerhafte Eingaben, wie zum Beispiel Tippfehler bei Attribut- oder Konzeptnamen, bereits vor der Compilierung erkannt werden können.

Besonders komfortabel ist die Funktionalität von MiningMart, dass man nicht nur jederzeit vom Fall-Editor in den Konzepteditor wechseln kann, sondern dass man die Anzeige der Konzepte im Konzepteditor auf die Konzepte beschränken kann, die zur Zeit im Fall-Editor bearbeitet werden. So kann man sich schnell einen Überblick verschaffen, ob die korrekten Konzepte in die Kette einbezogen bzw. erzeugt werden.

6.3. Operatoren

Im letzten Abschnitt wurde der Aufbau von MiningMart im Detail beschrieben. Es wurde der Begriff der 'Operatoren', die eine Operation auf einem Konzept durchführen, eingeführt. Euler [21, 22] unterscheidet in seiner Arbeit zwischen 'primitiven' (engl. primitive)

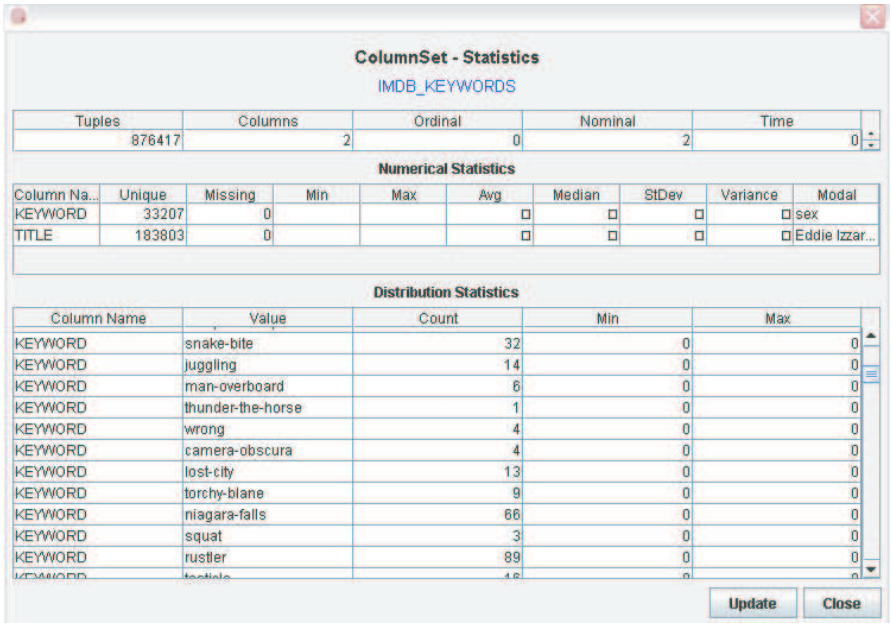


Abbildung 6.5.: Statistiken eines Konzeptes: Zu jedem Konzept in MiningMart können ausführliche Statistiken abgerufen werden. Dies gilt sowohl für Konzepte, die mit einer Tabelle in der Datenbank verknüpft sind, als auch für Mining-Konzepte, die z.B. als Ausgabe eines Operators erzeugt wurden. Bei Ausgaben von Operatoren handelt es sich meist um Sichten (Views) im verwendeten Datenbanksystem. Die Statistiken sind nicht nur zur ersten Sichtung und Analyse der Daten geeignet, sondern bieten auch eine gute Möglichkeit, Zwischenschritte im KDD-Prozess manuell zu überprüfen.

und 'zusammengesetzten' (engl. convenience) Operatoren. Diese Arbeit wird in Unterabschnitt 6.3.1 bzw. 6.3.2 beide Gruppen vorstellen.

Ein weiteres Unterscheidungskriterium, das es vor allem bei der Benutzung von MiningMart zu beachten gilt, ist, dass es zum einen Operatoren gibt, die ein neues, sogenanntes Zielkonzept, erstellen und zum anderen auch Operatoren, die das Ausgangangskonzept direkt modifizieren. Die folgenden Unterabschnitte werden explizit angeben, ob ein neues Konzept, basierend auf dem Ursprungskonzept, erstellt wird oder ob das bereits vorhandene Konzept, auf welches der Operator angewendet wird, modifiziert wird.

Andere Aspekte zur Unterscheidung und Kategorisierung von Operatoren sind sicherlich denkbar, sollen aber nicht weiter angeführt werden. Tabelle 6.1 enthält eine Übersicht zu den Begriffen und den wichtigsten Informationen aus diesem Abschnitt.

Weitere Operatoren können in Java auch selbst implementiert und in MiningMart eingebunden werden. Euler und Scholz haben dies ausführlich dokumentiert [18, 19, 54].

6.3.1. Primitive Operatoren

Euler führt genau drei primitive Operatoren an [22].

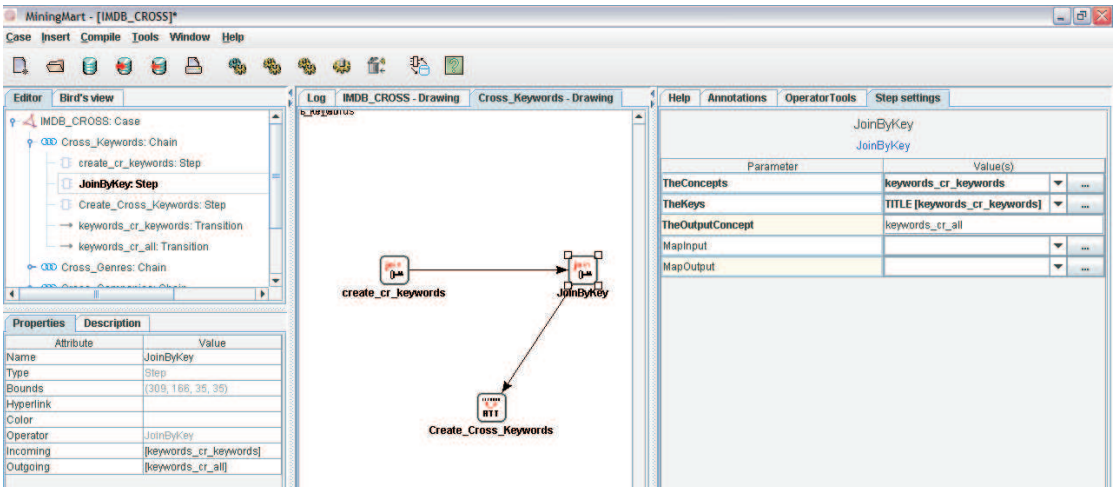


Abbildung 6.6.: Der Fall-Editor von MiningMart: Im linken Bereich des Bildschirms können in einer Baumansicht einzelne Elemente einer Kette über ihren Namen selektiert werden. Dazu können ihre Eigenschaften über die Tabelle links unten bearbeitet werden. Rechts hingegen werden die Parameter des gerade ausgewählten Operators angezeigt. Einige der Paramter sind fett hervorgehoben, was bedeutet, dass dort unbedingt Eingaben durch den Benutzer zu tätigen sind, um den Operator ausführen zu können. In der Mitte sieht man den eigentlichen Designer, der das Modellieren der Ketten erlaubt. Operatoren können beliebig angeordnet und über Transitionen miteinander verbunden werden.

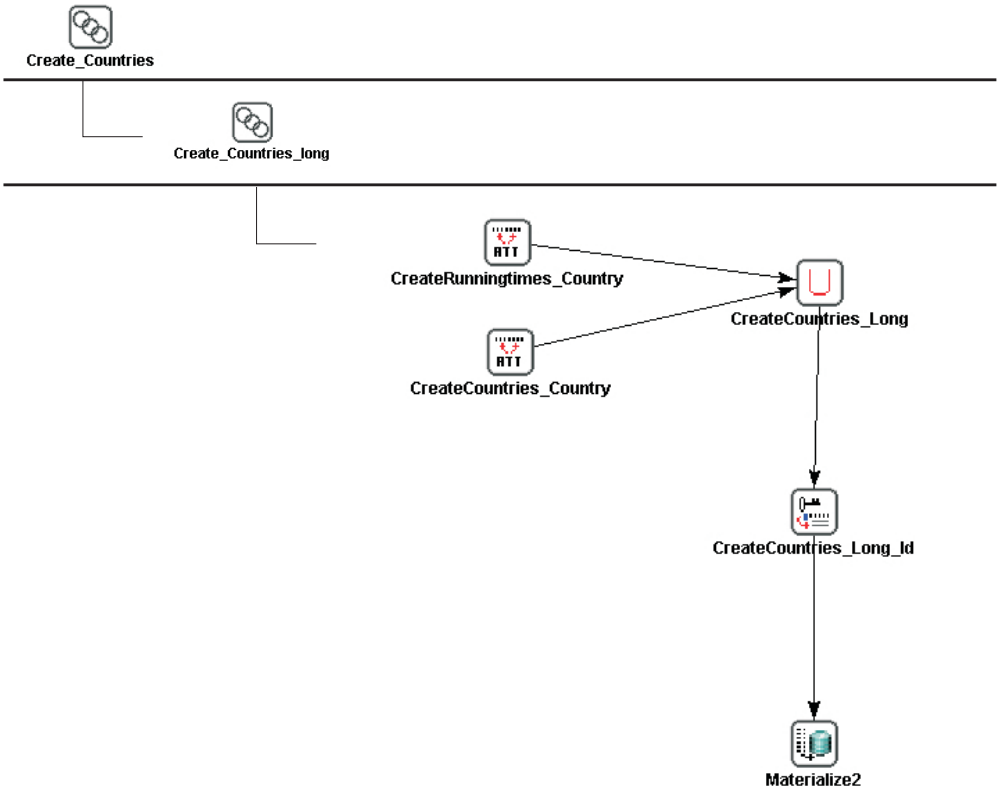


Abbildung 6.7.: Struktur von Ketten im Fall-Editor: Ein Fall kann beliebig viele Ketten enthalten. Jede Kette kann dann wiederum beliebig viele Unterketten beinhalten. Die Abbildung zeigt beispielhaft die Kette 'Create_Countries' und die untergeordnete Kette 'Create_Countries_long'.

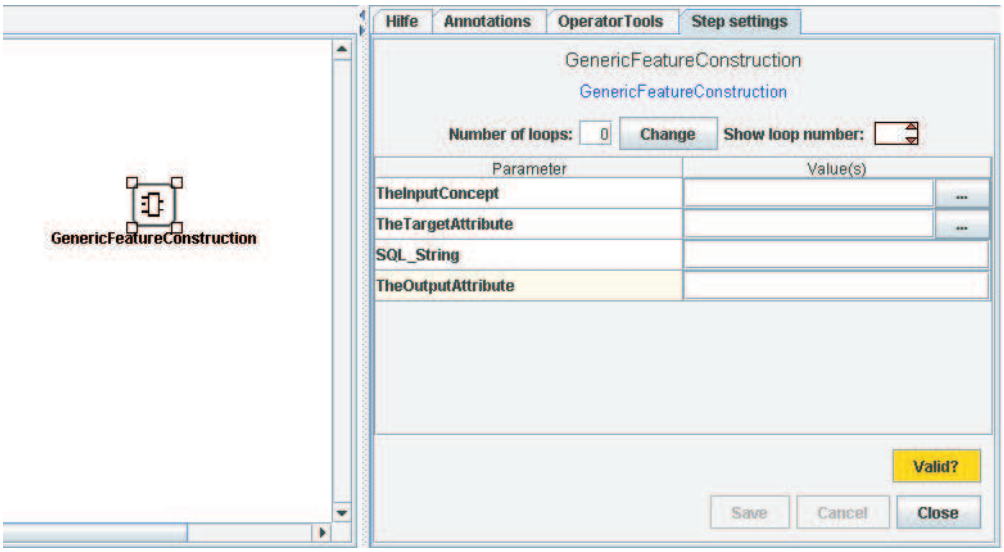


Abbildung 6.8.: Wichtige Aspekte zu Operatoren: Operatoren führen Operationen auf einem oder mehreren Konzepten durch. MiningMart bietet eine Vielzahl von Operatoren mit zusammengesetzter Funktionalität, so dass Ketten gut strukturiert und schnell verständlich sind. Diese Abbildung zeigt beispielhaft den Operator namens 'GenericFeatureConstruction', der in einem Eingabekonzept ein weiteres Attribut erzeugt (vgl. Abschnitt 6.3.1)

Abbildung 6.9.: Beispiel für eine Kette im Fall-Editor; die Kette generiert die Entitätsklasse 'Keywords' (vgl. Abschnitt 7.4). Zunächst wird genau ein Attribut selektiert. Im nächsten Schritt wird die ID als neues Attribut abgeleitet. Diese beiden Operatoren werden im Detail in Abschnitt 6.3 erläutert. Die Materialisierung ist ein spezieller Operator in MiningMart, der ein Konzept als Tabelle in einer Datenbank auf der relationalen Ebene realisiert. Dabei werden grundsätzlich alle Attribute und Tupel übernommen.

Attribut ableiten (Attribute derivation)

Dieser Operator erzeugt ein neues Attribut in einem Konzept. Dieses neue Attribut basiert meist auf einem bereits vorhandenen Attribut, welches mittels einer Funktion für das neue Attribut z.B. aufbereitet oder umgeformt wird. Dies ist jedoch keine Voraussetzung.

Um Missverständnisse zu vermeiden sei hier betont, dass es in MiningMart keinen Operator gibt, der genau diese Funktionalität besitzt.

Es gibt einen Operator namens 'Generic Feature Construction', der jedoch lediglich auf Tupelebene arbeitet und somit nicht die Anforderungen einer Attributableitung erfüllen kann. MiningMart erstellt im selben Konzept, auf welches der Operator angewendet wird, ein weiteres Attribut. Weiterhin geschieht die eigentliche Generierung des Attributwertes durch die Angabe von SQL-Funktionen. Die bereits aus dem letzen Abschnitt bekannte Abbildung 6.8 zeigt den Operator mit allen Parametern. Trotz der formalen Einschränkungen eignet sich der in MiningMart zur Verfügung stehende Operator sehr gut, um zum Beispiel eine Datenbereinigung bzw. Datenaufbereitung von einzelnen Attributen durchzuführen.

Attribut selektieren (Attribute selection)

Mit der Selektion von Attributen wird die Anzahl der Attribute in dem Eingabekonzept geringer. Der Operator besitzt als Parameter eine Liste von Attributen, die aus dem Konzept auszuwählen sind. Diese Liste wird meist vom Benutzer vorgegeben, kann jedoch auch automatisch erfolgen, z.B. das Ergebnis eines Mining-Algorithmus. Es ist auch denkbar, ein Kriterium anzugeben, welche Attribute auszuwählen sind.

MiningMart wählt dabei den ersteren Weg. Es ist eine Liste der zu selektierenden Attribute anzugeben. Dabei wird ein neues Konzept erstellt, welches nur noch die gewählten Attribute enthält. Zu finden ist der Operator unter der Bezeichnung 'FeatureSelectionByAttributes'.

Verbund über Schlüsselattribut (Join by key)

Dieser Operator verknüpft zwei oder mehrere Eingabekonzepte unter Berücksichtigung von Schlüsselattributen. Es können nur Konzepte mit gleicher Anzahl Schlüsselattribute verknüpft werden, d.h. jedes Schlüsselattribut muss in jedem Konzept vorhanden sein. Das Ausgabekonzept enthält alle Attribute der Eingabekonzepte inklusive der Schlüsselattribute, wobei diese nur einmal generiert werden. Der Verbund wurde bereits in Abschnitt 2.1.3 formal beschrieben.

MiningMart setzt genau dieses um. Der Operator namens 'JoinByKey' erwartet mindestens zwei Eingabekonzepte, zu denen dann die Schlüsselattribute anzugeben sind. Als Ergebnis wird ein Ausgabekonzept wie oben beschrieben erzeugt.

6.3.2. Zusammengesetzte Operatoren

Die folgenden Operatoren wurden für häufig auftretende KDD-Fälle entworfen. Sie ermöglichen eine einfachere und präzisere Modellierung und Beschreibung. Es ist jedoch hervorzuheben, dass sie alle redundant sind, da sie durch Kombinationen der primitiven Operatoren ersetzt werden können. [22] enthält noch Erläuterungen zu weiteren

zusammengesetzten Operatoren, wobei sich diese Arbeit auf die wichtigsten Operatoren beschränkt.

Selektion von Tupeln (Row selection)

Alle Attribute des Eingabekonzeptes werden in das Ausgabekonzept übernommen. Im Gegensatz zur Selektion von Attributen werden hier die Tupel mittels einer Bedingung gefiltert. Nur Tupel, die die Bedingung erfüllen, werden in das Ausgabekonzept übernommen.

MiningMart bietet einige Operatoren an, die eine Tupel-Selektion nach verschiedenen Kriterien durchführen. Als besonders hilfreich hat sich in der Anwendung der Operator mit der Bezeichnung 'RandomSampling' erwiesen, der zufällig eine festgelegte Anzahl von Tupeln auswählt. Beim Data Mining auf sehr großen Datenbanken können so in angemessener Zeit erste Lernergebnisse ermittelt werden, bevor man den gesamten Datenbestand benutzt.

Vereinigung (Union)

Zwei oder mehr Konzepte mit den gleichen Attributen werden zu einem Konzept vereinigt. Dabei enthält das Ausgabekonzept die Tupel aller Eingabekonzepte. In MiningMart besteht die Möglichkeit anzugeben, dass das Konzept keinerlei doppelte Einträge enthalten soll. Der Operator ist unter der Bezeichung 'Union' zu finden.

Aggregation

Der Operator aggregiert Werte über bestimmte Gruppierungsattribute (vgl. 'Group by' in SQL, Abschnitt 2.1.5) des Eingabekonzeptes. Das Ausgabekonzept enthält dann ein Tupel für jede Kombination der Gruppierungsattribute. Dieser Operator ist vor allem für den Einsatz bei der ersten Datenanalyse hilfreich, da man so z.B. einen Eindruck über die Verteilung der einzelnen Attribute erhält. Die wichtigsten Funktionen zur Aggregation sind Maximum, Minimum, Durchschnitt, Median, Summe und Anzahl.

Materialisierung

Der Operator 'Materialize' ist ein spezieller Operator in MiningMart. Er erstellt eine Tabelle auf der relationalen Datenbankebene, die eine Kopie des Konzeptes in MiningMart darstellt. MiningMart erzeugt häufig nur Sichten[7]. Der Zugriff auf sie ist jedoch nicht optimal, da die Ergebnismenge teilweise nicht im Speicher für schnellen Zugriff abgelegt werden kann. Es werden alle Attribute und Tupel übernommen.

Gleichzeitig wird dieses materialisierte Konzept auch innerhalb von MiningMart unter einem neuen Namen geführt, den der Anwender vorgeben kann. Über diesen Operator wird somit eine Schnittstelle zu der Welt außerhalb von MiningMart zur Verfügung gestellt. Beispielsweise kann eine Anwendung auf diese Weise Ergebnisse aus MiningMart übernehmen und der Zielgruppe präsentieren, die keinerlei Interesse an den Data

[7] engl.: views

Bezeichnung in Literatur (z.B. [22])	Bezeichnung in MiningMart	Anzahl Konzepte Eingabe	Anzahl Konzepte Ausgabe
Attribut selektieren	FeatureSelectionByAttribute	1	1
Attribut ableiten	zum Teil: GenericFeatureConstruction	1	0
Verbund über Schlüsselattribut	JoinByKey	min. 2	1
Selektion von Tupeln	RowSelectionByRandomSampling	1	1
Vereinigung	Union	min. 2	1
Aggregation	UserDefinedGrouping	1	0
Materialisierung	Materialize	1	1

Tabelle 6.1.: Übersicht über alle in Abschnitt 6.3 erläuterten Operatoren. Die Tabelle stellt die formalen Begriffe denen von MiningMart gegenüber und gibt Auskunft darüber, wie die Operatoren mit Konzepten verbunden werden können.

Mining-Schritten hat. Es werden jedoch weder Primärschlüsselattribute noch Fremdschlüsselbeziehungen zu anderen Konzepten auf die Relationen übertragen, die eventuell im Konzepteditor von MiningMart existieren.

Abbildung 6.9 zeigt ein Beispiel für den Einsatz einer Materialisierung. Die Kette erstellt ein Konzept - eine Entitätsklasse - die an weiteren Stellen im Modell benutzt werden soll. Daher erfolgt eine Materialisierung.

Materialisierung mit Primärschlüssel

Der Operator zur Materialisierung überträgt die im Konzepteditor von MiningMart als Primärschlüssel deklarierten Attribute bei einer Materialisierung nicht auf die relationale Ebene.

Während der Bearbeitung des Fallbeispiels im Rahmen dieser Arbeit stellte sich aber heraus, dass ein Operator, der die Primärschlüsselattribute auf der relationalen Ebene abbilden kann, sehr hilfreich wäre, um die Eingabe für den Lernschritt direkt aus MiningMart übernehmen zu können.

Somit ist nun in MiningMart der Operator 'MaterializeWithPKs' verfügbar, der auch Primärschlüsselattribute in den Tabellen auf relationaler Ebene erstellt.

Materialisierung von Relationen

Neben der Materialisierung von Konzepten mit Primärschlüsseln auf der relationalen Ebene ist es auch interessant, Beziehungen zwischen zwei oder mehr Konzepten auf die relationale Ebene zu übertragen.

So werden zum Beispiel $m : n$-Beziehungen über Kreuztabellen realisiert, die Fremdschlüsselbeziehungen zu den materialiserten Konzepten besitzen. Durch den Operator 'CreateManyToManyRelation' ist genau dies möglich. Es werden zwei Konzepte spezifiziert, zwischen denen eine Beziehung besteht. Weiterhin ist ein drittes Konzept anzugeben, welches die Kreuztabelle bildet. MiningMart erstellt nun zum einen die Beziehung zwischen den beiden Konzepten im Konzepteditor (vergleiche Abbildung 6.11), zum anderen werden alle drei Konzepte auf der relationalen Ebene materialisiert. Dabei wird

die Kreuztabelle mit entsprechenden Fremdschlüsselbeziehungen versehen. Die beiden anderen Tabellen erhalten einen Primärschlüssel, um dieses zu ermöglichen.

Dieser Operator enthält somit implizit den Operator 'MaterializeWithPKs' für die beiden verknüpften Konzepte. Wurde ein verknüpftes Konzept bereits materialisiert, so erkennt der Operator dies und fügt nur die notwendigen Fremdschlüsselbeziehungen in der Kreuztabelle ein.

Ein Ergebnis einer Materialisierung mit Berücksichtigung der Beziehungen ist in Abbildung 6.10 auf der nächsten Seite zu sehen. Die Abbildung wurde mit Hilfe der Datenbankverwaltungsbenutzeroberfläche von Oracle XE erstellt, die in der Lage ist, Fremdschlüsselbeziehungen auf diese Art zu visualisieren. Die Tabelle

RDT_CROSS_KEYWORDS

ist die Kreuztabelle, die über Fremdschlüssel auf die Tabellen

MM_TITLES und MM_KEYWORDS

verweist.

Erstere Tabelle enthält in diesem Fall eine Liste von Filmtiteln, die zweite eine Liste aller auftretenden Schlüsselwörter. Die Kreuztabelle beinhaltet nun ausschließlich Informationen, um Filmtitel mit einem oder mehreren Schlüsselwörtern zu verknüpfen. Filme ohne zugeordnete Schlüsselwörter treten somit in der Liste der Filme auf, jedoch nicht in der Kreuztabelle.

MiningMart besitzt einen weiteren Operator dieser Art namens 'CreateOneToMany-Relation'. Dieser Operator funktioniert vom Prinzip her genau wie 'CreateManyToManyRelation', jedoch für $1 : m$-Beziehungen.

6.4. Compiler

Der Compiler setzt eine Kette von Operatoren in eine Datenbankanweisung bzw. Datenbankanfrage um. Eine Kette wird also von vorne nach hinten abgearbeitet, so dass die Anfrage des letzten Operators einer Kette auf allen zuvor generierten Anweisungen aufsetzt. In der Regel generiert der Compiler lediglich eine SQL-Anweisung, die dann erst ausgeführt wird, wenn der Benutzer z.B. im Konzepteditor ein Konzept, welches durch eine Kette erstellt wurde, einsieht oder wenn ein Konzept auf der relationalen Ebene materialisert wird.

Der Compiler von MiningMart bietet vielseitige Möglichkeiten, einzelne Ketten zu bearbeiten und ist auch insofern effizient implementiert, als nur bei erkannten Änderungen an - unter Umständen abhängigen - Operatoren eine Compilierung erneut durchgeführt wird.

Die vom Compiler erzeugten SQL-Anweisungen eines Falles können - als Abfolge betrachtet - als ein generiertes Data Mining-Programm angesehen werden. Außerdem wird der Benutzer während des Vorgangs der Compilierung mit ausführlichen Informationen über den Fortschritt informiert (vergleiche Abschnitt C.1 auf Seite 137).

6.5. Charakterisierung

Abschließend sollen noch wichtige Aspekte von MiningMart dargestellt werden.

MiningMart kann durch die folgenden 5 Aspekte charakterisiert werden [43]:

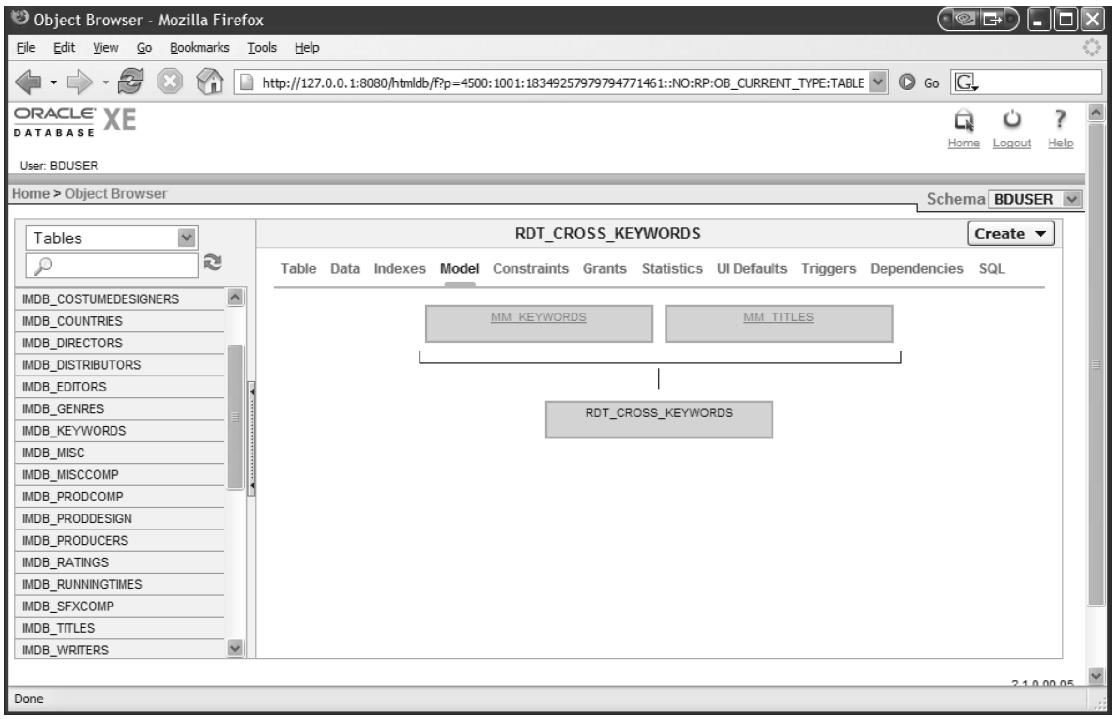


Abbildung 6.10.: Materialisierte Relation auf relationaler Ebene, dargestellt vom Oracle Datenbankverwaltungswerkzeug: Der Operator 'CreateManyToMany-Relation' hat eine Kreuztabelle unter Berücksichtigung der Beziehungen zu den verknüpften Konzepten materialisert. In dieser Abbildung ist das Modell für die Schlüsselwörter von Filmen zu sehen. Jedem Film (MM_TITLES) können ein oder mehrere Schlüsselwörter (MM_KEYWORDS) zugeordnet werden. Diese Verknüpfung wird über die Kreuztabelle RDT_CROSS_KEYWORDS auf relationaler Ebene realisiert. Auf der konzeptuellen Ebene gibt es sowohl eine Beziehung zum Konzept mit den Filmtiteln als auch eine Beziehung zu dem Konzept, das alle möglichen Schlüsselwörter beinhaltet (vergleiche Abbildung 6.11).

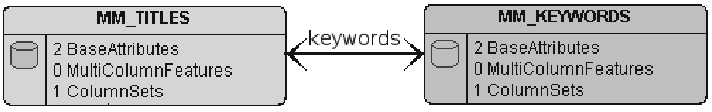


Abbildung 6.11.: Die Abbildung zeigt eine durch den Operator 'CreateManyToManyRelation' erstellte Relation namens 'keyword'. Der Operator bildet diese Beziehung auf der relationalen Ebene durch Erstellung einer Kreuztabelle ab (vergleiche Abbildung 6.10).

große Datenmengen MiningMart ist ein datenbankorientiertes System. Dadurch ist ein Zugriff auf alle SQL-fähigen Datenbanksysteme möglich. So ist zum Beispiel eine direkte Anwendung in Datenbanken aus der Wirtschaft problemlos möglich. Weiterhin sind viele Operatoren für den Zugriff auf große Datenmengen optimiert worden.

anspruchsvolle Datenvorverarbeitung Nicht nur kann der eigentliche Lernschritt im Data Mining Lernverfahren einsetzen, sondern die Datenvorverarbeitung kann auch davon profitieren. So können zum Beispiel mit MiningMart fehlende Werte für Attribute durch Werte ersetzt werden, die zuvor durch einen Lerner gelernt wurden. Eine Attributselektion und Attributgenerierung bereits während der Datenvorverarbeitung kann die Qualität der Daten, die als Eingabe für das Data Mining-Verfahren dienen, enorm verbessern.

Meta-Daten MiningMart nutzt Meta-Daten über Operatoren und die eigentlichen Businessdaten im Compiler, um eine dynamische KDD-Anwendung zu generieren.

Dokumentation Alle Vorgänge innerhalb der Fälle von MiningMart sind transparent und können an jeder Stelle mit Kommentaren versehen werden. Weiterhin können alle Elemente mit eigenen Bezeichnern versehen werden. Die Kommentare können dazu direkt aus dem Fall für Berichte verwendet werden. Die klare Trennung zwischen den Konzepten und den Ketten zur Bearbeitung der Daten trägt weiterhin zur Übersicht bei.

Flexibilität Bei allen Fällen handelt es sich um Meta-Modelle. Somit können Lösungen einfach auf andere Datenbanken übertragen und dort angewendet werden. Lediglich das Datenmodell des Falles ist auszutauschen. Zudem kann die Feinabstimmung über die grafische Benutzeroberfläche leicht durchgeführt werden.

7. Fallbeispiel: Durchführung der Datenvorbereitung und Modellierung der Internet Movie Database mit MiningMart

Das Fallbeispiel dieser Arbeit wurde systematisch am CRISP-DM Modell ausgerichtet, welches in Kapitel 4 auf Seite 31 ausführlich vorgestellt wurde. Die Aufgabe besteht darin, den Datenbestand der Internet Movie Database für eine Modellierung vorzubereiten und anschließend eine Ontologie (vergleiche Kapitel 3 auf Seite 19) mit MiningMart (vergleiche Kapitel 6 auf Seite 44) zu erstellen.

Die Internet Movie Database[1] enthält eine enorme Menge an Informationen zu Kinofilmen, TV Filmen und (Mini-)Serien. Zu jedem dieser Bestandteile können, neben Schauspielern und anderen beteiligte Personen und Firmen, auch zahlreiche weitere Details aus der Welt des Films abgerufen werden.

Jeder Benutzer darf zusätzliche Informationen im Datenbestand erfassen, d.h. es wird auf das Prinzip der Partizipation der Benutzer gesetzt, welches im Zusammenhang mit dem semantischen Web in Abschnitt 3.7 auf Seite 30 bereits vorgestellt wurde. Diese werden zwar von einem Komitee validiert, jedoch ist selbst dadurch eine einheitliche Auflistung der Informationen nicht immer gegeben, welches ein wichtiger Aspekt der Datenvorverarbeitung ist.

Abschließend soll gezeigt werden, dass die Anwendung eines relationalen Regellernsystems auf dieses Modell möglich ist. Verwendet wird dabei das Regellernsystem RDT/DM, welches bereits in Abschnitt 5.3 auf Seite 37 ausführlich vorgestellt wurde.

Im Vordergrund steht dabei die Transparenz des Prozesses, das heißt der Prozess soll zu jedem Zeitpunkt einsehbar sein. Weiterhin ist eine Modellierung gefordert, die Erweiterungen bzw. Modifikationen auf einfache Art und Weise zulässt. Daher wurde zur Modellierung das bereits behandelte Werkzeug MiningMart verwendet, welches eine komfortable Benutzeroberfläche zur Durchführung der Wissensentdeckung in Datenbanken bietet.

Die genauen Fragestellungen und Ansätze werden in Schritt 1 in Abschnitt 7.1 diskutiert.

Weiterhin kann MiningMart die Lerneingabe direkt zur Verfügung stellen, da sowohl MiningMart als auch RDT/DM auf der gleichen Datenbank arbeiten können. Es wurde in Kapitel 6 auf Seite 44 im Detail vorgestellt.

Eine Verbesserung der Lernergebnisse gegenüber zuvor auf dem Datensatz der Internet Movie Database durchgeführten Regellernaufgaben steht im Hintergrund. Vielmehr soll gezeigt werden, dass die Erstellung einer Ontologie auch die Lösung von komplexen Lernaufgaben vereinfacht und die Formulierung von verschiedenen Lernaufgaben beschleunigt, indem auch komplexe Lernziele, bedingt durch eine klare und verständliche Struktur, einfach ausgedrückt und die Eingaben für den Lernprozess schnell zur Verfügung gestellt werden können.

[1] URL: `http://www.imdb.com`

Abschnitt 7.1 behandelt den ersten Schritt des CRISP-DM Modells (vergleiche Kapitel 4 auf Seite 31), das Verstehen des Sachbereichs. Dabei wird insbesondere in Unterabschnitt 7.1.3 auf die Bezugsquellen der zu bearbeitenden Daten der Internet Movie Database eingegangen.

Schritt 2 in Abschnitt 7.2 geht dann näher auf den Aufbau der einzelnen Bestandteile der Rohdaten ein. Das Ergebnis der Datenanalyse wird kurz aufgelistet. Technische Aspekte werden dabei nicht aufgeführt. Die Bearbeitung eines konkreten Beispiels, geleitet durch das CRISP-DM Modell, steht im Vordergrund.

In Abschnitt 7.3 werden dann die konkreten Schritte genannt, die zur Lösung der Problemstellung durchgeführt werden. Sie bilden die Basis für Abschnitt 7.4, der den Kern dieses Kapitels bildet. Dort sind genaue Beschreibungen der für diese Arbeit in MiningMart erstellten Modelle bzw. Ketten zu finden.

Wie im KDD-Prozeß vorgesehen, erfolgt dann der Schritt der Evaluierung in Abschnitt 7.5.

Abschließend wird die Anwendungsphase beschrieben, wobei dieser Schritt in Kapitel 8 auf Seite 98 ausführlich erläutert wird.

Dateiname	in Modell	Wissen zu
actors.list	x	2
actresses.list	x	2
aka-names.list		2
aka-titles.list		1
alternate-versions.list		1
biographies.list		2
business.list	x	1
certificates.list		1
cinematographers.list	x	2
color-info.list		1
composers.list	x	2
costume-designers.list	x	2
countries.list	x	1
crazy-credits.list		1
directors.list	x	2
distributors.list	x	1
editors.list	x	2
genres.list	x	1
goofs.list		1
keywords.list	x	1
language.list		1
laserdisc.list		1
literature.list		1
locations.list		1
miscellaneous-companies.list	x	1
miscellanoeus.list	x	2
movie-links.list		1
movies.list	x	1
mpaa-ratings-reasons.list		1
plot.list		1
producers.list	x	2
production-companies.list	x	1
production-designers.list	x	2
quotes.list		1
ratings.list	x	1
release-dates.list		1
running-times.list	x	1
sound-mix.list		1
soundtracks.list		1
special-effects-companies.list	x	1
taglines.list		1
technical.list		1
trivia.list		1
writers.list	x	2
# Dateien 44	22	

Legende:	
1	Angaben zum Film
2	Angaben zu Person

Abbildung 7.1.: Liste aller Dateien, die von `IMDB.com` zur Verfügung gestellt werden. Diese Dateien werden in dieser Arbeit als Rohdaten bezeichnet. Es wurden nur die Dateien in das Modell übernommen, die mit einem 'x' in der zweiten Spalte gekennzeichnet sind. Für die Modellierung ist es außerdem wichtig zu erkennen, welche der Dateien Informationen zu einem Film und welche Datei Informationen zu den in der Branche beteiligten Personen liefert.

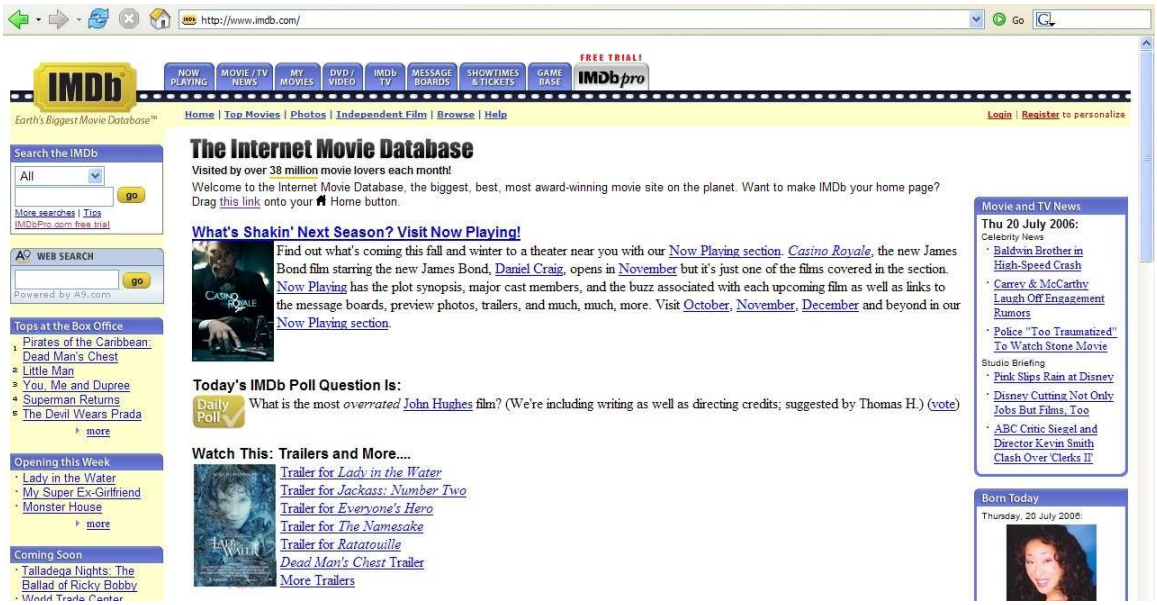


Abbildung 7.2.: Begrüßungsseite der Internet Movie Database (20.07.2006)

7.1. Schritt 1: Verstehen des Sachbereiches

Bevor man sich mit den Daten der Internet Movie Database auseinandersetzen kann, muss ein Verständnis dafür gewonnen werden, welche Informationen von den Anbietern überhaupt zur Verfügung gestellt werden. Weiterhin ist die Zielgruppe der Internetseite festzustellen.

7.1.1. Vorgehen

Zum Kennenlernen des Sachbereichs wurde zunächst die Internetseite der Internet Movie Database genau eingesehen.

Besucht man die Internetseite `http://www.imdb.com` (vergleiche Abbildung 7.2), so wird man mittlerweile von Informationen geradezu überflutet. Die zusammengestellten Informationen stammen jedoch nahezu vollständig aus der Internet Movie Database. Ausnahmen sind Werbung zur Finanzierung der Seite und Hinweise auf andere Projekte, die mit der Internet Movie Database im Zusammenhang stehen.

Diese Arbeit betrachtet den Teil der Datenbank, der für nicht-kommerzielle Zwecke kostenlos zugänglich ist. Es gibt mittlerweile eine weitere, kostenpflichtige Datenbank, die sich an Kunden richtet, die in der Filmindustrie arbeiten bzw. die Informationen professionell nutzen möchten.

Als Information zu der Datenbank geben die Verantwortlichen an, dass sie es sich zum Ziel gesetzt haben, jedes Detail zu einem Film erfassen zu wollen. Dazu sollen die Informationen übersichtlich auf der Seite präsentiert und einfach zu durchsuchen sein.

Eine genaue Übersicht, wieviele Informationen bereits erfasst wurden, kann ebenfalls abgerufen werden[2].

Bei der genauen Betrachtung der Internetseiten fallen zwei verschiedene 'Seitentypen' besonders ins Auge: Es steht entweder eine Person oder ein Film im Vordergrund. Bei

[2] URL:`http://www.imdb.com/database_statistics`

Abbildung 7.3.: Beispiel für die Darstellung der Informationen zu einem Schauspieler (20.07.2006)

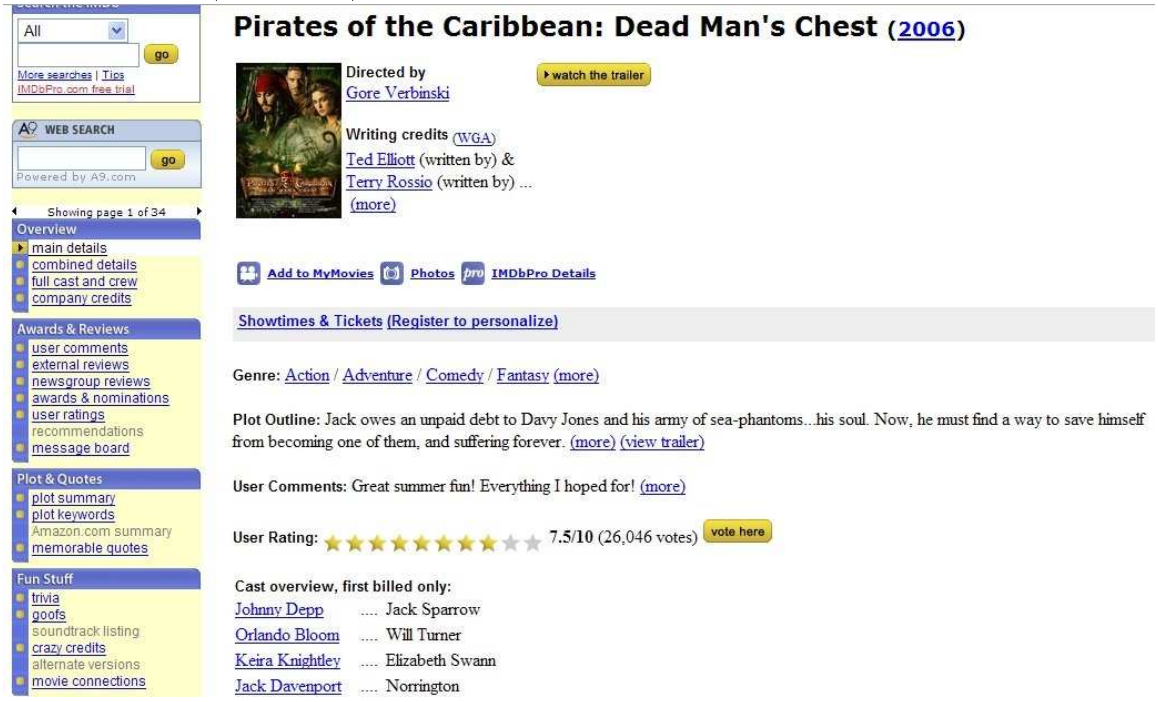


Abbildung 7.4.: Beispiel für die Darstellung der Informationen zu einem Kinofilm (20.07.2006)

Betrachtung einer Person wird auch auf deren Aktivitäten in der Filmindustrie verwiesen. Umgekehrt können die beteiligten Personen eines Filmprojekts ermittelt werden, wenn man einen Film im Detail einsieht. Abbildungen 7.4 und 7.3 zeigen diese beiden Seiten exemplarisch.

Die Informationen können über die Navigation auf verschiedene Arten zusammengestellt werden. Es fällt jedoch auf, dass keine spezielle Zielgruppe identifiziert werden kann. Die Internet Movie Database richtet sich an jeden, der an Informationen zu Filmen und Personen aus der Filmwelt interessiert ist.

Es ist anzumerken, dass in dieser Arbeit zur Vereinfachung von 'Filmen' gesprochen wird, obwohl auch zum Beispiel TV-Serien in der Datenbank erfasst werden. Die Arbeit betrachtet den gesamten Datenbestand der erfassten 'Titel', also Filme, Serien usw. ebenfalls.

7.1.2. Schlussfolgerungen

Durch die genaue Betrachtung kann der Sachbereich konkret abgegrenzt werden. Da die Internet Movie Database keine besonders eingeschränkte Zielgruppe hat, muss jedoch trotzdem bedacht werden, dass nicht alle Benutzer den gleichen Informationsbedarf haben. Die Navigation auf der Internetseite lässt zwar zu, bestimmte Themenbereiche auszuwählen, jedoch werden die Informationen nie für bestimmte Benutzergruppen aufbereitet. Dies soll als erste Anforderung für die Datenaufbereitung festgehalten werden.

Anforderung 1. *Informationen der Internet Movie Database sind nicht für bestimmte Benutzergruppen aufbereitet. Der KDD-Prozess soll die Möglichkeit schaffen, einfach und flexibel Informationen für bestimmte Benutzergruppen zur Verfügung zu stellen.*

Beispiel 7.1. *Mögliche Fragestellungen für verschiedene Benutzergruppen könnten lauten:*

- PRODUZENT*: Was charakterisiert europäische Klassenschlager?*
- REGISSEUR*: Welche Beleuchter sind auf der ganzen Welt einsetzbar?*
- JOURNALIST*: Welche Filme werden von Frauen hoch, von Männern niedrig bewertet?*
- FILMKRITIKER*: Welche Schauspieler treten nur in schlecht bewerteten Filmen auf?*

Weitere Aspekte müssen in diesem Schritt nach dem CRISP-DM Modell für diese Arbeit nicht untersucht werden, da sie nur von kommerzieller bzw. finanzplanungstechnischer Bedeutung sind.

Der nächste Abschnitt erläutert den Prozess des Zusammenstellens der Ausgangsdaten.

7.1.3. Zusammenstellen der Daten

Die Datenbankinformationen der Internet Movie Database liegen in vielen Textdateien vor, die im Internet über das File Transfer Protocol heruntergeladen werden können[3].

[3] URL: `http://www.imdb.com/interfaces#plain`

Eine andere Fassung der Daten, wie zum Beispiel bereits in ein Datenbanksystem eingelesen, ist nicht erhältlich. Ein Bezug von anderen Quellen, die die Daten unter Umständen bereits in Datenbanken eingelesen haben, wird rechtlich ausdrücklich untersagt. Daher bilden diese Textdateien die Basis dieses Fallbeispiels.

Abbildung 7.1 auf Seite 63 gibt einen Überblick über alle Textdateien, die heruntergeladen werden können. Weiterhin sind die Textdateien markiert, die in dieser Arbeit bearbeitet werden. Diese Textdateien werden im Folgenden immer mit dem Begriff 'Rohdaten' bezeichnet.

7.2. Schritt 2: Datenverständnis

Der nächste Schritt des Data Mining nach dem CRISP-DM Modell sieht vor, die notwendigen Daten zusammenzustellen, diese zu beschreiben, zu explorieren und dann die Qualität zu beurteilen.

7.2.1. Vorgehen

Es ist einfach, Zugriff auf die Daten der Internet Movie Database zu erhalten. Diese Dateien zu verstehen ist jedoch kompliziert, da es eine vollständige Dokumentation dieser Dateien, die Aufschluss über den Aufbau bzw. den genauen Inhalt der einzelnen Dateien geben würde, nicht gibt. Es gibt lediglich einige Kommentare zu Beginn oder am Ende jeder Datei, die die in dieser Datei abgelegten Hauptinformationen nennen.

Aus diesem Grund bildet die Beschreibung der vorliegenden Daten den Hauptteil des folgenden Abschnitts. Die durchgeführte Datenexploration wird ebenfalls mit einigen interessanten Beispielen belegt. Eine Beurteilung der Qualität der Daten ergibt sich bereits durch die Beschreibung der Daten nahezu von selbst, wird aber trotzdem am Ende des Abschnitts gegeben.

7.2.2. Schlussfolgerungen

Da es keinerlei Dokumentation zu den einzelnen Datendateien gibt, hilft nur der Ansatz, jede Datei von Hand mit einem Texteditor genau zu betrachten.

Auch in den Dateien selbst gibt es keinerlei Beschreibung, was genau in ihnen abgelegt ist.

So wird zum Beispiel in der Datei 'actresses.list' nur gesagt, dass es sich um die Liste der Schauspielerinnen handelt. Keinerlei Zusatzinformationen werden gemacht.

Eine genaue Spezifizierung aller möglichen Ausprägungen einer Information bzw. eine genaue Beschreibung der Attribute fehlt jedoch.

Die einzige Gemeinsamkeit, die alle Dateien haben, ist die erste Zeile. Dort ist ein CRC-Code zu finden, der zur Überprüfung dient, ob die Datei keinerlei Übertragungsfehler enthält. Außerdem sind der Name der Datei und das Erstellungsdatum aufgeführt.

Beispiel 7.2. *Die Datei 'actresses.list', die für die Experimente in dieser Arbeit verwendet wurde, hat den folgenden Kopf:*

```
CRC: 0x303E8B09  File: actresses.list  Date: Fri Oct 28 01:00:00 2005
```

Eine eindeutige Identifikation und Bestimmung des Erstellungsdatums ist somit für alle Datendateien gegeben.

Weitere Angaben zum Inhalt und zur Struktur der Datei werden nicht gemacht. Der gesamte Dateikopf ist im Anhang in Abschnitt A.1 auf Seite 116 abgebildet. Auch der Abspann der Datei ist im darauf folgenden Abschnitt A.2 auf Seite 119 zu finden.

Nach der ersten Inspektion der Rohdaten wird bereits klar, dass eine direkte Übernahme in ein Datenbanksystem, um dort eine Datenanalyse und anschließend die Datenvorverarbeitung vorzunehmen, unmöglich ist.

Die Hauptgründe dafür sind:

1. keine klare, durchgehende Trennung von
   - Attributen und
   - verschiedenen Datensätzen

   Die Zuordnung von Informationen auf eine Person bzw. einen Film ist oftmals schwer.
2. keine vorgegebene Ausprägung der Attribute, d.h. Längenbeschränkung oder Wertebereich
3. kein einheitlicher Aufbau der einzelnen Dateien mit den Rohdaten
4. problematische Erkennung von Beginn und Ende der Daten
5. Sonderzeichen, die unter Umständen nicht in eine Datenbank übernommen werden können
6. Änderungen am Aufbau der Rohdaten

Diese Aussagen sollen kurz belegt werden. Die entsprechenden Inhalte der Rohdaten sind dabei im Anhang in einer längeren Fassung abgedruckt, so dass man sich einen umfassenderen Eindruck verschaffen kann. In diesem Abschnitt werden lediglich die Hauptaspekte dargestellt und erläutert. Die eigentliche Problematik wird aber meist erst dann klar, wenn man mehrere Zeilen aus den Rohdaten betrachtet.

Zu Punkt 1 können zunächst zwei verschiedene Typen von Datendateien identifiziert werden. Die meisten Datendateien (vergleiche Abschnitt A.4 auf Seite 121) trennen Attribute durch

- wiederholtes Leerzeichen ($\sqcup\sqcup$),
- Tabulatorzeichen oder
- Klammerung.

Leider ist die Anzahl der Trennzeichen nicht immer gleich, so dass die Zuordnung der Attribute erschwert wird. Oftmals enthalten die Daten auch zwei verschiedene Trennungsmethoden in einer Zeile.

Beispiel 7.3 (mehrere, verschiedene Trennzeichen in einer Zeile)**.** *Personendaten enthalten häufig in einer Zeile mehrere Trennzeichen bzw. Arten der Trennung. So sind in der Liste der Kameraleute zum Beispiel folgende zwei Zeilen zu finden:*

```
1 Aabech,␣Bjørk    ⇥    ⇥Berøringer␣(1972)␣(TV)
2    ⇥    ⇥    ⇥Under␣uret␣(1985)␣(TV)␣␣(as␣Bjørk␣Aabeck)
```

Hier ist zunächst der Name des Kameramanns von den anderen Informationen durch Tabulator(en) getrennt und dazu der Filmtitel durch zwei Leerzeichen (␣␣) von weiteren Zusatzinformationen. Weiterhin ist dies ein Beispiel für Zuordnungsprobleme. Einem Menschen ist direkt klar, dass Zeile 2 *ebenfalls der Person aus Zeile* 1 *zuzuordnen ist, eine direkte Übernahme in ein Datenbanksystem ist aufgrund dieses Aufbaus jedoch nicht möglich. Bei den Personenangaben wird auch der Wunsch einer Dokumentation von Beschränkungen für die Anzahl der möglichen Attribute bzw. der Angabe eines Wertebereichs deutlich (vergleiche Punkt 2). Diese sind jedoch nicht vorhanden und daher wurde in dieser Arbeit der Weg gewählt, Informationen in den Dateien mit der Darstellung im Web zu vergleichen, um so auf alle Möglichkeiten zu schließen. Dazu wurden alle Dateien mit Hilfe von Hilfsprogrammen daraufhin untersucht, ob die gemachten Annahmen auf alle Daten zutreffen. Dies war ein zeitintensiver Prozess, der jedoch in dieser Arbeit nicht Schritt für Schritt aufgeführt werden kann.*

Dazu gibt es die Datendatei 'ratings.list', die feste Längen für die einzelnen Attribute vorgibt und daher auf ein Trennzeichen verzichtet (vergleiche Abschnitt A.3 auf Seite 121). Somit ist Punkt 3 bewiesen, da sich der Aufbau der vorgestellten Dateien grundsätzlich unterscheidet.

Die Erkennung des eigentlichen Beginns der Daten (Punkt 4), d.h. die Trennung zwischen Daten und einleitenden bzw. abschließenden Informationen, ist ebenfalls ein Problem. Bei Betrachtung aller Dateien fallen Kennungen wie

```
1 THE␣CINEMATOGRAPHERS␣LIST
2 ========================
3
4 Name    ⇥    ⇥    ⇥Titles
5 ----    ⇥    ⇥    ⇥------
```

oder

```
1
2 MOVIES␣LIST
3 ===========
4
```

auf. Diese Kennungen wurden einmalig für die Datenübernahme für die einzelnen Dateien ermittelt und zur Datenübernahme codiert.

Dies ist keine optimale Lösung, da sich der Aufbau der Dateien (Punkt 6) regelmäßig ändert. Die für die Implementierungen im Rahmen dieser Arbeit verwendeten Kennungen sind aber über den Zeitraum der Erstellung dieser Arbeit nicht verändert worden.

Die etwaigen Änderungen an dem Format sind jedoch eine Konsequenz aus dem Wunsch, möglichst alle Informationen erfassen zu wollen, und auch daraus, dass mehrere Personen den Datenbestand pflegen.

Die Erkennung des Endes der Daten ist ebenso schwierig. Es sei hier auf den Anhang verwiesen, der in Abschnitt A.2 auf Seite 119 ein vollständiges Beispiel anführt für den Übergang zwischen den Daten und dem Abspann.

Ein weiterer Aspekt bei der Bearbeitung der Rohdaten sind die verwendeten Sonderzeichen (Punkt 5). Die Internet Movie Database führt zum Beispiel alle Filmtitel möglichst in der Sprache auf, in der sie zum ersten Mal veröffentlicht wurden. Die Übernahme dieser Sonderzeichen hat sich bei der Bearbeitung des Fallbeispiels als problematisch herausgestellt, so dass eine Normierung vorgenommen wurde. Die Normierung ist so angelegt, dass eine Zurückführung auf den ursprünglichen Titel möglich ist.

Der Wissensentdeckungsprozess muss somit alle diese Punkte berücksichtigen und robust gestaltet werden.

Anforderung 2. *Die Rohdaten müssen in eine einheitliche Form gebracht werden, die von einem Datenbanksystem eingelesen werden kann.*

In der Praxis werden hierzu in der Regel sogenannte komma-separierte Textdateien verwendet. Bei diesen Dateien werden die Attribute durch ein individuell festgelegtes Zeichen - meist der Tabulator - getrennt und die Tupel durch Zeilenumbrüche getrennt. Häufig ist die erste Zeile für die Namen der Attribute reserviert.

Durch diesen Schritt ist die Nutzung nicht auf ein spezielles Datenbanksystem eingeschränkt, da nahezu alle Datenbanksysteme eine Funktion zum Einlesen von Dateien dieser Form bieten.

Anforderung 3 (Umwandlung der Rohdaten)**.** *Die Rohdaten sind in eine komma-separierte Form zu bringen. Dazu sind Normierungen am Zeichensatz vorzunehmen, um Probleme beim Einlesen in ein Datenbanksystem zu verhindern.*

Trotzdem muss dem Datenbanksystem mitgeteilt werden, wo welche Informationen in den komma-separierten Dateien zu finden sind.

Anforderung 4 (Einlesen in die Datenbank)**.** *Die Übernahme der komma-separierten Dateien in das Datenbanksystem ist durchzuführen. Dazu müssen dem Datenbanksystem Meta-Informationen zu den komma-separierten Dateien zur Verfügung gestellt werden.*

Erst dann kann die eigentliche Datenvorverarbeitung mit Hilfe von MiningMart beginnen.

Anforderung 5 (Datenvorverarbeitung mit MiningMart)**.** *Die Tabellen der Datenbank, die zuvor eingelesen wurden, müssen als Konzepte in MiningMart zur Verfügung gestellt werden, um die Modellierung der Ketten zu ermöglichen.*

Die Ketten, die im nächsten Abschnitt vorgestellt werden, stellen die Informationen auf der konzeptuellen Ebene zur Verfügung. Für den Lernprozess über RDT/DM ist es erforderlich, dass die Daten wieder als Tabellen in einer Datenbank verfügbar sind.

Anforderung 6 (Generieren der Lerneingabe)**.** *Die Datenvorverarbeitungsketten in MiningMart müssen die erstellten aufbereiteten Konzepte auf die relationale Ebene des Datenbanksystems abbilden. Wie bei der Erläuterung der Materialisierung bereits angesprochen, sind Beziehungen unter den Konzepten ebenfalls abzubilden.*

So kann das einzusetzende Lernverfahren RDT/DM nicht nur auf die aufbereiteten Informationen zugreifen, sondern auch die Beziehungen der Konzepte untereinander als zusätzliche Information für den Lernvorgang nutzen.

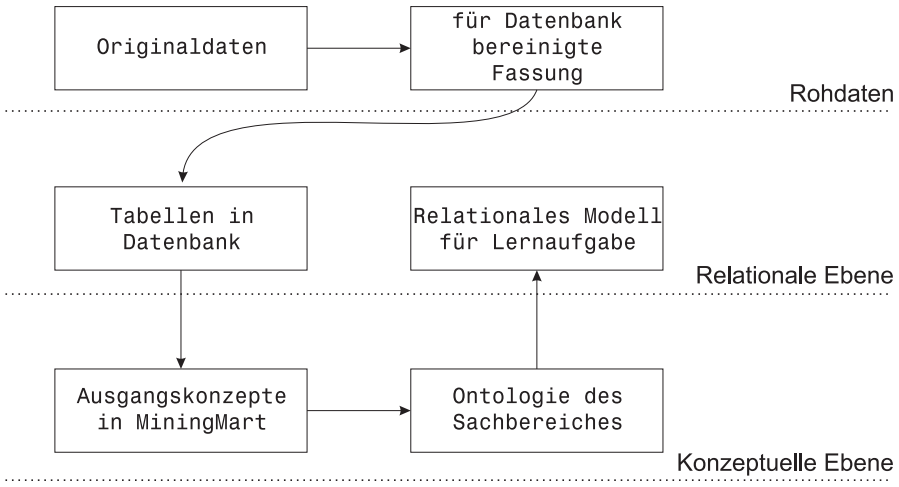


Abbildung 7.5.: Schema zur Beschreibung des genauen Ablaufs der durchgeführten Datenmodellierung unter Berücksichtigung der Ebene der Rohdaten, der relativen Datenbankstrukturen und der konzeptuellen Ebene. Abbildung 7.6 auf der nächsten Seite zeigt anhand der Schauspielerdaten diese Schritte im Detail.

Zu allen Anforderungen sind zwei Abbildungen in dieser Arbeit zu sehen, die den Ablauf grafisch aufzeigen. Abbildung 7.5 auf dieser Seite zeigt den Ablauf in Bezug auf die unterschiedlichen Abstraktionsebenen, d.h. die Ebene der Rohdaten, der relationalen Ebene und der konzeptuellen Ebene.

Die verschiedenen Repräsentationen einer Datendatei von der Rohdatenform bis hin zu der Bildung von Entitätsklassen sind in Abbildung 7.6 auf der nächsten Seite dargestellt.

Für Anforderung 3 ist eine Windows-Anwendung (vgl. Abbildung 7.7 auf Seite 73) geschrieben worden, die die Rohdaten in komma-separierte Form bringt. Diese Anwendung enthält Umformungsschritte für jede der in Abbildung 7.1 gekennzeichneten Dateien.

Als Datenbanksystem wird Oracle XE[4] verwendet.

Anforderung 4 wurde daher mit dem Programm 'SqlLoader' umgesetzt, welches in jeder Oracle-Version enthalten ist. Für jede komma-separierte Datei ist eine Datei mit Meta-Informationen zu erstellen, die den Aufbau der komma-separierten Datei beschreibt. Diese Dateien sind im Anhang im Abschnitt B.1.2 auf Seite 126 vollständig abgedruckt.

Eine Exploration der Daten wird in der Regel durchgeführt, um die Problemstellungen des durchzuführenden Data Minings genauer abzugrenzen. Dazu werden zahlreiche Anfragen, Visualisierungen und Berichte über die Daten erstellt. Ergebnisse dieser Exploration fließen dann in die Beschreibung der vorliegenden Daten und vor allem in die Einschätzung zur Qualität der Daten ein.

Bevor die Modellierung der Datenvorverarbeitungsketten mit MiningMart begonnen wird, werden nun einige interessante Aspekte der Datenexploration vorgestellt. Vor allem soll gezeigt werden, mit welchen Mitteln diese durchgeführt wurde.

Die Daten der Internet Movie Database liegen nun in einem Datenbanksystem vor, so

[4]Zum Zeitpunkt der Verfassung dieser Diplomarbeit ist die Version mit der Bezeichnung 'Oracle Database 10g Express Edition Release 10.2.0.1.0 - Beta' für Studenten frei zugänglich gewesen und wurde daher für die Datenvorverarbeitung mit MiningMart verwendet.

Abbildung 7.6.: Ablauf der Umwandlung der Schauspielerdaten. Zunächst liegen die Daten in dem Format vor, in dem sie im Internet heruntergeladen werden können. Der erste Umwandlungsschritt bringt die Daten in eine Form, die in eine Oracle-Datenbank eingelesen werden kann. Die Tabelle wird dann als Konzept in MiningMart eingebunden. Im letzten Schritt werden die Konzepte aufbereitet und so die Entitätsklassen gebildet. Abbildung 7.5 auf der vorherigen Seite zeigt den verallgemeinerten Ablauf.

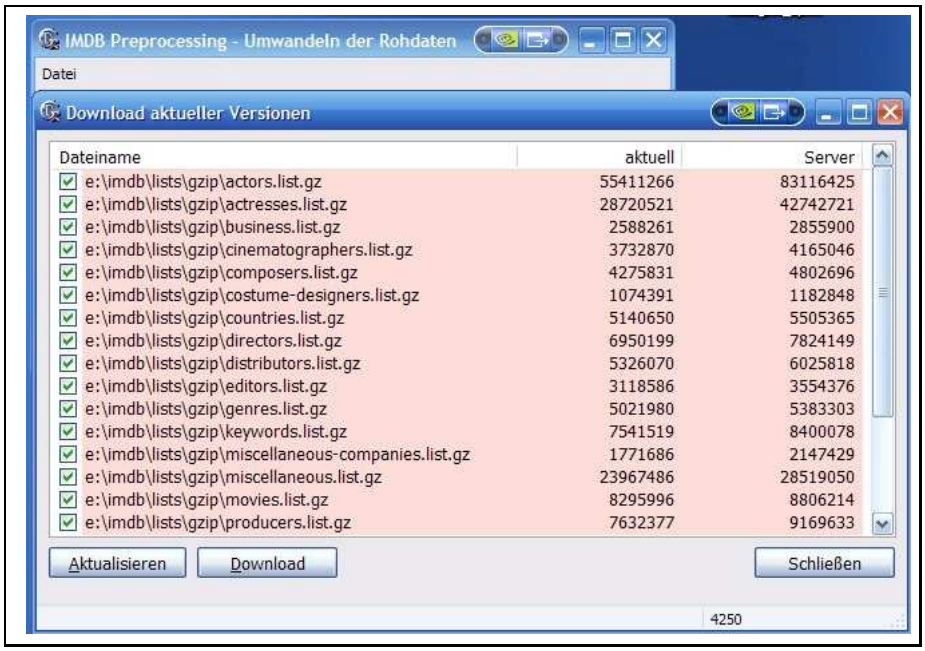


Abbildung 7.7.: Windows-Software zum Umformen der Rohdaten in komma-separierte Form. Die Rohdaten werden automatisch aus dem Internet heruntergeladen und anschließend umgewandelt. Vor dem Herunterladen wird sichergestellt, dass nicht bereits eine aktuelle Version lokal vorhanden ist.

dass SQL-Anfragen formuliert werden können. Somit können bereits triviale Fragen auf diesem Wege beantwortet werden, wie zum Beispiel:

- Welche Schauspieler spielen in Film x mit?
- Welche Bewertung haben alle Filme mit Schauspieler y?
- Welche Kameraleute haben mit Schauspieler y gearbeitet?

Bei allen diesen Fragestellungen handelt es sich um einfache SQL-Anweisungen. So enthält zum Beispiel die Tabelle mit den Schauspielern ein Attribut, welches den Filmtitel enthält. Alle Schauspieler auszugeben ist somit eine Selektion mit anschließender Projektion.

Die zweite Frage ist zwar schon ein wenig anspruchsvoller, aber auch die Tabelle mit den Bewertungen hat ein Attribut, welches den Filmtitel enthält. Man selektiert somit alle Filmtitel eines Schauspielers und selektiert genau diese Filmtitel in der Tabelle mit den Bewertungen.

Anfragen dieser Art sind leicht durchzuführen, wenn man die Struktur der Tabellen kennt, jedoch besitzen die Tabellen bisher keinerlei Beziehungen zueinander. Die Beziehungen sind durch den Benutzer manuell herzustellen.

Die Vollständigkeit der Daten wird als Qualitätsaspekt noch in diesem Abschnitt angesprochen werden. Bei einer Datenbank, die sich mit Filmen beschäftigt, drängt sich somit unmittelbar die Frage auf, wieviele Filme erfasst bzw. wieviele Filme in einem bestimmten Jahr erfasst wurden. Abbildung 7.8 auf der nächsten Seite zeigt die Anzahl der

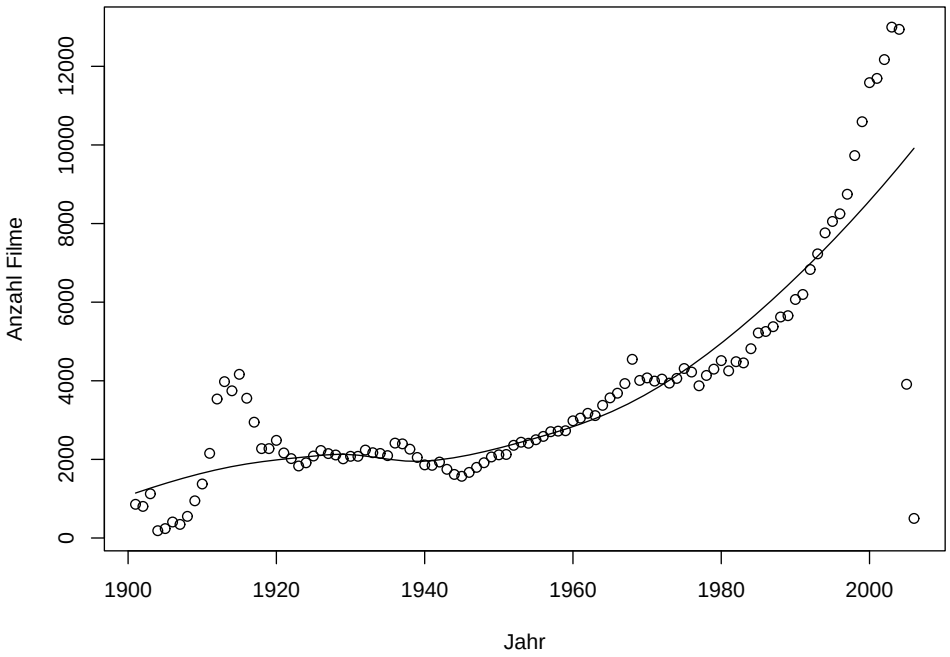


Abbildung 7.8.: Der Graph zeigt die Anzahl der Filme, die zu einem bestimmten Jahr in der Internet Movie Database eingetragen wurden. Die eingezeichnete Kurve ist eine Interpolation nach der Methode der Kleinsten Quadrate.

Filme, die zu einem bestimmten Jahr in der Datenbank eingetragen sind. Zunächst wird man sagen, die Datenbank vernachlässige Filme aus den Jahren vor circa 1970, aber man muss hier die Entwicklung der Filmindustrie, die hier nicht betrachtet werden soll, als Beurteilungskriterium heranziehen. Vor 1970 wurden bedeutend weniger Filme produziert, da das Fernsehen, Video und andere Medien noch gar nicht als 'Markt' vorhanden waren. Interessant ist jedoch die Entwicklung vor dem Ersten Weltkrieg. Jedes Jahr wurden mehr Filme produziert, was nach dem Ersten Weltkrieg bis ins Jahr 1960 nie wieder so deutlich eingetreten ist. Erst danach kann man wieder von einem Trend zu mehr Filmen pro Jahr sprechen. Die nach der Methode der Kleinsten Quadrate eingezeichnete Kurve zeigt diesen Trend ebenfalls auf.

Die Daten scheinen also für alle Filmjahre ordnungsgemäß erfasst worden zu sein, da sich geschichtliche und wirtschaftliche Aspekte anhand dieses Graphen nachvollziehen lassen. Sollte es sich um ein kommerzielles Data Mining Projekt handeln, so sollte man sicherlich Statistikinstitute einbeziehen, die Produktionszahlen der einzelnen Jahre kennen und diese mit den Jahreszahlen der Datenbank vergleichen. Eine visuelle Analyse dieser Art ist jedoch schon sehr aussagekräftig.

Ein weiterer interessanter Aspekt der in der Internet Movie Database erfassten Daten, besonders im Hinblick darauf, einen Film im Lernprozess als gut oder schlecht einzuordnen, sind die Bewertungen der Filme. Grundsätzlich ist festzuhalten, dass nicht zu jedem

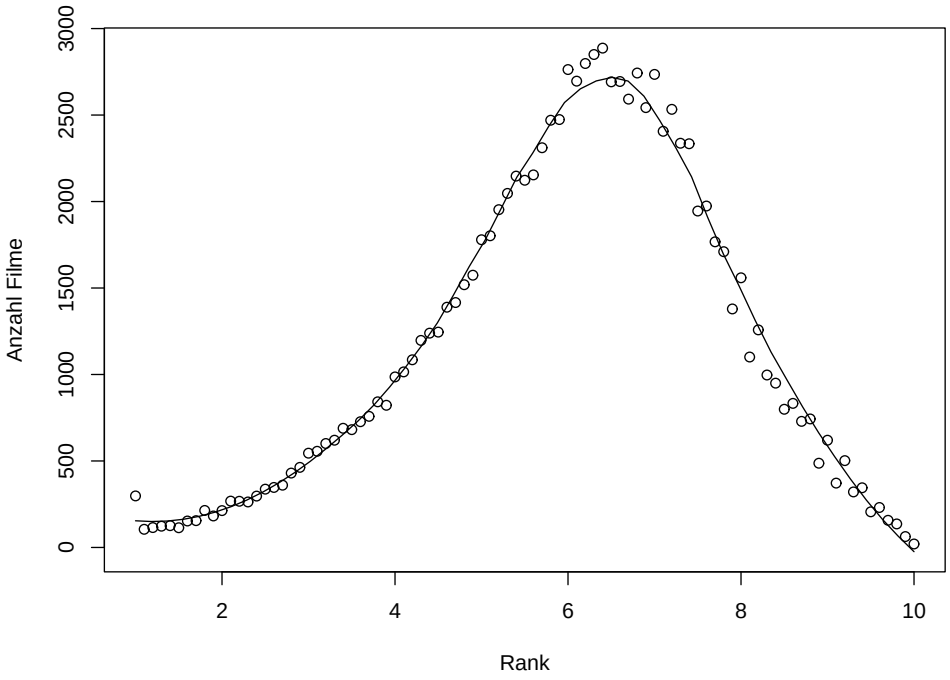


Abbildung 7.9.: Der Graph zeigt die Anzahl der Filme (y) zu einem Ranking (x). Das Ranking ist ein Wert zwischen 0 und 10, wobei 10 als die Bestbewertung für einen Film festgelegt ist. Es ist klar zu sehen, dass die meisten Filme ein Ranking zwischen 5 und 8 aufweisen. Bei der eingezeichneten Kurve handelt es sich um eine Interpolation nach der Methode der Kleinsten Quadrate.

Film eine Bewertung existiert. In den für diese Arbeit verwendeten Daten sind 15% aller Filme bewertet worden[5]. Benutzer der Datenbank haben im Internet die Möglichkeit neben der Verfassung einer ausführlichen, schriftlichen Bewertung, dem Film eine Gesamtnote zwischen 0 und 10 zu geben. 10 stellt dabei die Bestnote dar. In den Rohdaten ist darauf basierend für jeden Film der Durchschnittswert dieser Bewertungen und die Anzahl der abgegebenen Stimmen zu finden. Zusätzlich ist die Verteilung der Stimmen in Prozent gegeben.

Es ist nun vor der Modellierung interessant, wie diese Durchschnittswerte einzuschätzen sind. Aus den Werten einer SQL-Anfrage wurde daher der Graph in Abbildung 7.9 auf dieser Seite erstellt. Dazu wurde nach der Methode der Kleinsten Quadrate eine Kurve interpoliert. Der Graph zeigt, dass es nur wenige ganz schlechte und ebenso auch nur wenige sehr gute Filme gibt. Die Häufung der Werte liegt im Bereich $5 \leq rank \leq 8$. Durch diesen Graphen können Lernaufgaben zum Beispiel auf Filme von bestimmter Qualität - setzt man die Bewertungen mit Qualität gleich - beschränkt werden.

Als letzter Aspekt der Datenexploration sollen die Schlüsselwörter, die einem Film

[5] Absolute Werte: 112.943 von 749.674 Filmen sind bewertet

	Genre	# Filmen zugeordnet		Genre	# Filmen zugeordnet
1.	Short	109.558	15.	Music	8.874
2.	Drama	95.158	16.	Musical	8.307
3.	Comedy	76.900	17.	Fantasy	7.280
4.	Documentary	55.597	18.	Sci-Fi	6.930
5.	Adult	28.379	19.	Mystery	6.620
6.	Animation	21.672	20.	War	5.298
7.	Action	18.812	21.	Biography	3.388
8.	Romance	18.242	22.	Sport	2.723
9.	Family	16.639	23.	History	2.367
10.	Crime	16.058	24.	Game-Show	1.402
11.	Thriller	13.790	25.	Reality-TV	995
12.	Adventure	13.168	26.	News	915
13.	Horror	9.449	27.	Talk-Show	898
14.	Western	9.306	28.	Film-Noir	421

Tabelle 7.1.: Alle 28 Genres der Internet Movie Database. Die absolute Zahl gibt an, wie häufig ein Genre Filmen zugeordnet wurde.

zugeordnet werden, und die Einordnung der Filme in Genres betrachtet werden. Jedem Film können beliebig viele Schlüsselwörter und Genres zugeordnet werden. Insgesamt gibt es in der Internet Movie Database jedoch 33.207 verschiedene Schlüsselwörter. Davon werden allein 9.477 nur genau einem Film zugeordnet, was circa 29% entspricht.

Wie sinnvoll die große Vielzahl an Schlüsselwörtern ist, mag somit in Frage gestellt sein. Die Unterscheidungskriterien scheinen bei den Schlüsselwörtern meist viel zu sehr auf einzelne Aspekte eines Filmes bezogen zu sein[6]. Hinzu kommt, dass die Benutzer der Internet Movie Database schlichtweg überfordert werden, die Schlüsselwörter zu überblicken und sinnvoll zuzuordnen. Für Lernaufgaben sollten die Schlüsselwörter, die nur wenigen Filmen zugeordnet werden, nicht selektiert werden.

Bei den Genres ist die Verteilung und Anzahl jedoch vollkommen anders. Zum einen werden nur 28 verschiedene Genres in der Datenbank geführt. Dazu, wie man in Abbildung 7.10 erkennen kann, gibt es nur 4 Genres, die nicht häufig einem Film zugeordnet werden. Tabelle 7.1 auf dieser Seite zeigt die absoluten Werte für alle Genres.

Abschließend ist in diesem Abschnitt die Qualität der Daten auszuweisen. Die Schwierigkeiten, eine gezielte Datenvorverarbeitung beginnen zu können, sind ausführlich dargelegt worden. Nun sollen die folgenden Aspekte der Rohdaten betrachtet werden:

- Sind die Daten vollständig?
- Sind die Informationen korrekt?
- Sind die Daten ineinander schlüssig, d.h. sind z.B. Filmtitel über alle Datendateien konsistent, wird immer dieselbe Schreibweise verwendet?

Die ersten beiden Punkte sind bei öffentlichen Projekten nach der gewünschten Vorgehensweise des Semantischen Webs meist schwer einzuschätzen. Die Datenexploration

[6] Beispiele: yellow-gas, record-collecting, attitude-adjustment

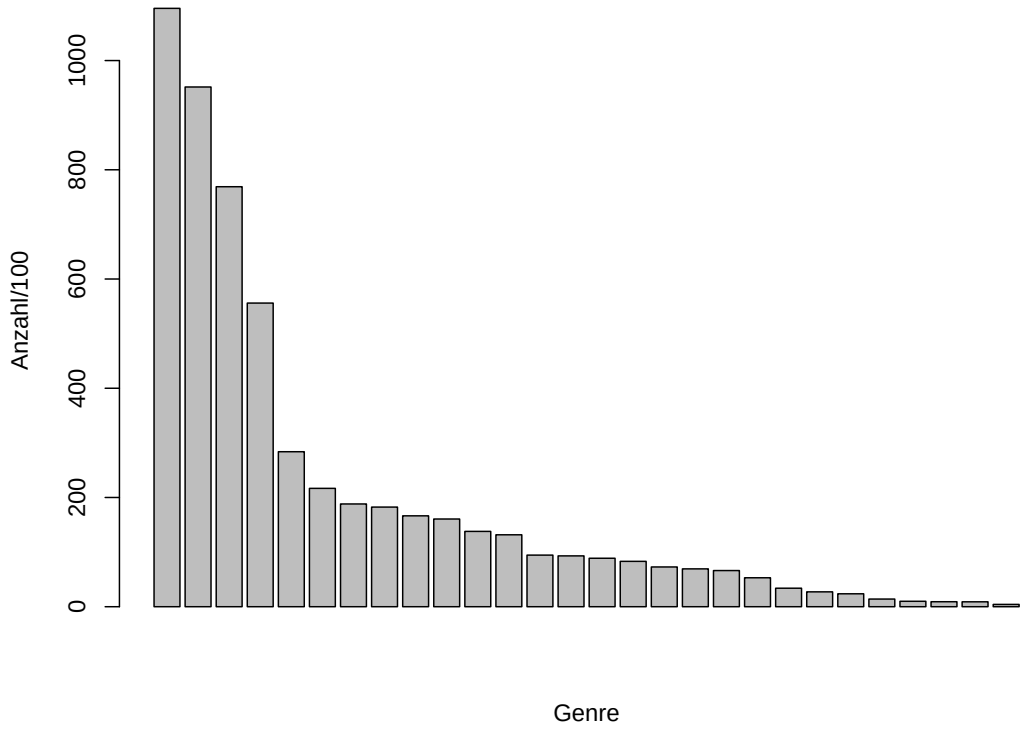


Abbildung 7.10.: Jedem der Balken im Graph ist eines der Genres aus der Internet Movie Database zugeordnet (zur Zeit wird zwischen 28 Genres unterschieden). Es ist zu erkennen, dass es nur wenige Genres gibt, die nur wenigen Filmen zugeordnet sind.

ging bereits näher auf die Vollständigkeit bzw. Verteilung der Daten ein. Es ist festzuhalten, dass die Korrektheit der Informationen von den Verantwortlichen der Internet Movie Database immer kontrolliert und mit höchster Sorgfalt in die Datenbank eingepflegt wird. Dennoch handelt es sich um Benutzereingaben, in die sich immer wieder Fehler einschleichen können. Es ist jedoch somit nicht garantiert, dass es zu jedem Film die gleiche Detailtiefe bei den Informationen gibt. So stellt man z.B. bei der Benutzung der Webseite schnell fest, dass Hollywood-Filme mit sehr vielen Details versehen sind, kleinere Produktionen jedoch gerade einmal mit dem Titel erwähnt werden.

Trotzdem sind die Daten konsistent, d.h. eine Person oder ein Film wird in allen Dateinamen immer auf die gleiche Art und Weise eingeführt. Die Verantwortlichen weisen auf der Internetseite auch ausdrücklich darauf hin, dass sie nur Daten einpflegen, die zugeordnet werden können. Weiterhin werden sämtliche Informationen immer auf die gleiche Art und Weise aufgeführt, die die einzelnen Daten miteinander verknüpfen.

Dies gilt jedoch nicht für Zusatzinformationen. So gibt es Datendateien, die Informationen einem Land zuordnen. Die Länderbezeichner weisen jedoch eine grosse Vielzahl von Schreibweisen auf. Es können weitere Beispiele für dieses Phänomen gefunden werden

(siehe zum Beispiel Tabelle 7.2 auf der nächsten Seite). Somit muss festgehalten werden, dass die Daten, die die Datendateien miteinander verknüpfen, sehr hoher Kontrolle unterliegen, jedoch die Ausprägungen der Zusatzinformationen nicht klar festgelegt und somit viele verschiedene Schreibweisen des gleichen Aspektes festzustellen sind.

Im letzten Absatz wurde bereits erwähnt, dass die verschiedenen Informationen über Zeichenketten, also Filmtiteln und Personennamen, miteinander verknüpft sind. Das heißt, es gibt eine Datendatei mit allen Informationen zum Filmtitel, wie z.B. Jahr der Produktion, dazu dann eine weitere Datendatei, in der genau dieser Filmtitel erneut aufgeführt wird. In dieser sind weitere Informationen erfasst, zum Beispiel ein Schlüsselwort. Nach dem Einlesen der Rohdaten in die Datenbank wurde daran nichts verändert, das heißt zum Beispiel in der Tabelle mit allen Schauspielerinformationen ('imdb_actors') gibt es mehrere Tupel (siehe Tabelle 7.2 auf der nächsten Seite), die immer wieder denselben Namen des Schauspielers enthalten.

In Abschnitt 2.1.8 auf Seite 16 wurden Normalformen eingeführt, durch deren Anwendung, der Normalisierung, so etwas vermieden wird. Nach den dort vorgestellten Aspekten würde man die Namen aller Schauspieler in eine eigene Tabelle verlagern und entsprechende Verknüpfungsschlüssel generieren. Die Tabelle mit den Informationen zum Schauspieler würde dann über diesen Schlüssel auf den Namen verweisen. Ein solcher Schlüssel ist dann meist eine Ganzzahl, die von Datenbanksystemen effizienter zu verwalten ist. Weiterhin wird so eine Änderung des Datenmodells auf der relationalen Ebene vorgenommen, die hier nicht betrachtet werden soll.

In dieser Arbeit soll vielmehr die Modellierung mit MiningMart genau diesen Normalisierungsprozess durch die Modellierung einer Ontologie in gewisser Weise nachempfinden. Dadurch sind Abhängigkeiten schneller und einfacher zu erkennen. Weiterhin sind Anfragen dann schneller und einfacher[7] durchzuführen.

[7] genauer: die SQL-Ausdrücke für die Anfragen werden kürzer und einfacher nachvollziehbar

imdb_actors	`SELECT name,title,role FROM imdb_actors WHERE (name='Hanks, Tom') AND [...]`	
name	**title**	**role**
Hanks, Tom	Apollo 13 (1995)	Jim Lovell
Hanks, Tom	Bachelor Party (1984)	Rick Gassko
Hanks, Tom	Big (1988)	Josh
Hanks, Tom	Bonfire of the Vanities, The (1990)	Sherman McCoy
Hanks, Tom	Cast Away (2000)	Chuck Noland
Hanks, Tom	Catch Me If You Can (2002)	Carl Hanratty
Hanks, Tom	Celluloid Closet, The (1995)	Himself
Hanks, Tom	Charlie Wilson's War (2006)	Charlie Wilson
Hanks, Tom	Cold Case, A (2006)	Andy Rosenzweig
Hanks, Tom	Da Vinci Code, The (2006)	Robert Langdon
Hanks, Tom	Dragnet (1987)	Pep Streebeck
Hanks, Tom	Elvis Has Left the Building (2004)	Mailbox Elvis
Hanks, Tom	Every Time We Say Goodbye (1986)	David
Hanks, Tom	Forrest Gump (1994)	Forrest Gump
Hanks, Tom	Great Buck Howard, The (2006)	
Hanks, Tom	Green Mile, The (1999)	Paul Edgecomb
Hanks, Tom	He Knows You're Alone (1980)	Elliot
Hanks, Tom	Joe Versus the Volcano (1990)	Joe Banks
Hanks, Tom	Ladykillers, The (2004)	Professor G.H. Dorr
Hanks, Tom	League of Their Own, A (1992)	Jimmy Dugan
Hanks, Tom	Man with One Red Shoe, The (1985)	Richard
Hanks, Tom	Money Pit, The (1986)	Walter Fielding, Jr.
Hanks, Tom	Nothing in Common (1986)	David Basner
Hanks, Tom	Philadelphia (1993)	Andrew Beckett
Hanks, Tom	Punchline (1988)	Steven Gold
Hanks, Tom	Radio Flyer (1992)	Older Mike
Hanks, Tom	Risk Pool, The (2006)	Sam Hall
Hanks, Tom	Road to Perdition (2002)	Michael Sullivan
Hanks, Tom	Saving Private Ryan (1998)	Captain John H. Miller
Hanks, Tom	Sleepless in Seattle (1993)	Sam Baldwin
Hanks, Tom	Splash (1984)	Allen Bauer
Hanks, Tom	Terminal, The (2004)	Viktor Navorski
Hanks, Tom	That Thing You Do! (1996)	Mr. White
Hanks, Tom	Toy Story (1995)	Woody
Hanks, Tom	Toy Story 2 (1999)	Sheriff Woody
Hanks, Tom	Turner & Hooch (1989)	Det. Scott Turner
Hanks, Tom	Volunteers (1985)	Lawrence Bourne III
Hanks, Tom	You've Got Mail (1998)	Joe Fox
Hanks, Tom	'burbs, The (1989)	Ray Peterson
Hanks, Tom	'Bosom Buddies' (1980)	Kip 'Buffy' Wilson
Hanks, Tom	'Celebrity Profile' (1999)	Himself

Tabelle 7.2.: Ausgewählte Einträge zu Tom Hanks. Es ist zu erkennen, dass jedes Tupel den Namen des Schauspielers erneut aufführt und sich auch gespielte Rollen wiederholen. Weiterhin ist diese Tabelle ein gutes Beispiel für unterschiedliche Auffassungen von zwei Benutzern der Internet Movie Database. So hat ein Benutzer bei 'Toy Story' die Rolle als 'Woody' bezeichnet, 4 Jahre später wurde für die gleiche Rolle 'Sheriff Woody' eingetragen.

7.3. Schritt 3: Datenvorbereitung

Die Datenvorbereitung beinhaltet die Auswahl der zu analysierenden Daten, die Datenbereinigung, die Merkmalsgenerierung und ein eventuelles Neuanordnen der Daten.

Alle diese Schritte werden in MiningMart mit Datenvorverarbeitungsketten realisiert, wobei man zu diesem Zeitpunkt bereits ein Modell entwickelt. Nach dem CRISP-DM Modell ist die Modellierung jedoch erst der nächste Schritt. Aus diesem Grund werden die Aspekte, die unter die Datenvorbereitung fallen, ebenfalls im folgenden Abschnitt über die durchgeführte Modellierung erläutert. Zusammenhänge sind so leichter verständlich. Weiterhin tritt somit auch der konzeptuelle Ansatz in den Vordergrund und es wird nicht mehr über Relationen in Datenbanken argumentiert, sondern vielmehr über Konzepte.

7.4. Schritt 4: Modellierung

Das Modell bildet den Kern des bearbeiteten Fallbeispiels. Mit Modell wird die Menge der Vorverarbeitungsketten in MiningMart bezeichnet.

Zur Modellierung ist in MiningMart ein Fall mit zahlreichen Ketten angelegt worden (siehe Kapitel 6 auf Seite 44), wobei eine hierarchische Anordnung dieser Ketten existiert (vergleiche Abbildung 6.7 auf Seite 53). Das heißt es gibt wenige Ketten auf der ersten Ebene, die dann jedoch viele Unterketten beinhalten.

Auf der ersten Ebene wird zwischen Datenvorverarbeitung, die Ketten gemäß Schritt 3 und 4 des CRISP-DM Modells enthält, und der Anwendungsphase unterschieden. Die Ketten für die Anwendungsphase generieren Lerneingaben für bestimmte Benutzergruppen, welches als Anforderung 6 auf Seite 70 in Schritt 1 festgehalten wurde. Diese Ketten können dann auch als Eingabe für eine grafische Benutzeroberfläche dienen, die die Daten für bestimmte Benutzergruppen (vergleiche Beispiel 7.1 auf Seite 66) aufbereitet und eventuell sogar grafisch darstellt.

Die Datenvorverarbeitung wird dabei dann nicht nach den einzelnen Aspekten des CRISP-DM Modells strukturiert, sondern vielmehr nach den bearbeiteten Konzepten und deren Eigenschaften.

Der nächste Abschnitt wird zunächst den Ansatz vermitteln und begründen, der bei der Modellierung gewählt wurde. Anschließend werden einzelne Ketten im Detail vorgestellt.

7.4.1. Ansatz

In Schritt 2 wurde eine ausführliche Datenexploration durchgeführt. Neben den hier bereits aufgeführten Erkenntnissen entwickelt man bei der Arbeit mit den Daten ein Verständnis dafür, welche die wichtigsten Aspekte bzw. Attribute der Datenmengen sind.

Bei der Analyse der Datenqualität wurde dazu auf Redundanzen innerhalb der Daten verwiesen und auf die fehlenden Beziehungen der Tabellen zueinander. Aus dieser Feststellung heraus ergab sich die Idee, zunächst die zentralen Dinge herauszuarbeiten, die in der Datenbank erfasst werden und auch nahezu in jeder Tabelle wiederzufinden sind. Das wohl offensichtlichste Attribut, das in jeder Tabelle zu finden ist, ist der Filmtitel.

Zusammengefasst sind die folgenden Dinge immer wieder zu finden:

- Filmtitel
- Personennamen
- Firmennamen.

Diese drei Dinge zeichnen sich besonders dadurch aus, dass sie auch die drei Grundpfeiler der Internet Movie Database bilden: Unabhängig davon was man abruft, handelt es sich immer um Informationen zu einem Film, einer Person oder einer beteiligten Firma. Dazu sind diese Informationen miteinander verknüpft. Das heißt zum Beispiel, Firmen werden Filmen und Personen zugeordnet.

Die Idee ist nun, diese drei Punkte als Entitätsklassen zu modellieren. Die abhängigen Konzepte verweisen dann auf diese Entitätsklassen.

Beispiel 7.4. *Gegeben sei ein Konzept, welches den Filmtitel und zusätzliche Informationen, wie zum Beispiel Produktionsjahr für diesen Film, enthält. Aus allen Filmtiteln wird nun eine Entitätsklasse gebildet, wobei jeder Filmtitel eindeutig identifiziert werden können muss. Das ursprüngliche Konzept führt den Filmtitel nicht mehr als Attribut auf, sondern erhält eine Beziehung zu der Entitätsklasse Filmtitel.*

Diese Idee wurde bei der hier beschriebenen Modellierung umgesetzt, wobei die Anzahl der Entitätsklassen noch erhöht wurde.

Insgesamt werden 8 Entitätsklassen gebildet:

1. Filmtitel
2. Personennamen
3. Namen für Figuren
4. Genres
5. Schlüsselwörter
6. Firmennamen
7. Länderbezeichner
    - vollständig
    - abgekürzt.

Für jede dieser 8 Entitätsklassen ist eine Kette zu modellieren, die die entsprechende Entitätsklasse bildet und sicherstellt, dass diese eindeutig zugeordnet werden können.

Es wurde überlegt, die abgekürzten Länderbezeichner auf vollständige Länderbezeichner abzubilden. Davon wurde jedoch abgesehen, da die kurzen Länderbezeichner konsequent im Zusammenhang mit Firmen verwendet werden, die vollständigen Länderbezeichner im Zusammenhang mit Filmen. Um diese Zusatzinformation nicht im grundlegenden Modellierungsansatz zu verlieren, wurde keine Angleichung der abgekürzten Ländernamen auf vollständige Bezeichner vorgenommen.

Abbildung 7.11.: Kette zur Erstellung der Entitätsklasse der Genres (analog zur Erstellung der Entitätsklasse der Schlüsselwörter)

Der nächste Schritt ist die Bildung der Beziehungen der Konzepte zueinander, das heißt in den Ausgangskonzepten sind die Attribute, die durch Entitätsklassen modelliert wurden, durch Beziehungen zu ersetzen. Hier kommt man nicht mehr mit 8 Ketten aus, da zu jedem einzelnen der Ausgangskonzepte ein neues, mit Beziehungen versehenes zu erstellen ist.

Nun wird auch klar, warum die Ketten nicht nach Vorverarbeitungsschritten strukturiert wurden.

Es ist daher im MiningMart Fall zum einen eine Kette mit allen Unterketten zur Erstellung der Entitätsklassen zu finden, zum anderen gibt es eine Kette mit Unterketten zur Erstellung der Beziehungen.

Es ist nun noch zu klären, wie genau die Entitätsklassen mit Hilfe von MiningMart erstellt werden. Anzumerken sei schon jetzt, dass die Ketten eine Art ausbaufähige Vorlagen, ganz im Sinne des Fallbasierten Schließens, darstellen, da sicherlich noch weitere Schritte zur Merkmalsgenerierung für andere Lernaufgaben oder Auswertungen anderer Art denkbar sind.

7.4.2. Erstellung der Entitätsklassen

Zunächst sind mit Hilfe von MiningMart die Konzepte zu erstellen, die die Entitätsklassen des Modells bilden. Anschließend können diese Konzepte mit den urprünglichen Daten verknüpft werden. Diese Verknüpfung wird zum einen auf konzeptueller Ebene, zum anderen auf relationaler Ebene Bestand haben.

Die folgenden Unterabschnitte werden die erstellten Ketten zur Bildung der Entitätsklassen kurz vorstellen. Technische Details, zum Beispiel die Belegung der Parameter der einzelnen Operatoren, werden nicht aufgeführt.

Es ist anzumerken, dass alle Ausgangskonzepte, die direkt auf Datendateien der Internet Movie Database basieren, immer den Namen der Datendatei mit dem Prefix 'imdb_' tragen.

Beispiel 7.5. *Der Datendatei 'keywords.list' entspricht in MiningMart das Konzept 'imdb_keywords'.*

Genres und Schlüsselwörter

Die Erstellung der Entitätsklassen für Genres und Schlüsselwörter ist analog, was am Ende dieses Abschnitts verdeutlicht wird. Daher wird nur die Kette für die Genres ausführlich vorgestellt.

Das Ausgangskonzept für die Genres (imdb_genres) enthält zwei Attribute:

1. title, ein Filmtitel
2. genre, ein Genre

Um die Entitätsklasse aller Genres zu erstellen, sind also im ersten Schritt alle Genres auszuwählen. Weiterhin soll jede Bezeichnung für ein Genre nur einmal in die Menge der Entitäten aufgenommen werden. Die Kette in Abbildung 7.11 auf der vorherigen Seite wendet daher als ersten Schritt einen 'FeatureSelectionByAttributes'-Operator auf das Ausgangskonzept an. In dem so erstellten Konzept, mit lediglich noch dem Genre als Attribut, wird ein neues Attribut, ein eindeutiger Bezeichner, generiert. Der Operator 'CreatePrimaryKey' führt genau dies aus und erstellt ein neues Konzept mit den zwei gewünschten Attributen. Dieses Konzept stellt bereits die Entitätsklasse dar. Der letzte Schritt in der Kette ist optional. Dort wird eine Materialisierung der Entitätsklasse auf die relationale Ebene vorgenommen.

Das Ausgangskonzept für die Schlüsselwörter (imdb_keywords) besitzt die Attribute

1. title, ein Filmtitel
2. keyword, ein Schlüsselwort

Es ist die gleiche Kette zu verwenden wie für die Genres, da lediglich eine Anpassung der Attribut- und zu erstellenden Konzeptnamen durchzuführen ist.

Dies ist ein erstes Beispiel für die Wiederverwendbarkeit von Ketten in MiningMart.

Personen und Figuren

Die nächsten Ketten, die vorgestellt werden, erstellen die Entitätsklassen für Personennamen und Namen für Figuren. Sicherlich ist diese Trennung nicht in allen Fällen sinnvoll, da Figuren in gewisser Weise auch häufig Personennamen tragen, jedoch werden bei Filmen die meisten Figuren nur mit Vornamen oder mit anderen Bezeichnungen wie zum Beispiel 'Mann in der Bar' angegeben.

Personennamen werden in der Internet Movie Database, genau wie die Filmtitel, über mehrere Datendateien konsistent gleich geschrieben, um eine Verknüpfung der Informationen zu erreichen. Eine Liste aller in der Datenbank gespeicherten Personen kann man also durch Selektion aller Personennamen über alle Konzepte, die Personennamen enthalten, bilden. Anschließend muss sichergestellt werden, dass jeder Personenname in dem neu erstellten Konzept nur einmal aufgeführt wird und jede Person einen eindeutigen Bezeichner erhält. Das Vorgehen ist also bisher identisch mit dem zuletzt vorgestellten für Schlüsselwörter und Genres.

Jedoch enthält die Internet Movie Database auch Informationen darüber, unter welchem Namen ein Schauspieler in einem bestimmten Film aufgeführt wird. So wird zum Beispiel der Schauspieler 'Andrew Robinson' in einem seiner ersten Filme als 'Andy Robinson' aufgeführt. Dies ist aber ebenso eine Personenbezeichnung, so dass auch diese alternativen Namensbezeichner (engl. also known as = aka) in die Entitätenmenge der Entitätsklasse Personen eingeordnet werden müssen.

Dies ist in MiningMart einfach möglich, würde aber bei einer manuellen SQL-Datenvorverarbeitung nur mit sehr vielen, unübersichtlichen Anweisungen zu erledigen sein.

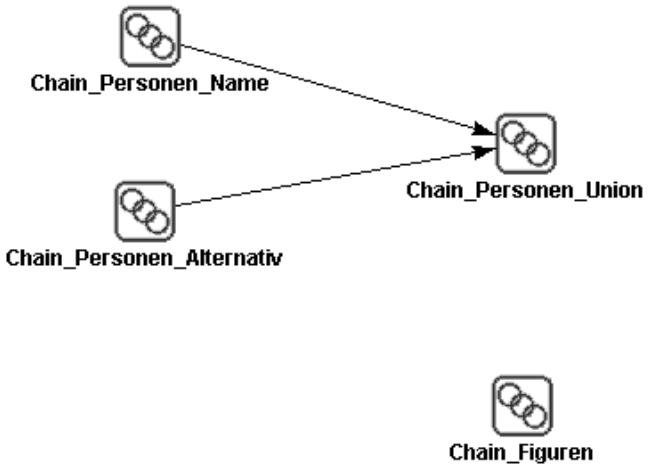


Abbildung 7.12.: Struktur der Ketten zur Erstellung der Entitätsklassen für Personen und Figuren. Es ist zu beachten, dass die Konzepte der beiden Ketten auf der linken Seite über Transitionen der Kette auf der rechten Seite zur Verfügung gestellt werden.

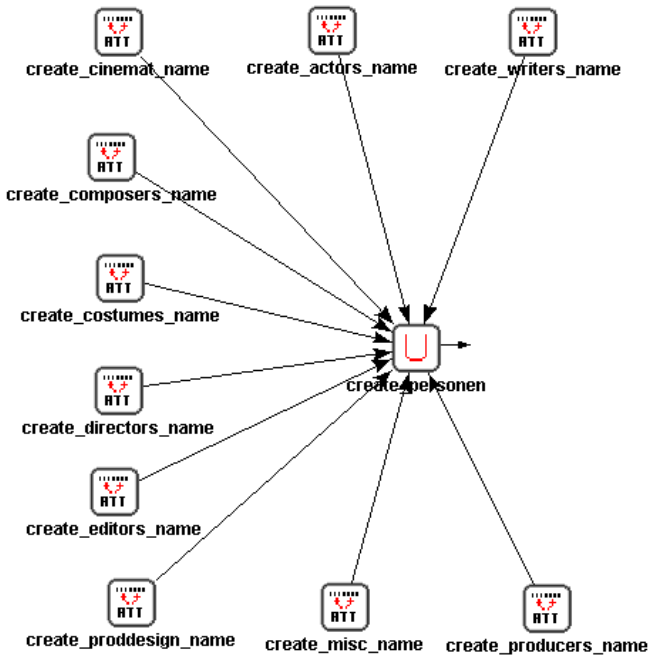


Abbildung 7.13.: Kette zur Selektion der Namen der Schauspieler und Schauspielerinnen

Abbildung 7.12 auf dieser Seite zeigt das Vorgehen in der Kettenstruktur von Mining-Mart.

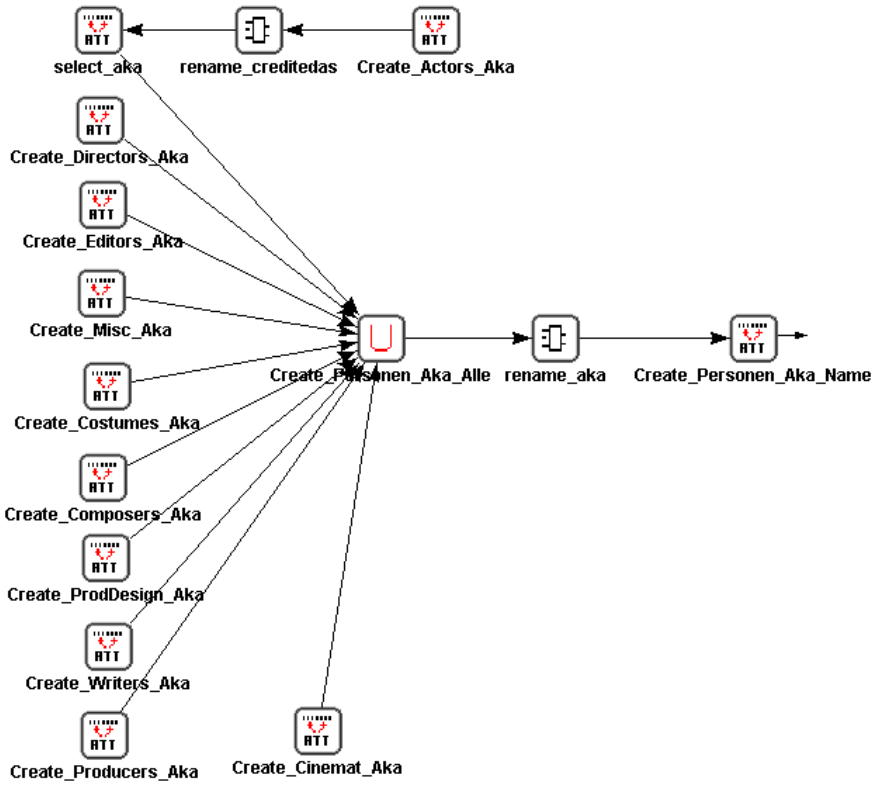


Abbildung 7.14.: Kette zur Selektion der alternativen Namen der Schauspieler und Schauspielerinnen

Abbildung 7.15.: Kette zur Selektion der Namen der Figuren

Es werden insgesamt drei Ketten benötigt:

- Eine Kette zur Selektion der Personennamen
- eine Kette zur Selektion der alternativen Personennamen
- eine Kette zur Vereinigung der beiden Konzepte, um ein Konzept für die Entitätsklasse 'Person' zu erstellen.

Die Kette zur Selektion der Personennamen, die in Abbildung 7.13 zu sehen ist, zeigt keine neuen Operatoren im Vergleich zum letzten Abschnitt. Bei der Selektion der alternativen Personennamen – zu sehen in Abbildung 7.14 auf dieser Seite – muss man nun beachten, dass bei einer späteren Vereinigung der beiden erstellten Konzepte die Bezeichnungen der Attribute identisch sein müssen. Somit wird als letzter Schritt nach

Abbildung 7.16.: Kette zur Erstellung der Entitätsklasse mit Personen. Zu beachten ist hier besonders die Struktur, da auf Konzepte aus anderen Ketten zugegriffen wird. Dies ist an dem eingehenden Pfeil am Vereinigungs-Operator zu erkennen. Abbildung 7.12 zeigt die zwei Transitionen, die in diese Kette hineinführen und dadurch die Konzepte der anderen Ketten zur Verfügung stellen.

der Selektion der alternativen Personennamen das Attribut 'aka' ebenfalls in 'name' umbenannt.

So kann dann mit Hilfe von der Kette in 7.16 die Entitätsklasse mit allen Personennamen wie zuvor spezifiziert erzeugt werden.

Es ist bereits an diesen Ketten zu sehen, dass die Anzahl der benutzten Konzepte und die durchzuführenden Operationen bei einer manuellen Datenvorverarbeitung für die ausführende Person verständlich ist, eine weitere Person sich aber nur mühselig einarbeiten könnte. Weiterhin sind Änderungen an den SQL-Anweisungen nur schwer möglich, wobei in MiningMart auf einfache Art und Weise ein Operator eingefügt werden kann. Möchte man zum Beispiel auf alle Personennamen eine Funktion anwenden, so muss nur ein zusätzlicher Operator bei der Zusammenführung der Konzepte eingefügt werden. Dies ist auf einfache und effiziente Weise möglich. MiningMart passt alle davon abhängigen Operatoren an und der Compiler erzeugt alle abhängigen Konzepte daraufhin neu.

Dazu sei darauf hingewiesen, dass die SQL-Anweisungen für ein Konzept im Konzepteditor eingesehen werden können. So kann jeder Schritt auch auf relationaler Ebene nachvollzogen werden, falls dies gewünscht oder erforderlich ist.

Die Erstellung der Entitätsklasse 'Figuren' erfordert nur eine Selektion des Attributes 'role' auf einem Konzept, da in keinem anderen Konzept als 'imdb_actors' Informationen dazu erfasst werden. Die Kette ist in Abbildung 7.15 auf der vorherigen Seite zu sehen. Diese Kette ist auch nicht von den Konzepten, die für die Erstellung der Entitätsklasse 'Personen' generiert wurden, abhängig. Daher sind keine Transitionen in Abbildung 7.12, die aus der Kette 'Chain_Figuren' hinein- bzw. herausführen.

Länderbezeichnungen

Es wurde bereits erläutert, dass zwei Entitätsklassen für Länderbezeichnungen erstellt werden. Hier wird die Kette für die Erstellung der Entitätsklasse für die vollständigen ('long') Länderbezeichner vorgestellt.

Beispiel 7.6. *Es sollen Beispiele für abgekürzte und vollständige Länderbezeichnungen gegeben werden.*

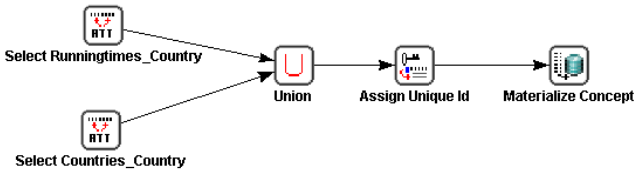


Abbildung 7.17.: Kette zur Erstellung der Entität mit vollständigen Länderbezeichnungen

Vollständige Länderbezeichner lauten zum Beispiel:

- *Thailand*
- *Tunisia*
- *Turkey*

Abgekürzte Länderbezeichner hingegen:

- *do*
- *dz*
- *de*

Abbildung 7.17 auf dieser Seite zeigt die Kette, die zunächst die Länderbezeichnungen selektiert und anschließend die von anderen Ketten bereits vorgestellten Schritte vornimmt.

Die andere Kette zur Erstellung der Entitätsklasse mit den verkürzten Länderbezeichnern ist ebenso aufgebaut, es wird nur eine Selektion auf anderen Konzepten durchgeführt.

Zusätzlich wird jedoch mit einem zusätzlichen Operator eine Ersetzung der Länderbezeichner vorgenommen: Die Datenexploration zeigte, dass unterschiedliche Schreibweisen bezüglich Groß- und Kleinschreibung bei den verkürzten Länderbezeichnern existieren, daher werden die Länderbezeichner in Kleinschreibung umgewandelt.

Wiederum fällt auf, dass durch die Erstellung der Kette für die vollständigen Länderbezeichner die andere Kette direkt davon abgeleitet und eine Anpassung bedingt durch ein Ergebnis der Datenexploration ohne Schwierigkeiten möglich ist.

Filmtitel

Die Entitätsklasse mit den Filmtiteln kann auf zwei Weisen erstellt werden. Die Rohdaten der Internet Movie Database enthalten bereits eine Datendatei, in der jeder Filmtitel genau einmal aufgeführt wird. Eine einfache Selektion dieses Attributs aus dem entsprechenden Konzept (imdb_titles) und ein anschließendes Generieren der eindeutigen Bezeichner würde somit ausreichen.

[7] also zum Beispiel: us, Us, US

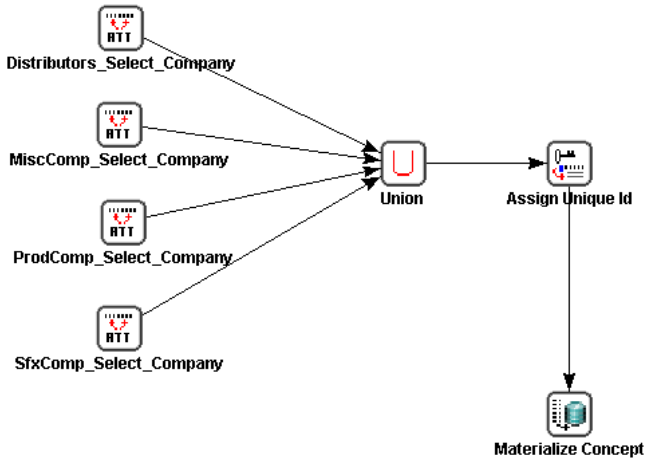


Abbildung 7.18.: Kette zur Erstellung der Entitätsklasse mit Firmenbezeichnungen

Während der Erstellung der Arbeit ist jedoch aufgefallen, dass die Liste der Titel nicht ganz genau mit denjenigen in den anderen Datendateien übereinstimmt, da die Datendateien zu unterschiedlichen Zeitpunkten erstellt werden. Daher gibt es noch eine weitere Kette zur Erstellung der Entitätsklasse mit den Filmtiteln, die auf allen Konzepten außer 'imdb_titles', die den Filmtitel enthalten, eine Selektion auf das Attribut mit dem Filmtitel durchführt und so wie bereits von anderen Ketten bekannt die Entitätsklasse erstellt.

Die Ketten müssen nicht hier abgebildet werden, da keinerlei neue Aspekte eingeführt werden.

Firmenbezeichnungen

Zuletzt sollen die Ketten zur Erstellung der Firmenbezeichnungen erwähnt werden. Es wird wiederum auf allen Konzepten, die einen Firmenbezeichner enthalten, eine Selektion durchgeführt und anschließend in der Vereinigung der Konzepte ein eindeutiger Bezeichner für jeden Eintrag generiert. Die Kette ist in Abbildung 7.18 zu sehen.

Überblick über die Entitätsklassen

Alle Entitätsklassen wurden mit den erläuterten Ketten in MiningMart erstellt. Tabelle 7.3 auf der nächsten Seite gibt Auskunft über die Anzahl der Entitäten in den einzelnen Klassen. Weiterhin sei auf Abbildung 7.19 verwiesen, die alle Entitäten mit Attributbezeichnern zeigt.

7.4.3. Erstellung der Kreuztabellen

Der Begriff der Kreuztabelle (engl. cross-table) wird normalerweise benutzt, um Beziehungen zwischen Tabellen auf der relationalen Ebene auszudrücken. In dem hier zu

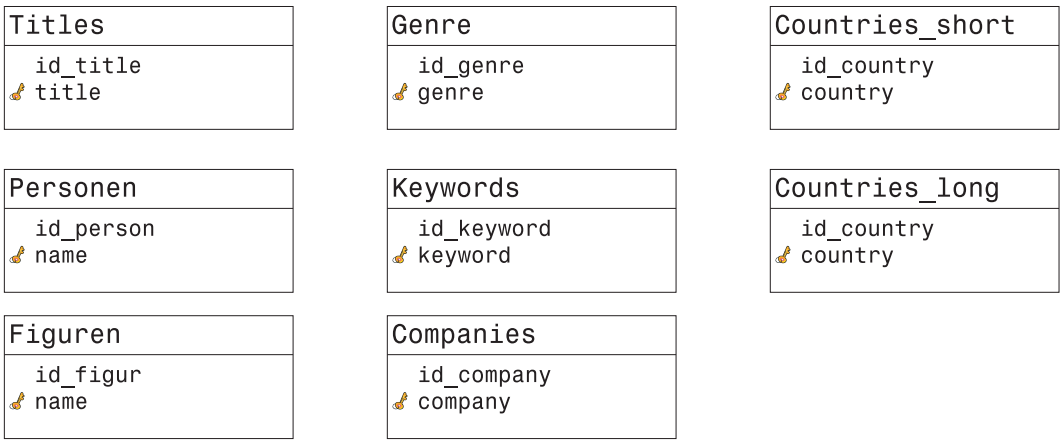


Abbildung 7.19.: Alle Entitätsklassen des Modells mit Attributen. Jede Entitätsklasse besitzt genau zwei Attribute. Beide Attribute sind eindeutig, d.h. auf der relationalen Ebene würde die Relation auch mit nur einem Attribut weiterhin die gleiche Anzahl Tupel enthalten. Aus diesem Grund wurden auch nicht die IDs, sondern vielmehr das eigentlich mit Aussage belegte Attribut in dem Modell als Schlüsselattribut ausgezeichnet.

Entitätsklasse	# **Entitäten**
Figuren	1.356.249
Filmtitel	749.674
Firmen	123.701
Genres	28
Länder, kurz	227
Länder, voll	179
Personen	1.843.682
Schlüsselwörter	33.207

Tabelle 7.3.: Anzahl Tupel der einzelnen Entitätsklassen

erstellenden Modell müssen die Entitäten mit den ursprünglichen Daten verknüpft werden. Dies geschieht dadurch, dass die Informationen, die in der Entitätsklasse bereits gespeichert sind, in den Ausgangskonzepten durch eindeutige Bezeichner ersetzt werden, die bei der Erstellung der Entitätsklassen generiert wurden.

Beispiel 7.7. *Die Entitätsklasse 'Titles' enthält zwei Attribute: den Filmtitel und einen eindeutigen Bezeichner.*

Das Konzept 'imdb_genres' beinhaltet zum Beispiel ebenfalls das Attribut Filmtitel. Für jedes Tupel wird dieses Attribut mit genau einem eindeutigen Bezeichner der zugehörigen Entität aus 'Titles' ersetzt. So werden alle Attributwerte der Filmtitel durch die entsprechenden eindeutigen Bezeichner der Entitätsklasse 'Titles' ersetzt werden.

Werden nach dem in Beispiel 7.7 beschriebenen Vorgehen alle Attribute, zu denen eine Entitätsklasse existiert, ersetzt, so wird das auf diese Weise gebildete Konzept in dieser

Arbeit als 'Kreuztabelle' bezeichnet.

Ein anderer Begriff wäre sicherlich auch denkbar, aber die Nähe zur relationalen Ebene wird dadurch klar. Im Konzepteditor in MiningMart wird eine solche Kreuztabelle als Beziehung zwischen zwei Konzepten sichtbar. Die gebildete Kreuztabelle dient somit als Mittel der Zuordnung. Weiterhin wird genau diese Kreuztabelle zusammen mit den zugehörigen Entitätsklassen auf der relationalen Ebene mit Hilfe von MiningMart-Operatoren materialisiert, um eine – dem Modell entsprechende – Eingabe für das auf der Datenbank arbeitende Lernverfahren RDT/DM zu erstellen. Das Modell wird mit Hilfe spezieller Materialisierungsoperatoren auf diese Ebene abgebildet, die in Abschnitt 6.3.2 auf Seite 56 besprochen wurden.

Es können in dieser Arbeit nicht alle Ketten zur Erstellung der Kreuztabellen abgebildet werden. Es können aber drei Typen von Kreuztabellen identifiziert werden, die interessante Aspekte aufwerfen. Alle weiteren Kreuztabellen können mit nahezu identischen Schritten erzeugt werden. Es sei bereits jetzt auf die Abbildungen 7.20, 7.23 und 7.25 verwiesen. Es handelt sich bei den Abbildungen um Skizzen, auf welche Entitätsklassen die Kreuztabellen verweisen und welche Attribute durch eindeutige Bezeichner ersetzt werden.

In den folgenden Unterabschnitten werden drei charakteristische, interessante Ketten ausführlich beschrieben.

Kreuztabelle der Schlüsselwörter

Das Konzept mit den Schlüsselwörtern besteht aus den zwei Attributen Filmtitel und Schlüsselwort (siehe Abschnitt 7.4.2). Für beide Attribute wurde eine Entitätsklasse erstellt, so dass die Attributwerte durch eindeutige Bezeichner der Entitäten ersetzt werden können.

Abbildung 7.21 zeigt die Kette zur Erstellung der Kreuztabelle. Die beiden ersten beiden Operatoren vom Typ 'JoinByKey' führen einen Verbund mit der jeweiligen Entitätsklasse durch, was dazu führt, dass die Ausgangskonzepte neben den ursprünglichen Attributen auch noch zusätzlich die eindeutigen Bezeichner der Entitäten mitführen. Daher ist ein Operator vom Typ 'FeatureSelection' anzuwenden, der nur noch die beiden Attribute mit den eindeutigen Bezeichnern selektiert.

Die Datenexploration zeigte bei der Analyse der Schlüsselwörter und den Genres, dass einige Einträge fälschlicherweise doppelt erfasst wurden. Daher enthält die Kette einen Operator zum Entfernen doppelter Einträge. Dieser Operator hätte bereits zu Beginn der Kette stehen können, aber aus Praxiserfahrung arbeitet der Operator beziehungsweise das Datenbanksystem, welches die Daten verwaltet, schneller auf den numerischen Attributwerten als auf den Zeichenketten, die durch die eindeutigen Bezeichner ersetzt wurden. Somit wurde der Operator an dieser Position in die Kette eingefügt.

Der letzte Schritt ist die Erstellung der Beziehungen zwischen den Konzepten im Konzepteditor. Weiterhin wird das Konzept auch auf relationaler Ebene materialisiert. Es wird dabei sichergestellt, dass die Entitäten mit entsprechenden Primärschlüsseln dort vorhanden sind und die Kreuztabelle diese durch Fremdschlüssel referenziert.

Nach genau diesem Schema ist auch die Kette zur Erstellung der Kreuztabelle für die Genres (imdb_genres), für die Laufzeiten der Filme (imdb_runningtimes) und für die Bewertungen der Filme (imdb_ratings) zu modellieren.

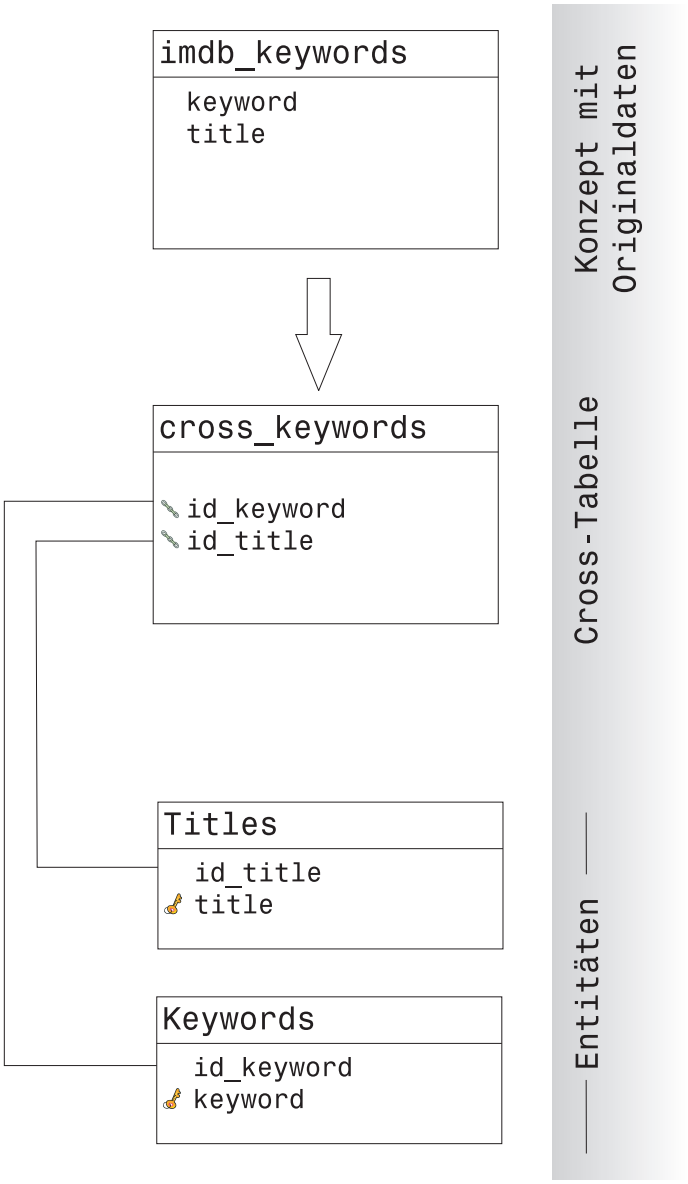


Abbildung 7.20.: Zusammensetzung der Kreuztabelle für 'imdb_keywords'. Das Konzept enthält Schlüsselwörter zu Filmen. Beide Attribute des Konzepts werden durch eindeutige Bezeichner der entsprechenden Entitätsklassen ersetzt.

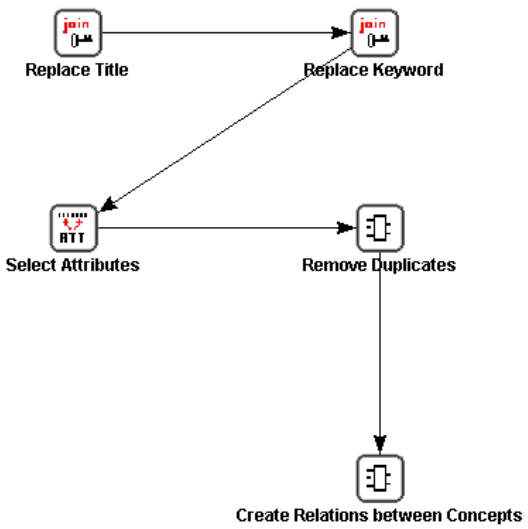


Abbildung 7.21.: Kette zur Erstellung der Kreuztabelle für die Schlüsselwörter

Diese Kette stellt den einfachsten Fall dar, da sichergestellt ist, dass die zu ersetzenden Attribute immer mit Werten belegt sind. Dies ist bei den personenbezogenen Kreuztabellen noch ein wichtiger Aspekt.

Das Schema in Abbildung 7.20 zeigt die genauen Abhängigkeiten zwischen den einzelnen Konzepten.

Kreuztabelle für firmenbezogene Konzepte

Als firmenbezogene Konzepte werden die Konzepte bezeichnet, in denen beteiligte Firmen zu Filmen aufgeführt werden. In dieser Arbeit wurden folgende Firmentypen berücksichtigt:

- Distributoren (imdb_distributors)
- Spezialeffektfirmen (imdb_sfxcomp)
- Produktionsfirmen (imdb_prodcomp)
- weitere Firmen (imdb_misccomp).

Alle diese Konzepte zeichnen sich durch die dieselben Attribute aus, so dass nur eine Kette für diesen Typ vorgestellt werden muss.

Die hier abgedruckte Kette in Abbildung 7.22 auf der nächsten Seite bezieht sich dabei auf die Spezialeffektfirmen, ebenso Schema 7.23.

Im Gegensatz zu der im letzten Abschnitt vorgestellten Kette gibt es nun drei Attribute, die durch eindeutige Bezeichner ersetzt werden müssen. Dies geschieht durch Benutzung der gleichen Operatoren wie im letzten Abschnitt. Auch der abschließende Schritt zur Erstellung der Relationen ist hier zu finden. Doppelte Einträge müssen nicht

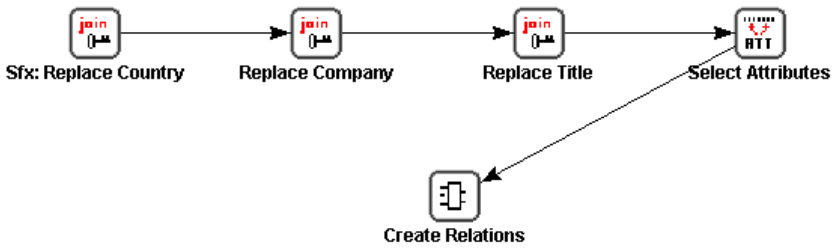


Abbildung 7.22.: Kette zur Erstellung der Kreuztabelle für die firmenbezogenen Konzepte

entfernt werden, da die Datenexploration keine Notwendigkeit hierfür zeigte. Eine alternative Modellierung könnte diesen Schritt natürlich noch in die Kette einbeziehen.

Kreuztabelle für personenbezogene Konzepte

Personenbezogene Konzepte beinhalten Informationen zu Personen, die mit Filmen in der Datenbank in Beziehung gebracht werden konnten. Dies sind die folgenden Konzepte:

- Schauspieler und Schauspielerinnen (imdb_actors)
- Kameraleute (imdb_cinematographers)
- Komponisten (imdb_composers)
- Kostümdesigner (imdb_costume-designers)
- Regisseure (imdb_directors)
- Schriftsteller (imdb_writers)
- Cutter (imdb_editors)
- Designer der Produktion (imdb_proddesigner)
- Produzenten (imdb_producers)
- Weitere (imdb_miscellanoeus).

Wie bereits bei den firmenbezogenen Konzepten haben auch diese Konzepte alle denselben Aufbau.

Das Schema, das die Zusammensetzung der Kreuztabelle aufzeigt, ist in Abbildung 7.25 auf Seite 96 zu finden.

Die Kette ist in Abbildung 7.24 auf Seite 95 zu sehen und beinhaltet einen vollkommen neuen Aspekt bei der Erstellung der Kreuztabellen. Wie im Schema zu sehen ist, sind die alternativen Personennamen ebenfalls mit der Entitätsklasse 'Personen' zu verknüpfen. Bisher konnte in allen Ketten der Operator 'JoinByKey' benutzt werden, der über einen Verbund die entsprechenden eindeutigen Bezeichner dem Ausgangskonzept hinzufügt. Das Problem bei den personenbezogenen Informationen ist, dass nicht alle Einträge der

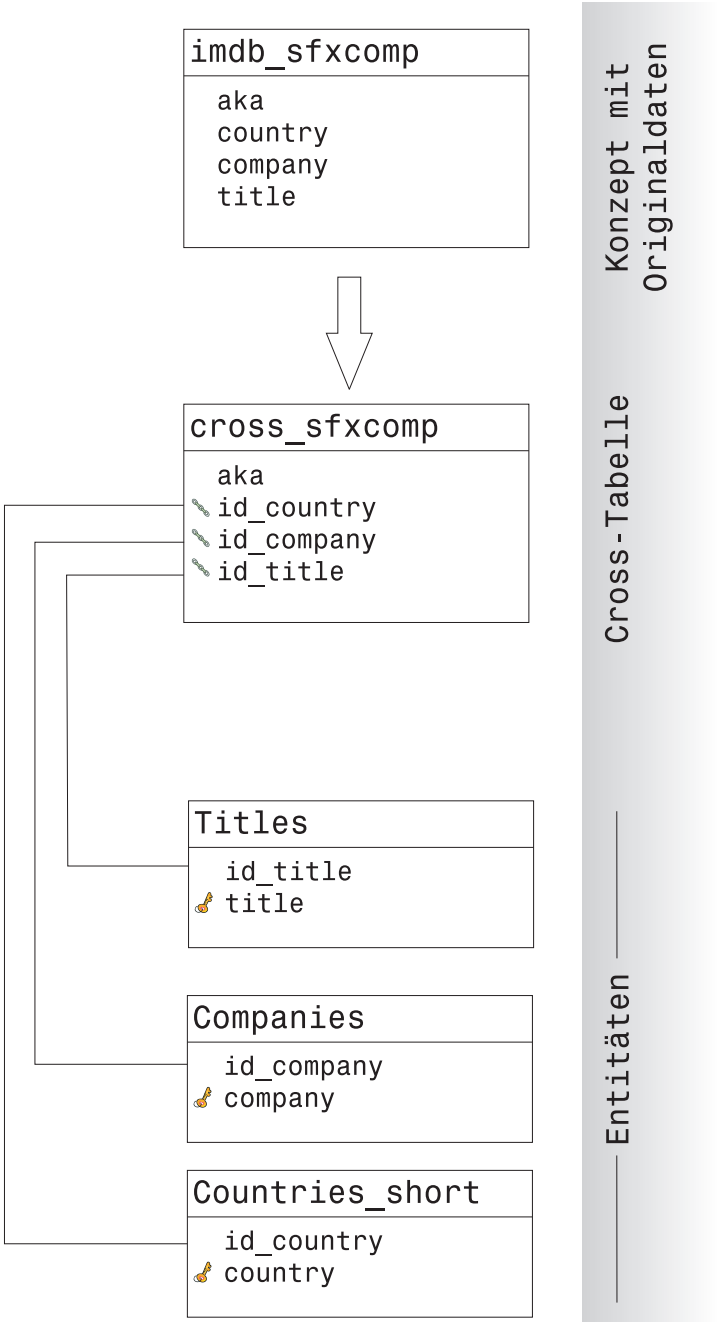


Abbildung 7.23.: Zusammensetzung der Keuztabelle für firmenbezogene Konzepte. Das entsprechende Konzept enthält an einem Film beteiligte Firmen (zum Beispiel Spezialeffekte). Die Firmen sind dazu Ländern (verkürzte Angabe) zugeordnet. Daher folgt die Verknüpfung zu den verkürzten Länderbezeichnungen (Countries_short). Die Bezeichnungen der Firmen bilden ebenfalls eine Entitätsklasse, so dass auch dieses Attribut durch einen eindeutigen Bezeichner ersetzt werden kann.

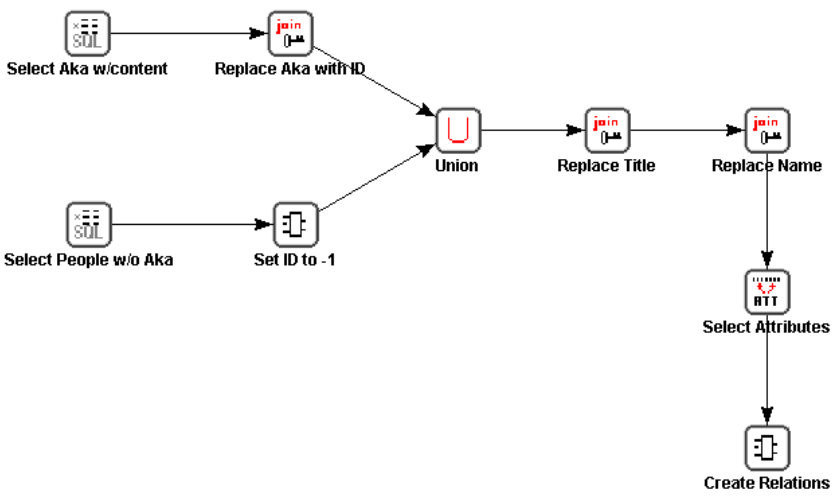


Abbildung 7.24.: Kette zur Erstellung der Kreuztabelle für die personenbezogenen Konzepte

Konzepte einen alternativen Namen aufweisen. Der Einsatz des Operators 'JoinByKey' hätte zur Konsequenz, dass die Kreuztabelle nur Einträge aufweist, die auch einen alternativen Namen haben. Das würde bedeuten, alle Informationen zu Personen, die keinen alternativen Namen in einem Filmprojekt benutzt haben, gingen verloren. Dies beträfe offensichtlich den größten Teil der Daten der Konzepte.

Aus diesem Grund wird eine Selektion vorangestellt, die das jeweilige personenbezogene Konzept in zwei Konzepte aufteilt. Zum einen in ein Konzept, welches alle Personeneinträge mit alternativem Namen enthält, und zum anderen ein weiteres Konzept, welches alle Personeneinträge ohne alternative Namensangabe enthält. Für das Konzept, welches die alternativen Namen enthält, fährt man nun fort wie bei den anderen Ketten. Mittels 'JoinByKey' wird der eindeutige Bezeichner für den alternativen Namen zum Konzept hinzugefügt. In dem Konzept, welches alle Personeneinträge ohne alternative Namen enthält, wird eine Merkmalsgenerierung durchgeführt. Es wird ein Attribut für den eindeutigen Bezeichner generiert und mit einem Wert belegt, der nicht als eindeutiger Bezeichner vergeben wurde. Da nur positive Bezeichner von MiningMart vergeben werden, wurde in dieser Kette für diesen Fall der Wert '−1' gewählt.

Nach dieser unterschiedlichen Behandlung können die beiden Konzepte wieder zu einem vereinigt werden, da die Attributmengen wieder identisch sind. Jetzt kann in der Kette fortgefahren werden wie in allen zuvor beschriebenen Ketten.

7.4.4. Selektion der Daten für den Lernprozess

Die Selektion der Daten ist nicht nur im Hinblick auf das zu verwendende Lernverfahren wichtig, sondern auch im Hinblick auf das Lernziel.

Als Lernverfahren soll in dieser Arbeit das Lernverfahren RDT/DM dienen, bei dem es sich um ein relationales Regellernsystem handelt. Bisherige Experimente bezogen einige Tabellen der Datenbank ein, jedoch wurden nie Benutzergruppen, wie in Anforderung 1

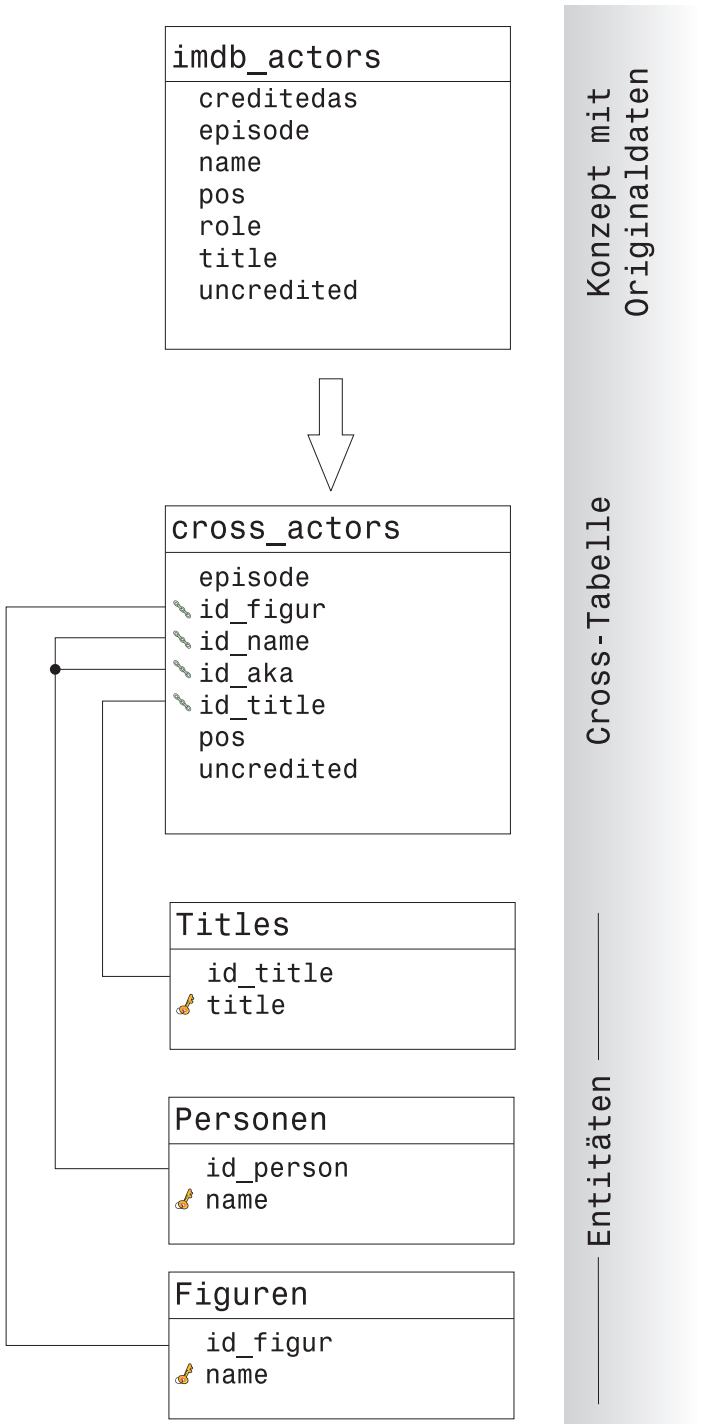


Abbildung 7.25.: Zusammensetzung der Kreuztabelle für 'imdb_actors'. Das Konzept enthält alle Schauspieler und Schauspielerinnen der Internet Movie Database. Dazu werden der Name aufgeführt, unter dem der Schauspieler bzw. die Schauspielerin in dem Filmprojekt geführt wurde, und welche Figur verkörpert wurde. Somit existieren Verknüpfungen zu den Entitätsklassen mit Personennamen und Figurennamen. Weiterhin bildet diese Kreuztabelle einen Spezialfall, da gleich zwei Attribute eindeutige Bezeichner aus derselben Entitätsklasse beziehen (bei Namen und alternativen Namen handelt es sich beide Male um Personennamen).

auf Seite 66 festgehalten, berücksichtigt.

Das folgende Kapitel wird exemplarisch auf den hier modellierten Daten lernen und dabei den Versuchsaufbau von Münstermann [48] nachvollziehen.

An dieser Stelle sei nur betont, dass die Verwendung der Operatoren zur Materialisierung der Konzepte in den vorgestellten Ketten bereits mit dem Hintergedanken, die Daten direkt weiterverwenden zu können, erfolgte.

Jedoch stellte sich nach ersten Experimenten heraus, dass Münstermann die Datenmenge nicht ohne Grund einschränkte und nicht auf allen Daten der von ihm gewählten Tabellen einen Lernlauf durchführte.

Aus diesem Grund ist eine feinere Selektion auf den Konzepten mit allen Daten durchzuführen. Diese gebildeten Konzepte werden dann materialisert und zum Lernen verwendet. Da der genaue Versuchsaufbau im folgenden Kapitel geschildert wird, werden auch die verwendeten Ketten dort vorgestellt.

7.5. Schritt 5: Evaluierung

In diesem Schritt ist nach CRISP-DM vorgesehen, dass die zu Beginn des Prozesses gestellten Ziele validiert werden. Weiterhin sind Defizite der Modellierung hier aufzudecken, was in der Regel eine anschließende Modifikation der Modellierung erfordert. Die Erstellung eines Überblicks über realisierte Anforderungen und eine Festlegung der Folgeschritte ist ebenfalls in diesem Schritt vorgesehen.

Im letzten Abschnitt wurde die Modellierung bereits ausführlich motiviert und begründet. Es wurden sowohl Vor- und Nachteile sowie Alternativen diskutiert. Die hier vorgestellte Lösung erfüllt die in den letzten Abschnitten gestellten Anforderungen. So besteht die Möglichkeit, die Daten getrennt für verschiedene Benutzergruppen zu betrachten (Anforderung 1). Die Rohdaten wurden dazu aufbereitet und in ein Datenbanksystem zur Verarbeitung mit MiningMart eingelesen (Anforderungen 2 bis 5). Das folgende Kapitel wird zeigen, dass auch Anforderung 6 erfüllt wird. Dort wird auch klar werden, dass die Modellierung flexibel für viele weitere Lernverfahren als Eingabe dienen kann. Ein Benutzer, der sich nur mit einem Lernverfahren beschäftigen möchte, kann mit Hilfe des erstellten Modells mit wenigen Ketten zur Selektion der Informationen übersichtlich und strukturiert seine Lerndaten zusammenstellen. Somit wird also auch der Schritt der Zusammenstellung der Lerndaten transparent und verständlich, der in der Praxis bisher meist eher undurchsichtig und schwer nachzuvollziehen war.

7.6. Schritt 6: Anwendungsphase

Wie bereits in der Einleitung dieses Kapitels beschrieben, ist die Anwendung der Modellierung in einem eigenen Kapitel beschrieben. Die Anwendung umfasst die Benutzung des Regellernwerkzeugs RDT.

Eine weitere mögliche Anwendung, zum Beispiel ein Programm zur Präsentation der Daten anzubieten, welches auf Konzepten bzw. relationalen Tabellen der Modellierung aufsetzt, wird hier nicht betrachtet.

8. Anwendungsphase: Regellernen mit RDT/DM

Dieses Kapitel beschreibt Experimente, die auf den aufbereiteten Daten mit dem Regelwerkzeug RDT/DM durchgeführt wurden.

Dabei werden zunächst in Abschnitt 8.1 notwendige Anpassungen an RDT/DM beschrieben, da die Version Münstermanns auf heutigen Systemen nicht mehr funktionierte.

Daraufhin wird in Abschnitt 8.2 auf die Hauptziele der Lernläufe eingegangen, wobei der Schwerpunkt auf der Beschreibung des Prozesses liegt, wie Daten dem Regellernwerkzeug mit Hilfe von MiningMart und der darin erfolgten Modellierung zur Verfügung gestellt werden können. Es werden dabei die Vorteile des Einsatzes von MiningMart gegenüber dem üblichen Vorgehen in der Praxis dargestellt.

Die Beschreibung der durchgeführten Lernläufe ist dann in Abschnitt. Die Ergebnisse werden nach den Beschreibungen erläutert und in Abschnitt 8.4 kurz diskutiert.

8.1. Notwendige Anpassungen an RDT/DM

Münstermann verfasste [48] im Jahr 2002. Seit diesem Zeitpunkt wurden sowohl die Sprache Java als auch die Datenbanktreiber samt Datenbank weiterentwickelt. Zudem benutzte er nicht Oracle als Datenbanksystem für das Lernverfahren, sondern PostgreSQL[1]. Aus diesem Grund ist die Implementierung und Konfiguration angepasst worden, um mit Hilfe von aktuellen Treibern mit einer aktuellen Version des Datenbanksystems kommunizieren zu können.

Ein Versuch, RDT/DM an das Datenbanksystem Oracle anzubinden, um direkt auf den von MiningMart auf der relationalen Ebene materialisierten Konzepten lernen zu können, ist nicht erfolgreich gewesen, da das System auf spezifische SQL-Anweisungen aufbaut, die nur unter postgreSQL zur Verfügung stehen. Aus diesem Grund wurden die von MiningMart materialisierten Datenbanktabellen samt Primär- und Fremdschlüsseln in eine postgreSQL-Datenbank übernommen. Alle Lernläufe wurden dann auf dieser Datenbank durchgeführt. Zum Einstieg in PostgreSQL kann zum Beispiel [15] herangezogen werden.

Die genauen Details der am Quellcode bzw. der Konfiguration durchgeführten Änderungen sind für diesen Abschnitt der Arbeit jedoch nicht von Bedeutung und daher im Anhang auf Seite 137 zu finden.

8.2. Grundlegende Aspekte für alle durchgeführten Experimente

Es soll gezeigt werden, dass bedingt durch die Modellierung die Durchführung des Lernschrittes im KDD-Prozess einfacher durchzuführen ist.

Um einen Vergleich zu ermöglichen, orientiert sich der Versuchsaufbau an Münstermann [48], der die Experimente durchführte, um seine Erweiterung zu RDT/DB namens RDT/DM vorzustellen.

[1] URL: `http://www.postgresql.org/`

Münstermann benutzte dabei die folgenden Informationen. In Klammern ist der Name des Ausgangskonzeptes der Datenvorverarbeitung angegeben:

- Filmtitel (imdb_movies)
- Schauspieler (imdb_actors)
- Regisseure (imdb_directors)
- Bewertungen der Filme (imdb_ratings)
- Schlüsselwörter (imdb_keywords)
- Genre (imdb_genres).

Sein Lernziel war es, Regeln zu bestimmen, die gute bzw. schlechte Filme beschreiben.

8.2.1. Zusammenstellen der Daten

Münstermann wählte zunächst aus allen Filmen die 100 besten und die 100 schlechtesten aus. Dazu definierte er eine Rangfolge, so dass die Filme unter Einbeziehung der Gesamtanzahl der abgegebenen Stimmen bewertet werden.

Von den Autoren der Internet Movie Database wird die folgende Formel als Denkanstoß geliefert, die Münstermann auch für die Erstellung der Rangfolge benutzte.

$$\text{Rang} = \left(\frac{v}{v+k}\right) \cdot X + \left(\frac{k}{v+k}\right) \cdot C \tag{8.1}$$

In Formel 8.1 sind dabei den Variablen die folgenden Werte zugeordnet:

X	=	arithmetisches Mittel[2]der Bewertungen zu diesem Film
v	=	Anzahl der Bewertungen für den Film
k	=	minimale Anzahl an Stimmen
C	=	das arithmetische Mittel über die Bewertungen aller Filme

Um die 100 besten Filme zu ermitteln, wurde der Rang mit $k = 400$, für die 100 schlechtesten mit $k = 50$ berechnet. Um diese Bewertung zu berechnen, benutzte Münstermann damals SQL-Anweisungen.

Es bietet sich nun an, das Modell in MiningMart um eine Kette zu erweitern, die diese Aufgabe löst.

Als problematisch erweist sich zur Zeit für MiningMart der Wert C. Dieser kann zwar mit Hilfe von MiningMart und der Statistiken im Konzepteditor ermittelt werden, jedoch besteht keine Möglichkeit, diesen Wert innerhalb der Parameter eines Operatoren zu benutzen. Aus diesem Grund sind die Ketten zur Ermittlung der Rangfolge in MiningMart in gewisser Weise hart codiert, jedoch können diese Änderungen über die Benutzeroberfläche recht schnell durchgeführt werden.

Man geht also so vor, dass in dem Ausgangskonzept mit den Filmbewertungen zwei neue Merkmale kreiert werden, welche nach Formel 8.1 zu berechnen sind. Es sind also zwei Attribute zu generieren, da der Rang einmal mit $k = 400$ und einmal mit $k = 50$

[2] = engl. *mean*, nicht zu verwechseln mit dem Durchschnitt (= engl. *average*)

ermittelt werden muss. Die Selektion der gewünschten Anzahl an guten und schlechten Filmen kann dann mit Hilfe des generierten Attributs durchgeführt werden. Zusätzlich ist nach der Selektion noch ein weiteres (boolesches) Attribut zu generieren, welches Auskunft darüber gibt, ob es sich um einen Film aus der Menge der guten oder schlechten Filme handelt.

Auf diese Weise wurde für die Lernaufgabe ein neues Konzept mit 200 Einträgen generiert, welches dann Bewertungen der 100 besten und der 100 schlechtesten Filme enthält.

So konnte Münstermann nun die erforderlichen Tupel für die Tabellen mit Schlüsselwörtern, Genres, Filmtiteln, Schauspielern und Regisseuren bestimmen, indem er dort nur die Tupel aufgenommen hat, die einem der 200 Filme zugeordnet werden konnten.

Da die Entitätsklassen und Kreuztabellen, die in unserem Modell erstellt wurden, jedoch weiterhin zum Beispiel alle Filmtitel enthalten und nicht nur die Filmtitel, die untersucht werden sollen, kann nun für jedes dieser Konzepte, welches im Lernschritt verwendet werden soll, eine weitere Kette modelliert werden. Man muss erreichen, dass nur noch die Informationen in einem Konzept geführt werden, die im Lernschritt zu untersuchen sind.

In MiningMart ist dies einfach durch die Anwendung des 'JoinByKey'-Operators möglich, indem ein Verbund zwischen dem Konzept mit den ermittelten 200 Filmen und dem entsprechenden Konzept durchgeführt wird. Eine anschließende Selektion und Materialisierung liefert dann ein neues, an die Lernaufgabe angepasstes Konzept.

An dieser Stelle mag es in der Praxis komfortabler sein, die gesamten Konzepte auf der relationalen Ebene zu materialisieren und einmalig eine Abfolge von SQL-Anweisungen mit Hilfe von MiningMarts Konzepteditor zu erstellen, die aus der Gesamtmenge die entsprechenden Teilkonzepte erzeugen. Mit Hilfe einer Software, die dem Anwender Einstellungsmöglichkeiten für die Auswahl der zu lernenden Daten gibt, ist dieser Prozess, wie es nach dem CRISP-DM angedacht wird, zu automatisieren.

Münstermann erstellte auch noch eine weitere Hilfstabelle, die angibt, ob ein Schauspieler mindestens in 2 guten bzw. 2 schlechten Filmen mitspielt. Dies ist mit MiningMart bisher nur durch direkte Angabe von SQL-Anweisungen (Operator 'GenericFeatureConstruction') möglich. Sollte man ein Lernverfahren verwenden, welches auch auf einem Datenbanksystem arbeitet, so ist es einfacher, dies direkt zu verwenden. Stellt man jedoch eine Eingabe für eine SVM zusammen, die meist auf Textdateien arbeitet, so ist der Weg über MiningMart sicherlich schneller.

An dieser Stelle ist die Zusammenstellung der Daten bei der Benutzung von MiningMart und dem in dieser Arbeit erstellten Modell beendet.

Münstermann hingegen musste nun noch die entsprechenden Primärschlüssel und Fremdschlüsselbeziehungen anlegen. Dieser Schritt wurde durch das Modell bzw. durch den Einsatz der entsprechenden Materialisierungsoperatoren automatisch erledigt.

Die SQL-Anweisungen zur Erstellung der Datenbankschemata, die für die folgenden Lernläufe erstellt wurden, sind im Anhang B.2 auf Seite 130 zu finden.

8.2.2. Benutzergruppen

Durch die Modellierung von Kreuztabellen und einer einzigen Entitätsklasse mit Personennamen können auf einfache Art und Weise zum Beispiel beliebige Personengruppen wie Komponisten, Kameraleute, Cutter usw. in den Lernprozess einbezogen werden. Es

ist lediglich sicherzustellen, dass die Entitätsklasse 'Person' immer die benutzten Entitäten enthält.

Es ist jedoch davor zu warnen, bei der Arbeit mit RDT die Entitätsklassen grundsätzlich vollständig zu übernehmen, da ein Lernlauf sehr lange dauert oder unter Umständen aufgrund der Menge der Daten nicht erfolgreich durchgeführt werden kann.

8.3. Exemplarische Lernläufe

Es werden nun einige exemplarische Lernläufe durchgeführt.

Alle Lernläufe werden auf einer Menge von Filmen durchgeführt, die wie im letzten Abschnitt geschildert zusammengestellt wurden.

8.3.1. Experiment 1

Dieses Experiment bildet einen Lernlauf für einen fiktiven Kameramann nach, der sich fragt, an welchen Filmprojekten er sich beteiligen sollte. Die Lernaufgabe besteht also darin, genau wie bei Münstermann, Regeln für gute Filme zu ermitteln. Im Gegensatz zur Arbeit von Münstermann wurden allerdings in diesem Experiment nur Kameraleute in die Wissensbasis einbezogen und keinerlei Schauspieler. Die von Münstermann benutzte Bewertung der Schauspieler wurde aber auch für die Kameraleute übernommen.

Die Filmtitel wurden wie im letzten Abschnitt beschrieben anhand ihrer Bewertungen zusammengestellt. In Abhängigkeit davon wurden die

- Bewertungen
- Genres
- Länder
- Kameraleute.

ebenfalls in eigenen Tabellen abgelegt und über entsprechende Fremdschlüsselbeziehungen gemäß der Modellierung miteinander verknüpft. Schlüsselwörter werden im nächsten Lernlauf einbezogen. Tabelle 8.1 auf der nächsten Seite zeigt alle Tabellen mit Attributen.

Im Unterschied zur Wissensbasis von Münstermann wurden in dieser Arbeit bei den Genres und Schlüsselwörtern Kreuztabellen eingesetzt. Der Versuchsaufbau sieht die Benutzung der Belegungen von Genres in Prädikaten vor. RDT/DM kann jedoch aus den eindeutigen Bezeichnern, bei denen es sich in den Kreuztabellen um Zahlenwerte handelt, keinerlei Prädikate bilden.

Liegen numerische Werte vor, steht der Abbildungstyp 3 nicht zur Verfügung. Dieses Problem wurde durch das Ersetzen der eindeutigen Bezeichner durch alphanumerische Werte und Anpassung des Datentyps in der Datenbank gelöst.

Weiterhin sind die Entitätsklassen für Prädikatabbildungen im Lernlauf nicht zu benutzen, da Prädikate der Form '$genre\_Comedy(id\_genre)$' gebildet werden und 'id_genre' keinerlei Beziehung mehr zum Film, sondern nur zu der Kreuztabelle aufweist.

Es bleibt zu überlegen, ob ein neuer Abbildungstyp in einem solchen Fall neue Erkenntnisse bzw. einfacher zu interpretierende Regeln liefern würde.

Im Abschnitt C.2 ist die genaue Konfiguration für die Prädikatabbildungen der einzelnen Tabellen in RDT/DM nachzulesen. Es ist darauf hinzuweisen, dass zunächst nur für

Tabelle	**Attribute**		# Tupel
titles	id_title		200
	title		
rankings	id_title		200
	top	x	
	rank		
	wrank		
crossgenres	id_title		541
	id_genre	x	
crosscountries	id_title		249
	id_country	x	
crosscinemat	id_title		228
	id_person		
	id_aka		
	uncredited		
personen	id_person		223
	name		
genres	id_genre		28
	genre		
countries	id_country		22
	countries		
personentop	id_person		204
	top	x	

Tabelle 8.1.: Datenbanktabellen des ersten Experiments. Ein 'x' hinter einem Attribut gibt an, dass für jeden eindeutigen Attributwert ein Prädikat generiert wurde (vergleiche RDT/DM, Abbildungstyp 3 in [48]).

die Genres und Länderbezeichner Prädikate gebildet wurden. Es wurden 24 Prädikate für die Genres und 22 für die Länderbezeichner gebildet. Dazu wurde das Attribut 'uncredited' auf 2 Prädikate abgebildet, welches darüber Auskunft gibt, ob ein Kameramann im Abspann des Filmprojektes erwähnt wird.

Für die Bewertung der Filme wurden, wie bei den Versuchen von Münstermann, zwei Prädikate über das Attribut 'top' gebildet.

Die folgenden Regelschemata wurden dem Lernverfahren zur Verfügung gestellt:

1. $root(Q): \ Q(X).$
2. $ma1(Q, P1, P2): \ Q(X) \leftarrow P1(X), P2(X).$
3. $ma2(Q, R1, P1): \ Q(X) \leftarrow R1(X, Y), P1(X).$
4. $ma3(Q, P1, R1, P3): \ Q(X) \leftarrow P1(X), R1(X, Y), P3(Y).$

Das Akzeptanzkriterium wurde über das Verhältnis der Differenz der positiven und negativen abgedeckten Beispiele der zu untersuchenden Hypothese zu allen positiven Beispiele ausgedrückt:

$$\frac{|pos(h)|}{|concl(h)|} - \frac{|neg(h)|}{|concl(h)|} \geq 0,3 \tag{8.2}$$

In diesem Beispiel sind also 30% mehr positive als negative Beispiele abzudecken. Es kann hier von 'Prozenten' gesprochen werden, da genau 100 positive Beispiele in den Tabellen enthalten sind.

Das Beschneidungskriterium wird festgelegt über:

$$\frac{|pos(h)|}{|concl(h)|} \leq 0,1 \tag{8.3}$$

Somit müssen mindestens 10% aller positiven Beispiele durch die untersuchte Hypothese abgedeckt werden. Als Zielprädikat liefert

$$ranking\_top\_21(id\_title)$$

Regeln für gute Filme. 'ranking_top_21(id_title)' ist das Prädikat, welches von RDT/DM aus dem Attribut 'top' der Bewertungen erzeugt wird.

Die folgenden Regeln wurden nach einem Lernlauf mit einer Dauer von circa 2 Minuten gefunden:

1. $ranking\_top\_21(X) \leftarrow crosscinemat\_id\_person(X, Y), crosscinemat\_uncredited\_10(X).$
2. $ranking\_top\_21(X) \leftarrow crosscinemat\_id\_aka(X, Y), crosscinemat\_uncredited\_10(X).$
3. $ranking\_top\_21(X) \leftarrow crosscinemat\_uncredited(X, Y), crosscinemat\_uncredited\_10(X).$
4. $ranking\_top\_21(X) \leftarrow crossgenres\_id\_genre(X, Y), crosscinemat\_uncredited\_10(X).$
5. $ranking\_top\_21(X) \leftarrow crossgenres\_id\_genre(X, Y), crossgenres\_id\_genre\_2G9(X).$
6. $ranking\_top\_21(X) \leftarrow crossgenres\_id\_genre(X, Y), crossgenres\_id\_genre\_9G12(X).$

Die ersten beiden Regeln sagen aus, dass Kameraleute, die an erfolgreichen Filmen mitgewirkt haben, dann auch im Abspann mit Namen oder ihrem alternativen Namen aufgeführt werden. Weiterhin wurden mit den Regeln 5 und 6 zwei Genres gefunden, die auf erfolgreiche Filmprojekte schließen lassen. Es handelt sich um die Genres 'Drama' und 'Thriller', was bereits von Münstermann gefunden wurde.

Münstermann konnte im Jahre 2002 auch noch das Land 'USA' als Kriterium für einen guten Film nachweisen. Aufgrund der vielen Filme aus Hollywood ergibt diese Regel unmittelbar Sinn. Dieses Ergebnis konnte in diesem Lernlauf nicht nachgewiesen werden. Als Gründe können angeführt werden, dass zum einen die Internet Movie Database im

Tabelle	**Attribute**		# Tupel
keywords	id_keyword		3680
	keyword		
crosskeywords	id_title		7850
	id_keyword	x	

Tabelle 8.2.: Zusätzliche Datenbanktabellen des zweiten Experiments.

Jahre 2002 noch nicht mit vielen Filminformationen aus anderen Ländern bestückt war, wogegen nun auch zum Beispiel nahezu alle deutschen Filmproduktionen dort mit einer sehr großen Detailtiefe zu finden sind. Weiterhin gibt es in den letzten Jahren auch sehr viele Filmprojekte, die nicht mehr Hollywood entstammen und trotzdem viele gute Bewertungen erhalten. Dies wird durch das Fehlen der von Münstermann gefundenen Regel gezeigt.

Der nächste Lernlauf wird die Anzahl der Prädikate erhöhen, um noch mehr Kriterien zu finden.

8.3.2. Experiment 2

Im Vergleich zum ersten Experiment sollen nun noch Schlüsselwörter zu den Filmen in den Lernvorgang einbezogen werden. Dazu wird für jedes Schlüsselwort (also für jede Entität der Entitätsklasse 'Schlüsselwörter') ein Prädikat gebildet. Der fiktive Kameramann erhält somit bei einer gefundenen Regel, die Schlüsselwörter beinhaltet, auch noch Informationen darüber, welche Thematik ein Film behandeln sollte, damit das Projekt vielversprechend für ihn wird.

Es werden somit zwei neue Tabellen in den Lernlauf einbezogen. Tabelle 8.2 auf dieser Seite führt diese in der gleichen Form wie Tabelle 8.1 auf.

Regelschemata, Akzeptanz und Beschneidungskriterium werden nicht verändert. Für die Schlüsselwörter werden 3680 neue Prädikate im Vergleich zum letzen beschriebenen Versuch gebildet.

Nach 48 Minuten wurde das Lernergebnis ausgegeben, wobei nur eine weitere Regel bezüglich der Schlüsselwörter gefunden wurde.

$$ranking\_top\_21(X) \leftarrow crossgenres\_id\_genre(X,Y), \\ crosskeywords\_id\_keyword\_320729149(X).$$

Diese Regel ist jedoch recht aussagekräftig und liefert ein anderes Ergebnis im Vergleich mit dem Jahr 2002. Damals wurde das Schlüsselwort 'bicycle' als gutes Kriterum ermittelt. Dieser Lernlauf liefert jedoch das Schlüsselwort 'murder', um einen guten Film zu charakterisieren.

8.3.3. Experiment 3

In diesem letzten Experiment soll nun eine weitere Entitätsklasse in den Lernvorgang einbezogen werden. Dazu wurde die Kreuztabelle der Distributoren und die zugehörige

Tabelle	**Attribute**	# Tupel
companies	id_company	504
	company	
crossdistributors	id_title	1285
	id_company x	

Tabelle 8.3.: Zusätzliche Datenbanktabellen des dritten Experiments.

Entitätsklasse dem Lernverfahren zur Verfügung gestellt. Es werden, wie in den Experimenten zuvor, wiederum nur Distributoren betrachtet, die den 100 besten und 100 schlechtesten Filmen zugeordnet werden können.

Tabelle 8.3 auf dieser Seite zeigt die neuen Tabellen (und Prädikate) auf. Es wird dabei für jeden eindeutigen Firmenbezeichner ein Prädikat gebildet.

Alle anderen Parameter verbleiben wie in Experiment 2. Es wird jedoch nicht für jedes Schlüsselwort ein Prädikat gebildet. Die Schlüsselwörter werden also nur über die Tabellen auf Prädikate übertragen (Abbildungstypen 1 und 2).

Insgesamt werden in diesem Lernlauf 514 Prädikate verwendet. Die genauen Bezeichner sind dem Anhang in Abschnitt C.4 zu entnehmen.

Der Lernlauf benötigte circa 5 Minuten und fand unter anderem die folgenden interessanten Regeln:

1. $ranking\_top\_21(X) \leftarrow crossdist(X,Y), crosscountries\_id\_country\_14C80(X).$

2. $ranking\_top\_21(X) \leftarrow crossdist(X,Y), crosscountries\_id\_country\_15C68(X).$

Es wurden nun die Länder 'USA' und 'Deutschland[3]' als Kriterien für einen guten Film gefunden. Weiterhin wurden die folgenden Firmennamen innerhalb von Regeln der Form

$$ranking\_top\_21(X) \leftarrow rdtcrossdist(X,Y), crossdist\_id\_company\_{\_p}(X).$$

als Ergebnis ausgegeben. p steht hierbei für einen Bezeichner, mit deren Hilfe die zugehörige Firma eindeutig identifiziert werden kann.

[3] in der Datenbank als 'Germany' gespeichert

Es wurden insgesamt 11 Regeln ausgegeben, die Distributionsfirmen zu erfolgreichen Filmen ausweisen:

Name	**eindeutiger Bezeichner**
Columbia TriStar Home Video	D18275
United Artists	D29567
Warner Bros.	D76695
Buena Vista International	D110020
The Criterion Collection	D13355
Warner Home Video	D14750
Argentina Video Home (AVH)	D62514
Paramount Pictures	D71967
A-Film Distribution	D77390
Gativideo	D98255
Metropolitan Filmexport	D116996

Die Aufnahme der Distributoren in die Wissensbasis hat somit konkrete Informationen geliefert, welche Distributionsfirmen erfolgreiche Filme produzieren. Diese Aussage konnte durch Hinzunahme der entsprechenden Entitätsklasse mit den Firmenbezeichnern und der Kreuztabelle mit den Distributoren erzielt werden. Durch die Modellierung in MiningMart konnte die Wissensbasis schnell und unproblematisch für diese Lernaufgabe erweitert werden. Außerdem wurde durch dieses Experiment gezeigt, dass die Lernaufgaben sehr einfach für beliebige Benutzergruppen erweitert bzw. angepasst werden können.

8.4. Diskussion der Ergebnisse

Es wurde gezeigt, dass die durchgeführte Modellierung die Durchführung von Lernaufgaben auf den Daten der Internet Movie Database erheblich erleichtert. Durch die individuellen Ketten in MiningMart besteht die Möglichkeit, für jede Benutzergruppe individuelle Lernaufgaben zu erstellen. Der letzte Abschnitt zeigte exemplarisch Lernläufe, die auf die Benutzergruppe der Kameraleute zugeschnitten war. Andere Daten wurden absichtlich nicht in die Wissensbasis aufgenommen. Bei anderen Geschäftsdaten könnte dies zum Beispiel erforderlich sein, da nicht jede Benutzergruppe auf alle Daten Zugriff haben soll. Trotzdem können alle Benutzergruppen auf eine einfache Weise mit Daten für Versuchsdurchführungen versorgt werden. Durch die strukturierte Modellierung wird sichergestellt, dass Fehler bei der Zusammenstellung vermieden oder sofort erkannt werden können. MiningMart bietet Hilfestellung sowohl bei der Erstellung von speziellen Merkmalen, die einige Lernverfahren zum Beispiel als Bewertungskriterium benötigen, als auch bei der Sichtung der Daten, auf denen gelernt wird. So konnten bei den hier vorgestellten Versuchen die zusätzlichen Attribute zur Bewertung der Personen und Filme generiert werden. Weiterhin konnte über den Konzepteditor und dessen Statistikübersicht verifiziert werden, dass die gewünschten Daten an den Lernprozess weitergegeben wurden.

Die gelernten Ergebnisse decken sich zum Teil mit denen von Münstermann, jedoch konnten, bedingt durch die durchgehenden Änderungen an der Internet Movie Database und die Einbeziehung weiterer Informationen, auch neue Aspekte aufgedeckt werden.

Spezielle Anpassungen des Lernverfahrens an die Modellierung mit Entitätsklassen sind anzudenken, da die Prädikatenbezeichnungen, die eindeutige Bezeichner enthalten, doch recht störend sind und ein Ermitteln der zum Prädikat zugehörigen Entität nicht mehr notwendig wäre.

9. Zusammenfassung

Diese Arbeit hat gezeigt, dass bei der Wissensentdeckung in Datenbanken anhand des CRISP-DM Modells die Durchführung einer ausführlichen Datenexploration, anschließender Datenvorverarbeitung mit und Modellierung in MiningMart entscheidende Vorteile bringt.

Es wurden dazu zunächst wichtige Grundlagen zur Wissensrepräsentation und der Wissensentdeckung in Datenbanken abgesteckt, bevor dann das eigentliche Fallbeispiel, die Daten der Internet Movie Database, bearbeitet wurde.

Bereits die erste Datensichtung lieferte entscheidende Aspekte. So war die Struktur der Daten in einem Zustand, der nicht direkt in eine Datenbank übernommen werden konnte. Nur durch ein Kennenlernen des Sachbereiches konnte überhaupt ein Vorgehen zur Übernahme der Daten ausgearbeitet werden. Es wurde dann eine Software erstellt, um die Daten in ein Datenbanksystem einzulesen. Die anschließende Datenexploration auf der relationalen Ebene lieferte die Ideen für die hinterher tatsächlich realisierten Aufbereitungs- und Modellierungsansätze.

Dabei wurden die Datenbanktabellen mit Hilfe von MiningMart zuerst als Konzepte modelliert und anschließend auf der konzeptuellen Ebene eine Ontologie des Sachbereichs erstellt. Dieser Prozess ist anhand von strukturierten, übersichtlichen Datenvorverarbeitungsketten in MiningMart komplett transparent und robust für Änderungen gestaltet worden.

Des Weiteren konnte gezeigt werden, dass die Modellierung eine einfache Zusammenstellung der Daten für Lernverfahren ermöglicht. Es wurden die aufbereiteten Konzepte aus MiningMart wieder auf die relationale Ebene übertragen. Die Struktur des Modells wurde dabei mit den Möglichkeiten eines relationalen Datenbanksystems, wie zum Beispiel Fremdschlüsselbeziehungen, nachempfunden, so dass das eingesetzte multirelationale Lernverfahren diese Struktur zur Wissensentdeckung nutzen konnte. Auf diesen materialisierten Konzepten wurden direkt Lernläufe durchgeführt.

Ein Kernaspekt dieser Arbeit war, dass eine Modellierung mit MiningMart sogar dazu benutzt werden kann, Lernverfahren mit Beziehungen zwischen den einzelnen Komponenten der Lernaufgabe (in diesem Fall Datenbanktabellen) zu versorgen. Aber nicht nur die Struktur der Daten ist für das Lernsystem wichtig, auch die Qualität der Daten und die Selektion der Daten sind nicht zu vernachlässigende Aspekte. Beide Aspekte wurden durch die Bearbeitung des Fallbeispiels mit MiningMart in dieser Arbeit behandelt und diskutiert.

RDT/DM hätte nicht direkt auf die Rohdaten angewendet werden können. Münstermann hat die Daten umgewandelt und in eine Datenbank eingelesen, jedoch dokumentierte er die Datenvorverarbeitung nicht und der Schwerpunkt seiner Arbeit lag auf dem multi-relationalen Lernverfahren RDT.

Außerdem konnten mit Hilfe weniger weiterer Ketten spezielle Informationen für das eingesetzte Lernverfahren erstellt werden, die zuvor immer mit unübersichtlichen, nur schwer nachzuvollziehenden SQL-Anweisungen generiert wurden.

So ist es durch Benutzung der Ketten in MiningMart nicht mehr nur Experten mög-

lich, Eingaben für ein Lernverfahren zusammenzustellen, da MiningMart eine einfach zu bedienende grafische Oberfläche bietet und diese benutzt werden kann, um Eingaben direkt auf die relationale Ebene abzubilden. Der Benutzer muss bei Einsatz des erstellten Modells auch nicht mehr die Namen der Datenbanktabellen kennen, sondern kann gezielt mit Entitätsklassen und dazugehörigen Kreuztabellen arbeiten, die sprechende Namen aufweisen.

Durch die Erkenntnisse, die mittels der Modellierung erzielt wurden, konnten die Ergebnisse des Lernverfahrens zudem viel einfacher interpretiert und auch anhand des Modells nachvollzogen werden. Die Funktionalität von MiningMart hat sich also auch nach Durchführung der Versuche als hilfreich erwiesen, um Lernergebnisse nachzuvollziehen. Der Vorteil hierbei war, dass nicht mit SQL-Anweisungen auf einer Datenbank gearbeitet werden musste, sondern Anfragen auf verständlich benannten Konzepten durchgeführt werden konnten.

Die Modellierung ermöglicht ein Arbeiten mit der Wissensbasis auf einer Ebene, die von technischen Aspekten, wie SQL-Anfragen oder Tabellenbezeichnern, abstrahiert.

Die ausgearbeiteten Modellierungsketten können auch auf anderen Geschäftsdaten Anwendung finden. Ein Übertragen ist durch die Architektur, die in einem vorangegangenen Kapitel besprochen wurde, von MiningMart einfach zu realisieren. Dies wurde in dieser Arbeit ebenfalls gezeigt, da Ketten in Typen eingeordnet werden konnten, die häufig auf mehrere Konzepte mit ähnlichem Aufbau angewendet werden konnten. Alle diese Ketten wurden ausführlich beschrieben und sind im Hauptteil der Arbeit abgebildet.

Das Fallbeispiel und alle Experimente sind mit Hilfe des Anhangs dieser Arbeit einfach nachzuvollziehen. Eine Erweiterung der Ketten für den Einsatz anderer Lernverfahren ist ebenfalls einfach möglich.

Literaturverzeichnis

[1] A. Aamodt and E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *Artificial Intelligence Communications*, 7(1):39–59, 1994.

[2] Charles W. Bachman. Data Structure Diagrams. *DATA BASE*, 1(2):4–10, 1969.

[3] Dave Beckett and Brian McBride. RDF/XML Syntax Specification (Revised). Internet, URL: http://www.w3.org/TR/rdf-syntax-grammar/, Februar 2004.

[4] Joachim Biskup. *Grundlagen von Informationssystemen.* Vieweg, Lehrbuch Informatik, 1995.

[5] Peter Brockhausen and Katharina Morik. Wissensentdeckung in relationalen Datenbanken: Eine Herausforderung für das maschinelle Lernen. In Gholamreza Nakhaeizadeh, Hrsg., *Data Mining, theoretische Aspekte und Anwendungen, Wirtschaftsinformatik*, Seiten 193–211. Physica Verlag, 1998.

[6] Donald D. Chamberlin and Raymond F. Boyce. SEQUEL: A structured English query language. In *FIDET '74: Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*, pages 249–264, New York, NY, USA, 1974. ACM Press.

[7] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rüdiger Wirth. Crisp–Dm 1.0. Technical report, The CRISP–DM Consortium, August 2000.

[8] Peter Pin-Shan Chen. The entity-relationship modeltoward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976.

[9] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[10] E. F. Codd. Further normalization of the data base relational model. *IBM Research Report, San Jose, California*, RJ909, 1971.

[11] E. F. Codd. Extending the database relational model to capture more meaning. *ACM Trans. Database Syst.*, 4(4):397–434, 1979.

[12] E. F. Codd. *The relational model for database management: version 2.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[13] Erik Darwinkel and Olaf Rem. The concept editor: A description of part of the Mining Mart HCI.

[14] C.J. Date. *An Introduction to Database Systems.* The Systems Programming Series. Addison-Wesley Publishing Company, 7. edition, 2000.

[15] Korry Douglas. *PostgreSQL*. Sams, Paperback, 2nd edition, July 2005.

[16] Muthukkaruppan Annamalai Engineering. Guidelines for Constructing Reusable Domain Ontologies.

[17] Eric Miller et al. Semantic web, activity overview. Internet, URL: http://www.w3.org/2001/sw/, May 2006.

[18] Timm Euler. How to implement M4 operators. TechnicalReport TR12-04, IST Project MiningMart, IST-11993, 2002.

[19] Timm Euler. Operator specifications. TechnicalReport TR12-02, IST Project MiningMart, IST-11993, 2002.

[20] Timm Euler. Churn Prediction in Telecommunications Using MiningMart. In *Proceedings of the Workshop on Data Mining and Business (DMBiz) at the 9th European Conference on Principles and Practice in Knowledge Discovery in Databases (PKDD)*, Porto, Portugal, 2005.

[21] Timm Euler. Modelling Data Mining Processes on a Conceptual Level. In *Proceedings of the 5th International Conference on Decision Support for Telecommunications and Information Society*, Warsaw, Poland, 2005.

[22] Timm Euler. Dual Control of KDD Processes in KDD Tools. In *Proceedings of KDD '06*, August 2006.

[23] Timm Euler, Katharina Morik, and Martin Scholz. MiningMart: Sharing Successful KDD Processes. In Andreas Hotho and Gerd Stumme, Hrsg., *LLWA 2003 – Tagungsband der GI-Workshop-Woche Lehren – Lernen – Wissen – Adaptivität*, Seiten 121–122, October 2003.

[24] Timm Euler and Martin Scholz. Using Ontologies in a KDD Workbench. In P. Buitelaar, J. Franke, M. Grobelnik, G. Paaß, and V. Svátek, editors, *Workshop on Knowledge Discovery and Ontologies at ECML/PKDD '04*, pages 103–108, Pisa, Italy, September 2004.

[25] Ronald Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst.*, 2(3):262–278, 1977.

[26] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery: An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 1, pages 1–34. AAAI/MIT Press, 1996.

[27] Günther Görz, Josef Schneeberger, and Claus R. Rollinger. *Handbuch der künstlichen Intelligenz*. Oldenbourg Vg, München, 2002.

[28] T. R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.

[29] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[30] M. Grüninger and M. Fox. Methodology for the Design and Evaluation of Ontologies. In *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13*, 1995.

[31] N. Guarino and P. Giaretta. Ontologies and knowledge bases - towards a terminalogical clarification. In N.J. Mars, editor, *Towards Very Large Knowledge Bases - Knowledge Building and Knowledge Sharing 1995*, pages pages 25–32. IOS Press, Amsterdam, 1995.

[32] Clyde W. Holsapple and K. D. Joshi. A collaborative approach to ontology design. *Commun. ACM*, 45(2):42–47, 2002.

[33] D.R. Howe. *Data Analysis for Data Base Design.* Edword Arnold, 1983.

[34] Michael R.A. Huth and Mark D. Ryan. *Logic in Computer Science. Modelling and Reasoning About Systems.* Cambridge University Press, 2nd edition, 2004.

[35] William Kent. A Simple Guide to Five Normal Forms in Relational Database Theory. *Communications of the ACM*, 26(2):120–125, Feb. 1983.

[36] Jörg-Uwe Kietz. *Induktive Analyse relationaler Daten.* Doktorarbeit, Technische Universität Berlin, 1996.

[37] J.U. Kietz, A. Vaduva, and R. Zücker. Mining Mart: Combining Case-Based Reasoning and Multi-Strategy Learning into a Framework to reuse KDD Application. In R. Michalki and P. Brazdil, editors, *Proceedings of the fifth International Workshop on Multistrategy Learning (MSL2000), Guimares, Portugal*, 2000.

[38] Hanna Köpcke. Häufigkeitsbasierte Merkmalsgenerierung für die Wissensentdeckung in Datenbanken. Diplomarbeit, Fachbereich Informatik, Universität Dortmund, 2003.

[39] Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. Internet, URL: http://www.w3.org/TR/owl-features/, Februar 2004.

[40] K. Morik, S. Wrobel, J.-U. Kietz, and W. Emde. *Knowledge Acquisition and Machine Learning - Theory, Methods, and Applications.* Academic Press, London, 1993.

[41] Katharina Morik. *Maschinelles Lernen.* Skript zur gleichnahmigen Vorlesung, Lehrstuhl 8, Universität Dortmund, 3. Auflage, 1999.

[42] Katharina Morik and Martin Scholz. The MiningMart Approach. In *Workshop Management des Wandels der 32. GI Jahrestagung*, 2002.

[43] Katharina Morik and Martin Scholz. The MiningMart Approach to Knowledge Discovery in Databases. In Ning Zhong and Jiming Liu, editors, *Intelligent Technologies for Information Analysis*, chapter 3, pages 47–65. Springer, 2004.

[44] Katharina Morik, Martin Scholz, and Timm Euler. MiningMart: Final Report.

[45] S.H. Muggleton. Inductive Logic Programming. *New Generation Computing*, 8(4):295–318, 1991.

[46] S.H. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[47] S.H. Muggleton and L. De Raedt. Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[48] Dirk Münstermann. Wissensentdeckung in Datenbanken mit dynamischer Anpassung des Hypothesentests. Diplomarbeit, Fachbereich Informatik, Universität Dortmund, April 2002.

[49] D. Nardi and R. J. Brachman. *The Description Logic Handbook*, chapter An Introduction to Description Logics, pages 5–44. Cambridge University Press, 2002.

[50] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF (W3C Working Draft 12 October 2004). Internet, URL: http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/, Oktober 2004.

[51] Dorian Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers, 1999.

[52] Liam Quin. Extensible Markup Language (XML). Internet, URL: http://www.w3.org/XML/, April 2006.

[53] Dr. Harald Sack. Ontologien. in: Foliensatz und Skript, Seminar Semantic Web, Sommersemester 2006, Universität Jena, http://www.informatik.uni-jena.de, April 2006.

[54] Martin Scholz. Using constraints, conditions and assertions. TechnicalReport TR18-02, IST Project MiningMart, IST-11993, 2002.

[55] Uwe Schöning. *Logik für Informatiker*. Spektrum Akademischer Verlag, 5. Auflage, 2000.

[56] Michael E. Senko, Edward B. Altman, Morton M. Astrahan, and P. L. Fehder. Data structures and accessing in data-base systems. iii: Data representations and the data independent accessing model. *IBM Systems Journal*, 12(1):64–93, 1973.

[57] Peter Spyns, Robert Meersman, and Mustafa Jarrar. Data modelling versus ontology engineering. *SIGMOD Rec.*, 31(4):12–17, 2002.

[58] Gerd Stumme. Off to new shores: conceptual knowledge discovery and processing. *Int. J. Hum.-Comput. Stud.*, 59(3):287–325, 2003.

[59] Universität Dortmund, Lehrstuhl 8. MiningMart Casebase. URL: `http://mmart.cs.uni-dortmund.de/caseBase/index.html`.

[60] Mike Uschold. Building Ontologies: Towards a Unified Methodology. In *16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK, 1996.

[61] Mike Uschold and Michael Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[62] Thomas von Aquin. *Über Seiendes und Wesenheit. De ente et essentia.* Lateinisch-Deutsch, mit Einleitung, Übersetzung und Kommentar herausgegeben von Horst Seidl, Meiner, Hamburg, 1988.

[63] R. Wheeler and S. Aitken. Multiple algorithms for fraud detection. *Knowledge Based Systems*, 13(2-3):93–99, 2000.

[64] Wikipedia. Edgar F. Codd — Wikipedia, The Free Encyclopedia, 2006. [Online].

Anhang

A. Beispiele aus den Rohdaten

Die Rohdaten der Internet Movie Database wurden im Rahmen dieser Arbeit ausführlich untersucht. Die Ergebnisse und Schlussfolgerungen, die daraus gezogen wurden, sind im Hauptteil dieser Arbeit nachzulesen. Dieses Kapitel enthält Textinformationen aus einigen Datendateien, um diese Ergebnisse mit Beispielen zu belegen, die für den Hauptteil der Arbeit zu umfassend gewesen wären.

A.1. Einleitung der Datei 'actresses.list'

Die Erkennung des Anfangs der in die Datenbank einzulesenden Informationen bereitete oftmals Schwierigkeiten. In diesem Fall wurden die letzten beiden Zeilen des folgenden Ausschnitts als Kennung benutzt. Jede Datei besitzt eine andere Kennzeichnung des Anfangs der erfassten Informationen.

```
CRC: 0x303E8B09  File: actresses.list  Date: Fri Oct 28 01:00:00 2005

Copyright 1993-2005 The Internet Movie Database, Inc.  All rights reserved.

COPYING POLICY: Internet Movie Database (IMDb)
==============================================

    This is a database of movie related information compiled by
    Internet Movie Database Ltd (IMDb). While every effort has been
    made to ensure the accuracy of the database IMDb gives no
    warranty as to the accuracy of the information contained in the
    database. IMDb reserves the right to withdraw or delete
    information at any time.

    This service is provided for the information of users only. It is
    not provided with the intention that users rely upon the
    information for any purposes. Accordingly, IMDb shall under no
    circumstances be liable for any loss or damage, including but not
    limited to loss of profits goodwill or indirect or consequential
    loss arising out of any use of or inaccuracies in the
    information. All warranties express or implied are excluded to
    the fullest extent permissible by law.

    All information in this file is Copyright 2005 Internet Movie
    Database Limited. Reproduction, distribution or transmission by
    any means without the prior permission of IMDb is prohibited. All
    rights reserved.

    For further information contact <licensing@imdb.com>

         ------------------------------------------------------

    All data and software released by Internet Movie Database Ltd is
    freely available to anyone within certain limitations. You are
    encouraged to quote subsets of the database in USENET articles,
```

```
    movie related FAQs, magazine articles etc. We do ask, however,
    that you make reference to the source of the data and provide a
    pointer to the database for the benefit of the reader.

    Permission is granted by the copyright holder to allow free
    distribution of this file and any other part of the Internet
    Movie Database in an ELECTRONIC FORM ONLY, providing the
    following conditions are met:

         1. NO FEE OF ANY KIND, however indirect, will be charged
            for its distribution. If this file is being stored
            for later distribution to anyone that can be
            construed as a customer of yourself or your
            organisation YOU MUST contact Internet Movie Database
            Ltd for permission.

         2. Each of the database files may be distributed
            individually but only in an unaltered form. All the
            header and trailer information, including this notice
            and the details on how to access the database, must
            remain intact.

         3. Specifically the files may NOT be used to construct
            any kind of on-line database (except for individual
            personal use). Clearance for ALL such on-line data
            resources must be requested from Internet Movie
            Database Ltd

         4. In addition, copies of the Internet Movie Database
            frequently asked questions list and additions guide
            must be made available in the same area / by the same
            method as the other database files.

         5. CD-ROM distribution is prohibited without written
            permission from the Internet Movie Database Ltd

    Distribution by e-mail, BBS and Internet systems is positively
    encouraged within these limitations.

    The files and software which make up the movie database may be
    uploaded to commercial BBS systems providing that the above
    conditions are met and no *additional* fees are applied above the
    standard connect time or downloading charges.

    For further information contact <licensing@imdb.com>

         ------------------------------------------------------

Welcome to the latest version of the actresses list.

Please feel free to submit entries for actresses not already on the list,
or new entries for existing actresses.

This list is managed by Giancarlo Cairella <actors@imdb.com>; from 1993-1998
it was managed by Col Needham <col@imdb.com>; from 1990-1993 it was managed
by Andy Krieg <krieg@execpc.com>.
```

```
If you have any additions or corrections please respond by e-mail only. The
section at the end of the list contains details on the formats to use. The
copying policy is also described at the end of the list.


The Internet Movie Database consists of the following lists:

 List                 | Maintained by                                | Updated
 ---------------------|----------------------------------------------|----------
 Actors               | Giancarlo Cairella http://imdb.com/contact   | 24-06-05
 Actresses            | Giancarlo Cairella http://imdb.com/contact   | 24-06-05
 Alternative Names    | Duncan Smith http://imdb.com/contact/        | 24-06-05
 Alternative Titles   | Michel Hafner http://imdb.com/contact/       | 17-06-05
 Alternative Versions | Giancarlo Cairella http://imdb.com/contact   | 17-06-05
 Biographies          | Geoff Leonard http://imdb.com/contact/       | 24-06-05
 Business             | Giancarlo Cairella http://imdb.com/contact   | 17-06-05
 Cast Completion      | Giancarlo Cairella http://imdb.com/contact   | 17-06-05
 Certificates         | Peter Simeon http://imdb.com/contact/        | 24-06-05
 Cinematographers     | Michel Hafner http://imdb.com/contact/       | 17-06-05

[...]

 Running Times        | Mark Bailey http://imdb.com/contact/         | 24-06-05
 SFX Companies        | Mark Bailey http://imdb.com/contact/         | 24-06-05
 Sound Mix            | Mark Bailey http://imdb.com/contact/         | 24-06-05
 Soundtracks          | Ron Higgins http://imdb.com/contact/         | 24-06-05
 Tag Lines            | Mark Bailey http://imdb.com/contact/         | 24-06-05
 Technical Info       | Peter Simeon http://imdb.com/contact/        | 24-06-05
 Trivia               | Tim Norris http://imdb.com/contact/            | 24-06-05
 Writers              | Duncan Smith http://imdb.com/contact/        | 17-06-05
 ------------------------------------------------------------------------------
                                                                      dd/mm/yy

SEARCHING THE DATABASE
======================

The movie database frequently asked questions list contains more information
on the whole movie database project. For a copy send an e-mail message with
the subject "HELP FAQ" to <mail-server@imdb.com>. Here is a summary of the
ways to access the database:

(1) WWW interface

    The Internet Movie Database is available over the WWW. The following sites
    are owned and operated by or for the IMDb:

        http://us.imdb.com/                                  [USA]
        http://uk.imdb.com/                                  [UK]

    News and pointers to all IMDb sites are available at IMDb HQ:

        http://www.imdb.com/

(2) e-mail interface

    For details send a message with the subject HELP to <mail-server@imdb.com>
```

```
[...]

------------------------------------------------------------------------------
RULES:
1       Movies and recurring TV roles only, no guest appearances
2       Please submit entries in the format outlined at the end of the list
3       Feel free to submit new actresses

"xxxxx"        = a television series
"xxxxx" (mini) = a television mini-series
[xxxxx]        = character name
<xx>           = number to indicate billing position in credits
(TV)           = TV movie, or made for cable movie
(V)            = made for video movie (this category does NOT include TV
                 episodes repackaged for video, guest appearances in
                 variety/comedy specials released on video, or
                 self-help/physical fitness videos)

THE ACTRESSES LIST
==================
```

A.2. Abschluss der Datei 'actresses.list'

Dieser Abschnitt folgt unmittelbar auf den eigentlichen Bereich der Datei mit den Informationen, die erfasst werden. Ein Stopp-Zeichen, um den Abspann anzuzeigen, gibt es nicht.

Aus diesem Grund wurde zum Beispiel die Zeile, die ausschließlich aus '=' besteht als mit vorangehender Leerzeile als Erkennung für das Dateiende verwendet.

```
------------------------------------------------------------------------------
SUBMITTING UPDATES
==================

We rely on users of the database to submit corrections and additions to keep
the list as accurate and complete as possible. The most convenient method
for submitting them is via the IMDb website.  Every page about a title or
person has an Update button near the bottom which takes through the
process.

KEYWORD INTERFACE
=================

The older by-email system is now deprecated, but can be allowed
for prolific reviewers or content site owners where large numbers of
URLs to your information can be sent to us for inclusion on IMDb.

Contact us at http://imdb.com/helpdesk/contact

For further help see http://imdb.com/help/?adding/

------------------------------------------------------------------------------

COPYING POLICY
==============
```

```
This file is copyright (c) 1993-2005 by The Internet Movie Database Ltd. All
rights reserved.

Disclaimer
----------

    This is a database of movie related information compiled by
    Internet Movie Database Ltd (IMDb). While every effort has been
    made to ensure the accuracy of the database IMDb gives no
    warranty as to the accuracy of the information contained in the
    database. IMDb reserves the right to withdraw or delete
    information at any time.

    This service is provided for the information of users only. It is
    not provided with the intention that users rely upon the
    information for any purposes. Accordingly, IMDb shall under no
    circumstances be liable for any loss or damage, including but not
    limited to loss of profits, goodwill or indirect or consequential
    loss arising out of any use of or inaccuracies in the
    information. All warranties express or implied are excluded to
    the fullest extent permissible by law.

    All information in this file is Copyright 2005 Internet Movie
    Database Limited. Reproduction, distribution or transmission by
    any means without the prior permission of IMDb is prohibited. All
    rights reserved.

    For further information contact <licensing@imdb.com>

         ------------------------------------------------------

    All data and software released by Internet Movie Database Ltd is
    freely available to anyone within certain limitations. You are
    encouraged to quote subsets of the database in USENET articles,
    movie related FAQs, magazine articles etc. We do ask, however,
    that you make reference to the source of the data and provide a
    pointer to the database for the benefit of the reader.

    Permission is granted by the copyright holder to allow free
    distribution of this file and any other part of the Internet
    Movie Database in an ELECTRONIC FORM ONLY, providing the
    following conditions are met:

         1. NO FEE OF ANY KIND, however indirect, will be charged
            for its distribution. If this file is being stored
            for later distribution to anyone that can be
            construed as a customer of yourself or your
            organisation YOU MUST contact Internet Movie Database
            Ltd for permission.

         2. Each of the database files may be distributed
            individually but only in an unaltered form. All the
            header and trailer information, including this notice
            and the details on how to access the database, must
            remain intact.

         3. Specifically the files may NOT be used to construct
```

```
85              any kind of on-line database (except for individual
                personal use). Clearance for ALL such on-line data
                resources must be requested from Internet Movie
                Database Ltd

90          4. In addition, copies of the Internet Movie Database
               frequently asked questions list and additions guide
               must be made available in the same area / by the same
               method as the other database files.

95          5. CD-ROM distribution is prohibited without written
               permission from the Internet Movie Database Ltd

    Distribution by e-mail, BBS and Internet systems is positively
    encouraged within these limitations.
100
    The files and software which make up the movie database may be
    uploaded to commercial BBS systems providing that the above
    conditions are met and no *additional* fees are applied above the
    standard connect time or downloading charges.
105
    For further information contact <licensing@imdb.com>
```

A.3. Überblick über 'ratings.list'

Diese Datendatei zeichnet sich durch die feste Länge der einzelnen Attribute aus, d.h. die Attribtue werden nicht durch ein Trennzeichen getrennt, wie es bei vielen anderen Dateien der Fall ist. Leerzeichen sind durch '␣' hervorgehoben.

```
MOVIE␣RATINGS␣REPORT

New␣␣Distribution␣␣Votes␣␣Rank␣␣Title
␣␣␣␣␣␣␣0000000016␣␣␣␣␣175␣␣␣8.6␣␣!Huff␣(2004)␣(TV)
␣␣␣␣␣␣␣0000111101␣␣␣␣␣195␣␣␣5.4␣␣"'70s,␣The"␣(2000)␣(mini)
␣␣␣␣␣␣␣0.....0222␣␣␣␣␣␣11␣␣␣8.1␣␣"...e␣le␣stelle␣stanno␣a␣guardare"␣(1971)␣(mini)
␣␣␣␣␣␣␣0...000016␣␣␣␣␣␣93␣␣␣8.0␣␣"100␣Greatest␣Artists␣of␣Hard␣Rock"␣(2000)␣(mini)
␣␣␣␣␣␣␣1...1...25␣␣␣␣␣␣␣8␣␣␣8.0␣␣"100␣Greatest␣Discoveries"␣(2004)␣(mini)
␣␣␣␣␣␣␣....222.22␣␣␣␣␣␣␣5␣␣␣7.4␣␣"100␣Greatest␣Kid␣Stars"␣(2005)␣(mini)
␣␣␣␣␣␣␣300..0.112␣␣␣␣␣␣59␣␣␣4.0␣␣"100␣Most␣Metal␣Moments"␣(2004)␣(mini)
␣␣␣␣␣␣␣00....0215␣␣␣␣␣␣74␣␣␣8.5␣␣"100␣Scariest␣Movie␣Moments,␣The"␣(2004)␣(mini)
␣␣␣␣␣␣␣0000000115␣␣␣␣3064␣␣␣8.1␣␣"10th␣Kingdom,␣The"␣(2000)␣(mini)
␣␣␣␣␣␣␣0000000115␣␣␣␣␣196␣␣␣8.9␣␣"12␣stulev"␣(1977)␣(mini)
␣␣␣␣␣␣␣5.11.....3␣␣␣␣␣␣10␣␣␣2.0␣␣"125␣milioni␣di␣caz..te"␣(2001)␣(mini)
␣␣␣␣␣␣␣..00001123␣␣␣␣␣␣62␣␣␣8.2␣␣"1900␣House,␣The"␣(1999)␣(mini)
␣␣␣␣␣␣␣...00.1213␣␣␣␣␣␣22␣␣␣8.6␣␣"1915"␣(1982)␣(mini)
␣␣␣␣␣␣␣......2.26␣␣␣␣␣␣␣5␣␣␣8.3␣␣"1986␣World␣Series"␣(1986)␣(mini)
␣␣␣␣␣␣␣..00112210␣␣␣␣␣␣34␣␣␣6.8␣␣"20-e␣dekabrya"␣(1981)␣(mini)
```

A.4. Überblick über 'genres.list'

Die Genre sind ein Beispiel für die inkonsequente Trennung der Attribute durch Trennzeichen. Es werden unregelmäßig viele Trennzeichen benutzt, um den Filmtitel von dem Genre zu trennen. Weiterhin fallen die Wiederholungen des Filmtitels auf.

```
10:30␣Check-Out␣(2002)    ⇥    ⇥    ⇥    ⇥    ⇥Crime
10:30␣Check-Out␣(2002)    ⇥    ⇥    ⇥    ⇥    ⇥Drama
```

```
10:30␣Check-Out␣(2002)	Short
10:30␣P.M.␣Summer␣(1966)	Drama
"10:30␣Slot,␣The"␣(1999)	Comedy
"10:30␣Slot,␣The"␣(1999)	Music
10.32␣(1966)	Mystery
103␣Degrees␣(1993)	Documentary
103␣Degrees␣(1993)	Short
104th␣Street␣Curve,␣New␣York,␣Elevated␣Railway␣(1899)	Documentary
104th␣Street␣Curve,␣New␣York,␣Elevated␣Railway␣(1899)	Short
10.5␣(2004)␣(TV)	Action
10.5␣(2004)␣(TV)	Drama
105␣%␣alibi␣(1959)	Thriller
10.5:␣Apocalypse␣(2005)␣(TV)	Drama
10/65:␣Selbstverstümmelung␣(1965)	Short
1066:␣And␣All␣That␣(1939)␣(TV)	Musical
1066:␣And␣All␣That␣(1939)␣(TV)	Comedy
106␣Dana␣(1966)	Documentary
106␣Dana␣(1966)	Short
"106␣&␣Park:␣Prime"␣(2003)	Music
10.7␣(1997)	Short
1071␣Fifth␣Avenue:␣Frank␣Lloyd␣Wright␣&␣the␣Guggenheim␣Museum␣(1994)␣(TV)	Documentary
10-7␣for␣Life␣(1995)	Documentary
1080:␣Avalanche␣(2003)␣(VG)	Action
1080:␣Avalanche␣(2003)␣(VG)	Sport
"10-8:␣Officers␣on␣Duty"␣(2003)	Drama
"10-8:␣Officers␣on␣Duty"␣(2003)	Crime
"109"␣(2005)	Documentary
10:96:␣Training␣Night␣(2005)␣(TV)	Comedy
10%␣actif␣(2003)	Short
10%␣actif␣(2003)	Drama
10␣Again␣(2003)	Short
```

B. Datenbankschemata

Dieses Kapitel dient als Übersicht über alle Datenbankschemata, die für diese Arbeit erstellt wurden. Erstellt wurden zum einen die Datenbankschema, die die Daten aus den komma-separierten Dateien aufnehmen und zum anderen die Schemata der Tabellen, die als Eingabe für die Lernaufgabe gedient haben.

B.1. Metadaten der Rohdaten

Wie in Abschnitt 7.3 erwähnt, wurden die komma-separierten Daten in Tabellen eingelesen. Dieser Abschnitt listet die SQL-Anweisungen auf, die diese Tabellen in einer Oracle-Datenbank erstellen. Diese Tabellen bilden die Grundlage für die Ausgangskonzepte in MiningMart.

Der zweite Unterabschnitt enthält alle verwendeten SqlLoader-Beschreibungsdateien.

B.1.1. SQL-Anweisungen zur Erstellung der Datenbanktabellen

```
CREATE TABLE         IMDB_ACTORS (
                        NAME VARCHAR2(600) ,
                        TITLE VARCHAR2(600) ,
                        ROLE VARCHAR2(1000) ,
                        CREDITEDAS VARCHAR2(600) ,
                        UNCREDITED VARCHAR2(5) ,
                        POS VARCHAR2(20) ,
                        EPISODE VARCHAR2(600)
                        );

CREATE TABLE         IMDB_CINEMAT (
                        NAME VARCHAR2(500) ,
                        TITLE VARCHAR2(500) ,
                        AKA VARCHAR2(500) ,
                        WHAT VARCHAR2(700) ,
                        UNCREDITED NUMBER(22)
                        );

CREATE TABLE         IMDB_COMPOSERS (
                        NAME VARCHAR2(500) ,
                        TITLE VARCHAR2(500) ,
                        AKA VARCHAR2(500) ,
                        WHAT VARCHAR2(700) ,
                        UNCREDITED NUMBER(22)
                        );

CREATE TABLE         IMDB_COSTUMEDESIGNERS (
                        NAME VARCHAR2(500) ,
                        TITLE VARCHAR2(500) ,
                        AKA VARCHAR2(500) ,
                        WHAT VARCHAR2(500) ,
                        UNCREDITED NUMBER(22)
```

```
                        );

CREATE TABLE         IMDB_COUNTRIES (
                        TITLE VARCHAR2(500) ,
                        COUNTRY VARCHAR2(100) ,
                        REFERENCE VARCHAR2(500)
                        );

CREATE TABLE         IMDB_DIRECTORS (
                        NAME VARCHAR2(500) ,
                        TITLE VARCHAR2(500) ,
                        AKA VARCHAR2(500) ,
                        WHAT VARCHAR2(500) ,
                        UNCREDITED NUMBER(22)
                        );

CREATE TABLE         IMDB_DISTRIBUTORS (
                        TITLE VARCHAR2(500) ,
                        COUNTRY VARCHAR2(20) ,
                        COMPANY VARCHAR2(500) ,
                        AKA VARCHAR2(500)
                        );

CREATE TABLE         IMDB_EDITORS (
                        NAME VARCHAR2(500) ,
                        TITLE VARCHAR2(500) ,
                        AKA VARCHAR2(500) ,
                        WHAT VARCHAR2(500) ,
                        UNCREDITED NUMBER(22)
                        );

CREATE TABLE         IMDB_GENRES (
                        TITLE VARCHAR2(500) ,
                        GENRE VARCHAR2(100)
                        );

CREATE TABLE         IMDB_KEYWORDS (
                        TITLE VARCHAR2(500) ,
                        KEYWORD VARCHAR2(200)
                        );

CREATE TABLE         IMDB_MISC (
                        NAME VARCHAR2(500) ,
                        TITLE VARCHAR2(500) ,
                        AKA VARCHAR2(500) ,
                        WHAT VARCHAR2(600) ,
                        UNCREDITED NUMBER(22)
                        );

CREATE TABLE         IMDB_MISCCOMP (
                        TITLE VARCHAR2(500) ,
                        COUNTRY VARCHAR2(20) ,
                        COMPANY VARCHAR2(500) ,
                        AKA VARCHAR2(500)
                        );

CREATE TABLE         IMDB_TITLES (
```

```
                        TITLE VARCHAR2(500) ,
                        YEAR VARCHAR2(50) ,
                        NOTES VARCHAR2(50)
                        );

CREATE TABLE          IMDB_PRODCOMP (
                        TITLE VARCHAR2(500) ,
                        COUNTRY VARCHAR2(20) ,
                        COMPANY VARCHAR2(500) ,
                        AKA VARCHAR2(500)
                        );

CREATE TABLE          IMDB_PRODDESIGN (
                        NAME VARCHAR2(500) ,
                        TITLE VARCHAR2(500) ,
                        AKA VARCHAR2(500) ,
                        WHAT VARCHAR2(500) ,
                        UNCREDITED NUMBER(22)
                        );

CREATE TABLE          IMDB_PRODUCERS (
                        NAME VARCHAR2(500) ,
                        TITLE VARCHAR2(500) ,
                        AKA VARCHAR2(500) ,
                        WHAT VARCHAR2(600) ,
                        UNCREDITED NUMBER(22)
                        );

CREATE TABLE          IMDB_RATINGS (
                        TITLE VARCHAR2(500) ,
                        DIST CHAR(10) ,
                        VOTES NUMBER(22) ,
                        RANK NUMBER(22)
                        );

CREATE TABLE          IMDB_RUNNINGTIMES (
                        TITLE VARCHAR2(500) ,
                        COUNTRY VARCHAR2(100) ,
                        DURATION NUMBER(22) ,
                        NOTES VARCHAR2(500)
                        );

CREATE TABLE          IMDB_SFXCOMP (
                        TITLE VARCHAR2(500) ,
                        COUNTRY VARCHAR2(20) ,
                        COMPANY VARCHAR2(500) ,
                        AKA VARCHAR2(500)
                        );

CREATE TABLE          IMDB_WRITERS (
                        NAME VARCHAR2(500) ,
                        TITLE VARCHAR2(500) ,
                        AKA VARCHAR2(500) ,
                        WHAT VARCHAR2(500) ,
                        P1 NUMBER(22) ,
                        P2 NUMBER(22) ,
                        P3 NUMBER(22) ,
```

```
                        UNCREDITED NUMBER(22)
                        );
```

B.1.2. Beschreibungsdateien für Oracle SqlLoader

Die komma-separierten Dateien wurden mit SqlLoader in die Datenbank eingelesen. SqlLoader benötigt dazu Beschreibungen zu den komma-separierten Dateien, um die Zuordnung zu den Attributen in der Datenbank vorzunehmen. Für jede einzulesende Datendatei der Rohdaten wurde eine solche Beschreibungsdatei erstellt.

actors.ctl

```
load data
infile 'actors.list.csv' "str '\n'"
infile 'actresses.list.csv' "str '\n'"
insert
into table "HF"."IMDB_ACTORS"
fields terminated by '\t'
TRAILING NULLCOLS
(
  name char(600),
  title char(600),
  role char(1000),
  creditedas char,
  pos char,
  uncredited char,
  episode char(600)
)
```

cinemat.ctl

```
load data
infile 'cinematographers.csv' "str '\n'"
insert
into table "HF"."IMDB_CINEMAT"
fields terminated by '\t'
TRAILING NULLCOLS
(
  name char(500),
  title char(500),
  aka char(500),
  what char(700),
  uncredited integer external
)
```

composers.ctl

```
load data
infile 'composers.csv' "str '\n'"
insert
into table "HF"."IMDB_COMPOSERS"
fields terminated by '\t'
TRAILING NULLCOLS
(
  name char(500),
  title char(500),
  aka char(500),
  what char(700),
  uncredited integer external
)
```

costume-designers.ctl

```
load data
infile 'costume-designers.csv' "str '\n'"
insert
```

```
into table "HF"."IMDB_COSTUMEDESIGNERS"
fields terminated by '\t'
TRAILING NULLCOLS
(
  name char(500),
  title char(500),
  aka char(500),
  what char(500),
  uncredited integer external
)
```

countries.ctl

```
load data
infile 'countries.csv' "str '\n'"
insert
into table "HF"."IMDB_COUNTRIES"
fields terminated by '\t'
TRAILING NULLCOLS
(
  title char(500),
  country char(100),
  reference char(500)

)
```

directors.ctl

```
load data
infile 'directors.csv' "str '\n'"
insert
into table "HF"."IMDB_DIRECTORS"
fields terminated by '\t'
TRAILING NULLCOLS
(
  name char(500),
  title char(500),
  aka char(500),
  what char(500),
  uncredited integer external
)
```

distributors.ctl

```
load data
infile 'distributors.csv' "str '\n'"
insert
into table "HF"."IMDB_DISTRIBUTORS"
fields terminated by '\t'
TRAILING NULLCOLS
(
  title char(500),
  country char(20),
  company char(500),
  aka char(500)
)
```

editors.ctl

```
load data
infile 'editors.csv' "str '\n'"
insert
into table "HF"."IMDB_EDITORS"
fields terminated by '\t'
TRAILING NULLCOLS
(
  name char(500),
  title char(500),
```

```
  aka char(500),
  what char(500),
  uncredited integer external
)
```

genres.ctl

```
load data
infile 'genres.csv' "str '\n'"
insert
into table "HF"."IMDB_GENRES"
fields terminated by '\t'
TRAILING NULLCOLS
(
  title     char(500),
  genre     CHAR
)
```

keywords.ctl

```
load data
infile 'keywords.csv' "str '\n'"
insert
into table "HF"."IMDB_KEYWORDS"
fields terminated by '\t'
TRAILING NULLCOLS
(
  title     char(500),
  keyword     CHAR
)
```

miscellaneous-companies.ctl

```
load data
infile 'miscellaneous-companies.csv' "str '\n'"
insert
into table "HF"."IMDB_MISCCOMP"
fields terminated by '\t'
TRAILING NULLCOLS
(
  title char(500),
  country char(20),
  company char(500),
  aka char(500)
)
```

movies.ctl

```
load data
infile 'movies.csv' "str '\n'"
insert
into table "HF"."IMDB_TITLES"
fields terminated by '\t'
TRAILING NULLCOLS
(
  id        FILLER,
  title      char(500),
  year      char,
  notes      char
)
```

miscellaneous.ctl

```
load data
infile 'miscellaneous.csv' "str '\n'"
insert
into table "HF"."IMDB_MISC"
fields terminated by '\t'
```

```
TRAILING NULLCOLS
(
  name char(500),
  title char(500),
  aka char(500),
  what char(600),
  uncredited integer external
)
```

producers.ctl

```
load data
infile 'producers.csv' "str '\n'"
insert
into table "HF"."IMDB_PRODUCERS"
fields terminated by '\t'
TRAILING NULLCOLS
(
  name char(500),
  title char(500),
  aka char(500),
  what char(500),
  uncredited integer external
)
```

production-companies.ctl

```
load data
infile 'production-companies.csv' "str '\n'"
insert
into table "HF"."IMDB_PRODCOMP"
fields terminated by '\t'
TRAILING NULLCOLS
(
  title char(500),
  country char(20),
  company char(500),
  aka char(500)
)
```

production-designers.ctl

```
load data
infile 'production-designers.csv' "str '\n'"
insert
into table "HF"."IMDB_PRODDESIGN"
fields terminated by '\t'
TRAILING NULLCOLS
(
  name char(500),
  title char(500),
  aka char(500),
  what char(500),
  uncredited integer external
)
```

ratings.ctl

```
load data
infile 'ratings.csv' "str '\n'"
insert
into table "HF"."IMDB_RATINGS"
fields terminated by '\t'
TRAILING NULLCOLS
(
  dist      CHAR(10),
  rank      FLOAT EXTERNAL ,
  votes     INTEGER EXTERNAL,
```

```
  title        CHAR(500)
)
```

running-times.ctl

```
load data
infile 'running-times.csv' "str '\n'"
insert
into table "HF"."IMDB_RUNNINGTIMES"
fields terminated by '\t'
TRAILING NULLCOLS
(
  title     char(500),
  country     CHAR,
  duration  integer external,
  notes  char(500)
)
```

special-effects-companies.ctl

```
load data
infile 'special-effects-companies.csv' "str '\n'"
insert
into table "HF"."IMDB_SFXCOMP"
fields terminated by '\t'
TRAILING NULLCOLS
(
  title char(500),
  country char(20),
  company char(500),
  aka char(500)
)
```

writers.ctl

```
load data
infile 'writers.csv' "str '\n'"
insert
into table "HF"."IMDB_WRITERS"
fields terminated by '\t'
TRAILING NULLCOLS
(
  name char(500),
  title char(500),
  aka char(500),
  what char(500),
  p1 integer external,
  p2 integer external,
  p3 integer external,
  uncredited integer external
)
```

B.2. Metadaten der Lerneingabe

Aus den Ketten in MiningMart konnten, wie in Abschnitt 7.1.3 erläutert, direkt Tabellen auf der relationalen Ebene mit Primär- und Fremdschlüsseln angelegt werden, die der Modellierung in MiningMart entsprechen und als direkte Eingabe für das relationale Lernverfahren RDT/DM anzugeben sind.

In diesem Abschnitt sind die Strukturen aller Tabellen mit Angabe der Primär- und Fremdschlüssel nachzulesen.

```
--
-- PostgreSQL database dump
```

```
--

SET client_encoding = 'UTF8';
SET check_function_bodies = false;
SET client_min_messages = warning;

--
-- TOC entry 1805 (class 0 OID 0)
-- Dependencies: 4
-- Name: SCHEMA public; Type: COMMENT; Schema: -; Owner: postgres
--

COMMENT ON SCHEMA public IS 'Standard public schema';


SET search_path = public, pg_catalog;

--
-- TOC entry 13 (class 1255 OID 16403)
-- Dependencies: 4
-- Name: autoinc(); Type: FUNCTION; Schema: public; Owner: postgres
--

CREATE FUNCTION autoinc() RETURNS "trigger"
    AS '$libdir/autoinc', 'autoinc'
    LANGUAGE c;


ALTER FUNCTION public.autoinc() OWNER TO postgres;

SET default_tablespace = '';

SET default_with_oids = false;

--
-- TOC entry 1306 (class 1259 OID 31626)
-- Dependencies: 4
-- Name: rdtcrosscinemat; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE rdtcrosscinemat (
    id_title character varying,
    id_person character varying,
    id_aka character varying,
    uncredited character varying
);


ALTER TABLE public.rdtcrosscinemat OWNER TO postgres;

--
-- TOC entry 1309 (class 1259 OID 44543)
-- Dependencies: 4
-- Name: rdtcrosscountries; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE rdtcrosscountries (
    id_title character varying,
    id_country character varying
);


ALTER TABLE public.rdtcrosscountries OWNER TO postgres;

--
-- TOC entry 1301 (class 1259 OID 27307)
```

```
-- Dependencies: 4
-- Name: rdtcrossgenres; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE rdtcrossgenres (
    id_genre character varying(500) NOT NULL,
    id_title character varying NOT NULL
);


ALTER TABLE public.rdtcrossgenres OWNER TO postgres;

--
-- TOC entry 1312 (class 1259 OID 46463)
-- Dependencies: 4
-- Name: rdtpersonentop; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE rdtpersonentop (
    id_person character varying NOT NULL,
    top character varying
);

ALTER TABLE public.rdtpersonentop OWNER TO postgres;

--
-- TOC entry 1300 (class 1259 OID 27302)
-- Dependencies: 4
-- Name: rdttitles; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE rdttitles (
    titles character varying(500),
    id_title character varying NOT NULL
);


ALTER TABLE public.rdttitles OWNER TO postgres;


CREATE TABLE rdtcountries (
    country character varying(500),
    id_country character varying NOT NULL
);


ALTER TABLE public.rdtcountries OWNER TO postgres;

--
-- TOC entry 1302 (class 1259 OID 27309)
-- Dependencies: 4
-- Name: rdtcrosskeywords; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE rdtcrosskeywords (
    id_keyword character varying NOT NULL,
    id_title character varying NOT NULL
);


ALTER TABLE public.rdtcrosskeywords OWNER TO postgres;

--
-- TOC entry 1303 (class 1259 OID 27316)
-- Dependencies: 4
-- Name: rdtgenres; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
```

```
--

CREATE TABLE rdtgenres (
    genre character varying(100) NOT NULL,
    id_genre character varying NOT NULL
);


ALTER TABLE public.rdtgenres OWNER TO postgres;

--
-- TOC entry 1304 (class 1259 OID 27318)
-- Dependencies: 4
-- Name: rdtkeywords; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE rdtkeywords (
    keyword character varying(500),
    id_keyword character varying NOT NULL
);


ALTER TABLE public.rdtkeywords OWNER TO postgres;

--
-- TOC entry 1311 (class 1259 OID 45322)
-- Dependencies: 4
-- Name: rdtpersonen; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE rdtpersonen (
    name character varying(1000),
    id_person character varying NOT NULL
);


ALTER TABLE public.rdtpersonen OWNER TO postgres;

--
-- TOC entry 1791 (class 2606 OID 44565)
-- Dependencies: 1310 1310
-- Name: rdtcountries_pk; Type: CONSTRAINT; Schema: public; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY rdtcountries
    ADD CONSTRAINT rdtcountries_pk PRIMARY KEY (id_country);


--
-- TOC entry 1780 (class 2606 OID 30130)
-- Dependencies: 1303 1303
-- Name: rdtgenres_pk; Type: CONSTRAINT; Schema: public; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY rdtgenres
    ADD CONSTRAINT rdtgenres_pk PRIMARY KEY (id_genre);


--
-- TOC entry 1782 (class 2606 OID 30132)
-- Dependencies: 1304 1304
-- Name: rdtkeywords_pk; Type: CONSTRAINT; Schema: public; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY rdtkeywords
    ADD CONSTRAINT rdtkeywords_pk PRIMARY KEY (id_keyword);
```

```
--
-- TOC entry 1793 (class 2606 OID 45328)
-- Dependencies: 1311 1311
-- Name: rdtpersonen_pk; Type: CONSTRAINT; Schema: public; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY rdtpersonen
    ADD CONSTRAINT rdtpersonen_pk PRIMARY KEY (id_person);


--
-- TOC entry 1795 (class 2606 OID 46474)
-- Dependencies: 1312 1312
-- Name: rdtpersonentop_pk; Type: CONSTRAINT; Schema: public; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY rdtpersonentop
    ADD CONSTRAINT rdtpersonentop_pk PRIMARY KEY (id_person);

--
-- TOC entry 1775 (class 2606 OID 30136)
-- Dependencies: 1300 1300
-- Name: rdttitles_pk; Type: CONSTRAINT; Schema: public; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY rdttitles
    ADD CONSTRAINT rdttitles_pk PRIMARY KEY (id_title);


--
-- TOC entry 1785 (class 1259 OID 31668)
-- Dependencies: 1306
-- Name: fki_rdtcrosscinemat_fk_aka; Type: INDEX; Schema: public; Owner: postgres; Tablespace:
--

CREATE INDEX fki_rdtcrosscinemat_fk_aka ON rdtcrosscinemat USING btree (id_aka);


--
-- TOC entry 1786 (class 1259 OID 44281)
-- Dependencies: 1306
-- Name: fki_rdtcrosscinemat_fk_name; Type: INDEX; Schema: public; Owner: postgres; Tablespace:
--

CREATE INDEX fki_rdtcrosscinemat_fk_name ON rdtcrosscinemat USING btree (id_person);


--
-- TOC entry 1787 (class 1259 OID 31662)
-- Dependencies: 1306
-- Name: fki_rdtcrosscinemat_fk_person; Type: INDEX; Schema: public; Owner: postgres; Tablespace:
--

CREATE INDEX fki_rdtcrosscinemat_fk_person ON rdtcrosscinemat USING btree (id_person);


--
-- TOC entry 1788 (class 1259 OID 31656)
-- Dependencies: 1306
-- Name: fki_rdtcrosscinemat_fk_title; Type: INDEX; Schema: public; Owner: postgres; Tablespace:
--

CREATE INDEX fki_rdtcrosscinemat_fk_title ON rdtcrosscinemat USING btree (id_title);
```

```
--
-- TOC entry 1789 (class 1259 OID 44563)
-- Dependencies: 1309
-- Name: fki_rdtcrosscountries_fk_title; Type: INDEX; Schema: public; Owner: postgres; Tablespace:
--

CREATE INDEX fki_rdtcrosscountries_fk_title ON rdtcrosscountries USING btree (id_title);


--
-- TOC entry 1776 (class 1259 OID 30577)
-- Dependencies: 1301
-- Name: fki_rdtcrossgenres_fk_genre; Type: INDEX; Schema: public; Owner: postgres; Tablespace:
--

CREATE INDEX fki_rdtcrossgenres_fk_genre ON rdtcrossgenres USING btree (id_genre);


--
-- TOC entry 1777 (class 1259 OID 30566)
-- Dependencies: 1301
-- Name: fki_rdtcrossgenres_fk_titles; Type: INDEX; Schema: public; Owner: postgres; Tablespace:
--

CREATE INDEX fki_rdtcrossgenres_fk_titles ON rdtcrossgenres USING btree (id_title);


--
-- TOC entry 1778 (class 1259 OID 43194)
-- Dependencies: 1302
-- Name: fki_rdtcrosskeyw_fk_title; Type: INDEX; Schema: public; Owner: postgres; Tablespace:
--

CREATE INDEX fki_rdtcrosskeyw_fk_title ON rdtcrosskeywords USING btree (id_title);


--
-- TOC entry 1773 (class 1259 OID 31499)
-- Dependencies: 1299
-- Name: fki_rdtranking_fk_title; Type: INDEX; Schema: public; Owner: postgres; Tablespace:
--

CREATE INDEX fki_rdtranking_fk_title ON rdtranking USING btree (id_title);

--
-- TOC entry 1802 (class 2606 OID 31651)
-- Dependencies: 1774 1300 1306
-- Name: rdtcrosscinemat_fk_title; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY rdtcrosscinemat
    ADD CONSTRAINT rdtcrosscinemat_fk_title FOREIGN KEY (id_title) REFERENCES rdttitles(id_title);


--
-- TOC entry 1803 (class 2606 OID 44558)
-- Dependencies: 1774 1300 1309
-- Name: rdtcrosscountries_fk_title; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY rdtcrosscountries
    ADD CONSTRAINT rdtcrosscountries_fk_title FOREIGN KEY (id_title) REFERENCES rdttitles(id_title);


--
```

```
-- TOC entry 1800 (class 2606 OID 30561)
-- Dependencies: 1774 1300 1301
-- Name: rdtcrossgenres_fk_titles; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY rdtcrossgenres
    ADD CONSTRAINT rdtcrossgenres_fk_titles FOREIGN KEY (id_title) REFERENCES rdttitles(id_title);


--
-- TOC entry 1801 (class 2606 OID 43189)
-- Dependencies: 1774 1300 1302
-- Name: rdtcrosskeyw_fk_title; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY rdtcrosskeywords
    ADD CONSTRAINT rdtcrosskeyw_fk_title FOREIGN KEY (id_title) REFERENCES rdttitles(id_title);


--
-- TOC entry 1799 (class 2606 OID 31494)
-- Dependencies: 1774 1300 1299
-- Name: rdtranking_fk_title; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY rdtranking
    ADD CONSTRAINT rdtranking_fk_title FOREIGN KEY (id_title) REFERENCES rdttitles(id_title);

--
-- TOC entry 1806 (class 0 OID 0)
-- Dependencies: 4
-- Name: public; Type: ACL; Schema: -; Owner: postgres
--

REVOKE ALL ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON SCHEMA public FROM postgres;
GRANT ALL ON SCHEMA public TO postgres;
GRANT ALL ON SCHEMA public TO PUBLIC;


--
-- PostgreSQL database dump complete
--
```

C. Details zum Aufbau der Experimente mit MiningMart und RDT/DM

C.1. Beispiel für Ausgabe des MiningMart Compilers

Dies ist die Ausgabe bei der Compilierung der Kette zur Erstellung der Entitätsklasse mit den Firmenbezeichnern. Jeder Schritt der Kette wird kommmentiert und die benötigte Zeit wird ausführlich dokumentiert. Dazu werden die SQL-Anweisungen ausgegeben, die auf der relationalen Ebene ausgeführt werden.

```
29.07.2006 13:36:56: Start compiling ...
29.07.2006 13:36:56: Print-verbosity set to: Operator
29.07.2006 13:36:56:
--- 29.07.2006 13:36:56 ---
29.07.2006 13:36:56: Trying to execute Step 'MiscComp_Select_Company' (ID: 100104481) in lazy mode ...
29.07.2006 13:36:56: 29.07.2006 13:36:56 Waiting for resource to run garbage collection for step 'MiscComp_Select_Company'.
29.07.2006 13:36:56: 29.07.2006 13:36:56 Running garbage collection for step 'MiscComp_Select_Company'.
29.07.2006 13:36:56: 29.07.2006 13:36:56 : Preparing execution of step 'MiscComp_Select_Company', checking for MultiSteps...
29.07.2006 13:36:56: Waiting for resources to compile step 100104481.
29.07.2006 13:36:56: Executing step 'MiscComp_Select_Company' (ID: 100104481).
29.07.2006 13:36:56: FeatureSelectionByAttributes generated: VIEW CS_100104481 AS
                     (select BDUSER.IMDB_MISCCOMP.COMPANY from BDUSER.IMDB_MISCCOMP)
29.07.2006 13:36:56: Compilation of step 'MiscComp_Select_Company' done.
29.07.2006 13:36:56:
--- 29.07.2006 13:36:56 ---
29.07.2006 13:36:56: Execution for Step MiscComp_Select_Company without errors
29.07.2006 13:36:56: Mode was set to lazy
29.07.2006 13:36:56:
--- 29.07.2006 13:36:56 ---
29.07.2006 13:36:56: Trying to execute Step 'Distributors_Select_Company' (ID: 100104482) in lazy mode ...
29.07.2006 13:36:56: 29.07.2006 13:36:56 Waiting for resource to run garbage collection for step 'Distributors_Select_Company'.
29.07.2006 13:36:56: 29.07.2006 13:36:56 Running garbage collection for step 'Distributors_Select_Company'.
29.07.2006 13:36:56: 29.07.2006 13:36:56 : Preparing execution of step 'Distributors_Select_Company', checking for MultiSteps...
29.07.2006 13:36:56: Waiting for resources to compile step 100104482.
29.07.2006 13:36:56: Executing step 'Distributors_Select_Company' (ID: 100104482).
29.07.2006 13:36:56: FeatureSelectionByAttributes generated: VIEW CS_100104482 AS
                     (select BDUSER.IMDB_DISTRIBUTORS.COMPANY from BDUSER.IMDB_DISTRIBUTORS)
29.07.2006 13:36:56: Compilation of step 'Distributors_Select_Company' done.
29.07.2006 13:36:56:
--- 29.07.2006 13:36:56 ---
29.07.2006 13:36:56: Execution for Step Distributors_Select_Company without errors
29.07.2006 13:36:56: Mode was set to lazy
29.07.2006 13:36:56:
--- 29.07.2006 13:36:56 ---
29.07.2006 13:36:56: Trying to execute Step 'ProdComp_Select_Company' (ID: 100104497) in lazy mode ...
29.07.2006 13:36:56: 29.07.2006 13:36:56 Waiting for resource to run garbage collection for step 'ProdComp_Select_Company'.
29.07.2006 13:36:56: 29.07.2006 13:36:56 Running garbage collection for step 'ProdComp_Select_Company'.
29.07.2006 13:36:56: 29.07.2006 13:36:56 : Preparing execution of step 'ProdComp_Select_Company', checking for MultiSteps...
29.07.2006 13:36:56: Waiting for resources to compile step 100104497.
29.07.2006 13:36:56: Executing step 'ProdComp_Select_Company' (ID: 100104497).
29.07.2006 13:36:56: FeatureSelectionByAttributes generated: VIEW CS_100104497 AS
                     (select BDUSER.IMDB_PRODCOMP.COMPANY from BDUSER.IMDB_PRODCOMP)
29.07.2006 13:36:56: Compilation of step 'ProdComp_Select_Company' done.
29.07.2006 13:36:56:
--- 29.07.2006 13:36:56 ---
29.07.2006 13:36:56: Execution for Step ProdComp_Select_Company without errors
29.07.2006 13:36:56: Mode was set to lazy
29.07.2006 13:36:56:
--- 29.07.2006 13:36:56 ---
29.07.2006 13:36:56: Trying to execute Step 'SfxComp_Select_Company' (ID: 100104520) in lazy mode ...
29.07.2006 13:36:56: 29.07.2006 13:36:56 Waiting for resource to run garbage collection for step 'SfxComp_Select_Company'.
29.07.2006 13:36:56: 29.07.2006 13:36:56 Running garbage collection for step 'SfxComp_Select_Company'.
29.07.2006 13:36:56: 29.07.2006 13:36:56 : Preparing execution of step 'SfxComp_Select_Company', checking for MultiSteps...
29.07.2006 13:36:56: Waiting for resources to compile step 100104520.
29.07.2006 13:36:56: Executing step 'SfxComp_Select_Company' (ID: 100104520).
29.07.2006 13:36:56: FeatureSelectionByAttributes generated: VIEW CS_100104520 AS
                     (select BDUSER.IMDB_SFXCOMP.COMPANY from BDUSER.IMDB_SFXCOMP)
29.07.2006 13:36:57: Compilation of step 'SfxComp_Select_Company' done.
29.07.2006 13:36:57:
--- 29.07.2006 13:36:57 ---
29.07.2006 13:36:57: Execution for Step SfxComp_Select_Company without errors
29.07.2006 13:36:57: Mode was set to lazy
29.07.2006 13:36:57:
```

```
--- 29.07.2006 13:36:57 ---
29.07.2006 13:36:57: Trying to execute Step 'Union' (ID: 100104548) in lazy mode ...
29.07.2006 13:36:57: 29.07.2006 13:36:57 Waiting for resource to run garbage collection for step 'Union'.
29.07.2006 13:36:57: 29.07.2006 13:36:57 Running garbage collection for step 'Union'.
29.07.2006 13:36:57: 29.07.2006 13:36:57 : Preparing execution of step 'Union', checking for MultiSteps...
29.07.2006 13:36:57: Waiting for resources to compile step 100104548.
29.07.2006 13:36:57: Executing step 'Union' (ID: 100104548).
29.07.2006 13:36:57: Union generated: VIEW CS_100104548 AS
                     ((SELECT COMPANY FROM BDUSER.CS_100104482) UNION (SELECT COMPANY FROM BDUSER.CS_100104481)
                     UNION (SELECT COMPANY FROM BDUSER.CS_100104497)
                     UNION (SELECT COMPANY FROM BDUSER.CS_100104520))
29.07.2006 13:36:57: Compilation of step 'Union' done.
29.07.2006 13:36:57:
--- 29.07.2006 13:36:57 ---
29.07.2006 13:36:57: Execution for Step Union without errors
29.07.2006 13:36:57: Mode was set to lazy
29.07.2006 13:36:57:
--- 29.07.2006 13:36:57 ---
29.07.2006 13:36:57: Trying to execute Step 'Assign Unique Id' (ID: 100104942) in lazy mode ...
29.07.2006 13:36:57: 29.07.2006 13:36:57 Waiting for resource to run garbage collection for step 'Assign Unique Id'.
29.07.2006 13:36:57: 29.07.2006 13:36:57 Running garbage collection for step 'Assign Unique Id'.
29.07.2006 13:36:57: 29.07.2006 13:36:57 : Preparing execution of step 'Assign Unique Id', checking for MultiSteps...
29.07.2006 13:36:57: Waiting for resources to compile step 100104942.
29.07.2006 13:36:57: Executing step 'Assign Unique Id' (ID: 100104942).
29.07.2006 13:36:57: CreatePrimaryKey generated: VIEW CS_100104942 AS
                     (SELECT DISTINCT BDUSER.CS_100104548.COMPANY FROM BDUSER.CS_100104548)
29.07.2006 13:36:57: Compilation of step 'Assign Unique Id' done.
29.07.2006 13:36:57:
--- 29.07.2006 13:36:57 ---
29.07.2006 13:36:57: Execution for Step Assign Unique Id without errors
29.07.2006 13:36:57: Mode was set to lazy
29.07.2006 13:36:57:
--- 29.07.2006 13:36:57 ---
29.07.2006 13:36:57: Trying to execute Step 'Materialize Concept' (ID: 100181881) in lazy mode ...
29.07.2006 13:36:57: 29.07.2006 13:36:57 Waiting for resource to run garbage collection for step 'Materialize Concept'.
29.07.2006 13:36:57: 29.07.2006 13:36:57 Running garbage collection for step 'Materialize Concept'.
29.07.2006 13:36:57: 29.07.2006 13:36:57 : Preparing execution of step 'Materialize Concept', checking for MultiSteps...
29.07.2006 13:36:57: Waiting for resources to compile step 100181881.
29.07.2006 13:36:57: Executing step 'Materialize Concept' (ID: 100181881).
29.07.2006 13:37:07: Materialize generated: TABLE mm_companies_id AS BDUSER.mm_companies_id
29.07.2006 13:37:07: Compilation of step 'Materialize Concept' done.
29.07.2006 13:37:07:
--- 29.07.2006 13:37:07 ---
29.07.2006 13:37:07: Execution for Step Materialize Concept without errors
29.07.2006 13:37:07: Mode was set to lazy
29.07.2006 13:37:07: Compiling done.
```

C.2. Versuch 1: Benutzergruppe Kameraleute, ohne Schlüsselwörter

```
(DBMapper): Mappings(68): START
         ( Table/View ..................... mapto Literal.....................................)
DBMapping( rdtcrosscinemat..................... mapto rdtcrosscinemat(id_title,id_person,id_aka,uncredited) )
DBMapping( rdt_rdtcrosscinemat_id_person... mapto rdt_rdtcrosscinemat_id_person(id_title,id_person) )
DBMapping( rdt_rdtcrosscinemat_id_aka..... mapto rdt_rdtcrosscinemat_id_aka(id_title,id_aka)..... )
DBMapping( rdt_rdtcrosscinemat_uncredited.. mapto rdt_rdtcrosscinemat_uncredited(id_title,uncredited) )
DBMapping( rdt_rdtcrosscinemat_uncredited_1 mapto rdt_rdtcrosscinemat_uncredited_10(id_title)..... )
DBMapping( rdt_rdtcrosscinemat_uncredited_2 mapto rdt_rdtcrosscinemat_uncredited_21(id_title)..... )
DBMapping( rdtcrosscountries..... mapto rdtcrosscountries(id_title,id_country)..... )
DBMapping( rdt_rdtcrosscountries_id_country mapto rdt_rdtcrosscountries_id_country(id_title,id_country) )
DBMapping( rdt_rdtcrosscountries_id_country_1 mapto rdt_rdtcrosscountries_id_country_1C2(id_title)... )
DBMapping( rdt_rdtcrosscountries_id_country_2 mapto rdt_rdtcrosscountries_id_country_2C128(id_title). )
DBMapping( rdt_rdtcrosscountries_id_country_3 mapto rdt_rdtcrosscountries_id_country_3C5(id_title)... )
DBMapping( rdt_rdtcrosscountries_id_country_4 mapto rdt_rdtcrosscountries_id_country_4C43(id_title).. )
DBMapping( rdt_rdtcrosscountries_id_country_5 mapto rdt_rdtcrosscountries_id_country_5C15(id_title).. )
DBMapping( rdt_rdtcrosscountries_id_country_6 mapto rdt_rdtcrosscountries_id_country_6C9(id_title)... )
DBMapping( rdt_rdtcrosscountries_id_country_7 mapto rdt_rdtcrosscountries_id_country_7C161(id_title). )
DBMapping( rdt_rdtcrosscountries_id_country_8 mapto rdt_rdtcrosscountries_id_country_8C41(id_title).. )
DBMapping( rdt_rdtcrosscountries_id_country_9 mapto rdt_rdtcrosscountries_id_country_9C56(id_title).. )
DBMapping( rdt_rdtcrosscountries_id_country_10 mapto rdt_rdtcrosscountries_id_country_10C173(id_title) )
DBMapping( rdt_rdtcrosscountries_id_country_11 mapto rdt_rdtcrosscountries_id_country_11C160(id_title) )
DBMapping( rdt_rdtcrosscountries_id_country_12 mapto rdt_rdtcrosscountries_id_country_12C67(id_title). )
DBMapping( rdt_rdtcrosscountries_id_country_13 mapto rdt_rdtcrosscountries_id_country_13C177(id_title) )
DBMapping( rdt_rdtcrosscountries_id_country_14 mapto rdt_rdtcrosscountries_id_country_14C80(id_title). )
DBMapping( rdt_rdtcrosscountries_id_country_15 mapto rdt_rdtcrosscountries_id_country_15C68(id_title). )
DBMapping( rdt_rdtcrosscountries_id_country_16 mapto rdt_rdtcrosscountries_id_country_16C112(id_title) )
DBMapping( rdt_rdtcrosscountries_id_country_17 mapto rdt_rdtcrosscountries_id_country_17C142(id_title) )
DBMapping( rdt_rdtcrosscountries_id_country_18 mapto rdt_rdtcrosscountries_id_country_18C79(id_title). )
DBMapping( rdt_rdtcrosscountries_id_country_19 mapto rdt_rdtcrosscountries_id_country_19C115(id_title) )
DBMapping( rdt_rdtcrosscountries_id_country_20 mapto rdt_rdtcrosscountries_id_country_20C84(id_title). )
DBMapping( rdt_rdtcrosscountries_id_country_21 mapto rdt_rdtcrosscountries_id_country_21C13(id_title). )
DBMapping( rdt_rdtcrosscountries_id_country_22 mapto rdt_rdtcrosscountries_id_country_22C146(id_title) )
DBMapping( rdtcrossgenres..... mapto rdtcrossgenres(id_genre,id_title)..... )
DBMapping( rdt_rdtcrossgenres_id_genre..... mapto rdt_rdtcrossgenres_id_genre(id_title,id_genre)... )
DBMapping( rdt_rdtcrossgenres_id_genre_1... mapto rdt_rdtcrossgenres_id_genre_1G15(id_title)..... )
DBMapping( rdt_rdtcrossgenres_id_genre_2... mapto rdt_rdtcrossgenres_id_genre_2G9(id_title)..... )
```

```
      DBMapping( rdt_rdtcrossgenres_id_genre_3... mapto rdt_rdtcrossgenres_id_genre_3G6(id_title)........ )
      DBMapping( rdt_rdtcrossgenres_id_genre_4... mapto rdt_rdtcrossgenres_id_genre_4G2(id_title)........ )
      DBMapping( rdt_rdtcrossgenres_id_genre_5... mapto rdt_rdtcrossgenres_id_genre_5G20(id_title)....... )
 40   DBMapping( rdt_rdtcrossgenres_id_genre_6... mapto rdt_rdtcrossgenres_id_genre_6G19(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_7... mapto rdt_rdtcrossgenres_id_genre_7G16(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_8... mapto rdt_rdtcrossgenres_id_genre_8G23(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_9... mapto rdt_rdtcrossgenres_id_genre_9G12(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_10.. mapto rdt_rdtcrossgenres_id_genre_10G27(id_title)...... )
 45   DBMapping( rdt_rdtcrossgenres_id_genre_11.. mapto rdt_rdtcrossgenres_id_genre_11G3(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_12.. mapto rdt_rdtcrossgenres_id_genre_12G8(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_13.. mapto rdt_rdtcrossgenres_id_genre_13G4(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_14.. mapto rdt_rdtcrossgenres_id_genre_14G11(id_title)...... )
      DBMapping( rdt_rdtcrossgenres_id_genre_15.. mapto rdt_rdtcrossgenres_id_genre_15G24(id_title)...... )
 50   DBMapping( rdt_rdtcrossgenres_id_genre_16.. mapto rdt_rdtcrossgenres_id_genre_16G22(id_title)...... )
      DBMapping( rdt_rdtcrossgenres_id_genre_17.. mapto rdt_rdtcrossgenres_id_genre_17G5(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_18.. mapto rdt_rdtcrossgenres_id_genre_18G28(id_title)...... )
      DBMapping( rdt_rdtcrossgenres_id_genre_19.. mapto rdt_rdtcrossgenres_id_genre_19G25(id_title)...... )
      DBMapping( rdt_rdtcrossgenres_id_genre_20.. mapto rdt_rdtcrossgenres_id_genre_20G14(id_title)...... )
 55   DBMapping( rdt_rdtcrossgenres_id_genre_21.. mapto rdt_rdtcrossgenres_id_genre_21G17(id_title)...... )
      DBMapping( rdt_rdtcrossgenres_id_genre_22.. mapto rdt_rdtcrossgenres_id_genre_22G18(id_title)...... )
      DBMapping( rdt_rdtcrossgenres_id_genre_23.. mapto rdt_rdtcrossgenres_id_genre_23G1(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_24.. mapto rdt_rdtcrossgenres_id_genre_24G21(id_title)...... )
      DBMapping( rdtpersonentop................... mapto rdtpersonentop(id_person,top)..................... )
 60   DBMapping( rdt_rdtpersonentop_top.......... mapto rdt_rdtpersonentop_top(id_person,top)............ )
      DBMapping( rdt_rdtpersonentop_top_1........ mapto rdt_rdtpersonentop_top_10(id_person)............. )
      DBMapping( rdt_rdtpersonentop_top_2........ mapto rdt_rdtpersonentop_top_21(id_person)............. )
      DBMapping( rdtranking....................... mapto rdtranking(id_title,wrank,rank,top)............... )
      DBMapping( rdt_rdtranking_wrank............ mapto rdt_rdtranking_wrank(id_title,wrank)............. )
 65   DBMapping( rdt_rdtranking_rank............. mapto rdt_rdtranking_rank(id_title,rank)............... )
      DBMapping( rdt_rdtranking_top.............. mapto rdt_rdtranking_top(id_title,top)................. )
      DBMapping( rdt_rdtranking_top_1............ mapto rdt_rdtranking_top_10(id_title).................. )
      DBMapping( rdt_rdtranking_top_2............ mapto rdt_rdtranking_top_21(id_title).................. )
      DBMapping( rdttitles........................ mapto rdttitles(titles,id_title)....................... )
 70   DBMapping( rdt_rdttitles_titles............ mapto rdt_rdttitles_titles(id_title,titles)............ )
      Mappings(68): STOP
```

C.3. Versuch 2: Hinzunahme der Schlüsselwörter

```
      (DBMapper): Mappings(3750): START
              ( Table/View ..................... mapto Literal.......................................)
      DBMapping( rdtcrosscinemat................. mapto rdtcrosscinemat(id_title,id_person,id_aka,uncredited) )
      DBMapping( rdt_rdtcrosscinemat_id_person... mapto rdt_rdtcrosscinemat_id_person(id_title,id_person) )
  5   DBMapping( rdt_rdtcrosscinemat_id_aka...... mapto rdt_rdtcrosscinemat_id_aka(id_title,id_aka)...... )
      DBMapping( rdt_rdtcrosscinemat_uncredited.. mapto rdt_rdtcrosscinemat_uncredited(id_title,uncredited) )
      DBMapping( rdt_rdtcrosscinemat_uncredited_1 mapto rdt_rdtcrosscinemat_uncredited_10(id_title)...... )
      DBMapping( rdt_rdtcrosscinemat_uncredited_2 mapto rdt_rdtcrosscinemat_uncredited_21(id_title)...... )
      DBMapping( rdtcrosscountries............... mapto rdtcrosscountries(id_title,id_country)........... )
 10   DBMapping( rdt_rdtcrosscountries_id_country mapto rdt_rdtcrosscountries_id_country(id_title,id_country) )
      DBMapping( rdt_rdtcrosscountries_id_country_1 mapto rdt_rdtcrosscountries_id_country_1C2(id_title)... )
      DBMapping( rdt_rdtcrosscountries_id_country_2 mapto rdt_rdtcrosscountries_id_country_2C128(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_3 mapto rdt_rdtcrosscountries_id_country_3C5(id_title)... )
      DBMapping( rdt_rdtcrosscountries_id_country_4 mapto rdt_rdtcrosscountries_id_country_4C43(id_title).. )
 15   DBMapping( rdt_rdtcrosscountries_id_country_5 mapto rdt_rdtcrosscountries_id_country_5C15(id_title).. )
      DBMapping( rdt_rdtcrosscountries_id_country_6 mapto rdt_rdtcrosscountries_id_country_6C9(id_title)... )
      DBMapping( rdt_rdtcrosscountries_id_country_7 mapto rdt_rdtcrosscountries_id_country_7C161(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_8 mapto rdt_rdtcrosscountries_id_country_8C41(id_title).. )
      DBMapping( rdt_rdtcrosscountries_id_country_9 mapto rdt_rdtcrosscountries_id_country_9C56(id_title).. )
 20   DBMapping( rdt_rdtcrosscountries_id_country_10 mapto rdt_rdtcrosscountries_id_country_10C173(id_title) )
      DBMapping( rdt_rdtcrosscountries_id_country_11 mapto rdt_rdtcrosscountries_id_country_11C160(id_title) )
      DBMapping( rdt_rdtcrosscountries_id_country_12 mapto rdt_rdtcrosscountries_id_country_12C67(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_13 mapto rdt_rdtcrosscountries_id_country_13C177(id_title) )
      DBMapping( rdt_rdtcrosscountries_id_country_14 mapto rdt_rdtcrosscountries_id_country_14C80(id_title). )
 25   DBMapping( rdt_rdtcrosscountries_id_country_15 mapto rdt_rdtcrosscountries_id_country_15C68(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_16 mapto rdt_rdtcrosscountries_id_country_16C112(id_title) )
      DBMapping( rdt_rdtcrosscountries_id_country_17 mapto rdt_rdtcrosscountries_id_country_17C142(id_title) )
      DBMapping( rdt_rdtcrosscountries_id_country_18 mapto rdt_rdtcrosscountries_id_country_18C79(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_19 mapto rdt_rdtcrosscountries_id_country_19C115(id_title) )
 30   DBMapping( rdt_rdtcrosscountries_id_country_20 mapto rdt_rdtcrosscountries_id_country_20C84(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_21 mapto rdt_rdtcrosscountries_id_country_21C13(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_22 mapto rdt_rdtcrosscountries_id_country_22C146(id_title) )
      DBMapping( rdtcrossgenres.................. mapto rdtcrossgenres(id_genre,id_title)................. )
      DBMapping( rdt_rdtcrossgenres_id_genre..... mapto rdt_rdtcrossgenres_id_genre(id_title,id_genre)... )
 35   DBMapping( rdt_rdtcrossgenres_id_genre_1... mapto rdt_rdtcrossgenres_id_genre_1G15(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_2... mapto rdt_rdtcrossgenres_id_genre_2G9(id_title)........ )
      DBMapping( rdt_rdtcrossgenres_id_genre_3... mapto rdt_rdtcrossgenres_id_genre_3G6(id_title)........ )
      DBMapping( rdt_rdtcrossgenres_id_genre_4... mapto rdt_rdtcrossgenres_id_genre_4G2(id_title)........ )
      DBMapping( rdt_rdtcrossgenres_id_genre_5... mapto rdt_rdtcrossgenres_id_genre_5G20(id_title)....... )
 40   DBMapping( rdt_rdtcrossgenres_id_genre_6... mapto rdt_rdtcrossgenres_id_genre_6G19(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_7... mapto rdt_rdtcrossgenres_id_genre_7G16(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_8... mapto rdt_rdtcrossgenres_id_genre_8G23(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_9... mapto rdt_rdtcrossgenres_id_genre_9G12(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_10.. mapto rdt_rdtcrossgenres_id_genre_10G27(id_title)...... )
 45   DBMapping( rdt_rdtcrossgenres_id_genre_11.. mapto rdt_rdtcrossgenres_id_genre_11G3(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_12.. mapto rdt_rdtcrossgenres_id_genre_12G8(id_title)....... )
      DBMapping( rdt_rdtcrossgenres_id_genre_13.. mapto rdt_rdtcrossgenres_id_genre_13G4(id_title)....... )
```

```
      DBMapping( rdt_rdtcrossgenres_id_genre_14.. mapto rdt_rdtcrossgenres_id_genre_14G11(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_15.. mapto rdt_rdtcrossgenres_id_genre_15G24(id_title)..... )
 50   DBMapping( rdt_rdtcrossgenres_id_genre_16.. mapto rdt_rdtcrossgenres_id_genre_16G22(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_17.. mapto rdt_rdtcrossgenres_id_genre_17G5(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_18.. mapto rdt_rdtcrossgenres_id_genre_18G28(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_19.. mapto rdt_rdtcrossgenres_id_genre_19G25(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_20.. mapto rdt_rdtcrossgenres_id_genre_20G14(id_title)..... )
 55   DBMapping( rdt_rdtcrossgenres_id_genre_21.. mapto rdt_rdtcrossgenres_id_genre_21G17(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_22.. mapto rdt_rdtcrossgenres_id_genre_22G18(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_23.. mapto rdt_rdtcrossgenres_id_genre_23G1(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_24.. mapto rdt_rdtcrossgenres_id_genre_24G21(id_title)..... )
      DBMapping( rdtcrosskeywords................ mapto rdtcrosskeywords(id_keyword,id_title)............ )
 60   DBMapping( rdt_rdtcrosskeywords_id_keyword. mapto rdt_rdtcrosskeywords_id_keyword(id_title,id_keyword) )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_1 mapto rdt_rdtcrosskeywords_id_keyword_14210(id_title).. )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_2 mapto rdt_rdtcrosskeywords_id_keyword_22152(id_title).. )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_3 mapto rdt_rdtcrosskeywords_id_keyword_322495(id_title). )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_4 mapto rdt_rdtcrosskeywords_id_keyword_44514(id_title).. )
 65   DBMapping( rdt_rdtcrosskeywords_id_keyword_5 mapto rdt_rdtcrosskeywords_id_keyword_54637(id_title).. )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_6 mapto rdt_rdtcrosskeywords_id_keyword_65847(id_title).. )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_7 mapto rdt_rdtcrosskeywords_id_keyword_722094(id_title). )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_8 mapto rdt_rdtcrosskeywords_id_keyword_84381(id_title).. )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_9 mapto rdt_rdtcrosskeywords_id_keyword_923364(id_title). )
 70   DBMapping( rdt_rdtcrosskeywords_id_keyword_10 mapto rdt_rdtcrosskeywords_id_keyword_106137(id_title). )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_11 mapto rdt_rdtcrosskeywords_id_keyword_1117062(id_title) )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_12 mapto rdt_rdtcrosskeywords_id_keyword_12266(id_title).. )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_13 mapto rdt_rdtcrosskeywords_id_keyword_136423(id_title). )

 75   [...] [...] [...] [...] [...] [...] [...] [...] [...] [...] [...] [...] [...] [...] [...] [...] [...]

      DBMapping( rdt_rdtcrosskeywords_id_keyword_3676 mapto rdt_rdtcrosskeywords_id_keyword_367630221(id_title) )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_3677 mapto rdt_rdtcrosskeywords_id_keyword_367718520(id_title) )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_3678 mapto rdt_rdtcrosskeywords_id_keyword_367814671(id_title) )
 80   DBMapping( rdt_rdtcrosskeywords_id_keyword_3679 mapto rdt_rdtcrosskeywords_id_keyword_367926907(id_title) )
      DBMapping( rdt_rdtcrosskeywords_id_keyword_3680 mapto rdt_rdtcrosskeywords_id_keyword_368017085(id_title) )
      DBMapping( rdtpersonentop.................. mapto rdtpersonentop(id_person,top)..................... )
      DBMapping( rdt_rdtpersonentop_top.......... mapto rdt_rdtpersonentop_top(id_person,top)............. )
      DBMapping( rdt_rdtpersonentop_top_1........ mapto rdt_rdtpersonentop_top_10(id_person).............. )
 85   DBMapping( rdt_rdtpersonentop_top_2........ mapto rdt_rdtpersonentop_top_21(id_person).............. )
      DBMapping( rdtranking...................... mapto rdtranking(id_title,wrank,rank,top)............... )
      DBMapping( rdt_rdtranking_wrank............ mapto rdt_rdtranking_wrank(id_title,wrank)............... )
      DBMapping( rdt_rdtranking_rank............. mapto rdt_rdtranking_rank(id_title,rank)................. )
      DBMapping( rdt_rdtranking_top.............. mapto rdt_rdtranking_top(id_title,top)................... )
 90   DBMapping( rdt_rdtranking_top_1............ mapto rdt_rdtranking_top_10(id_title).................... )
      DBMapping( rdt_rdtranking_top_2............ mapto rdt_rdtranking_top_21(id_title).................... )
      DBMapping( rdttitles....................... mapto rdttitles(titles,id_title)....................... )
      DBMapping( rdt_rdttitles_titles............ mapto rdt_rdttitles_titles(id_title,titles)............. )
      Mappings(3750): STOP
```

C.4. Versuch 3: Hinzunahme der Distributoren

```
      (DBMapper): Mappings(574): START
              ( Table/View ..................... mapto Literal.....................................)
      DBMapping( rdtcrosscinemat................. mapto rdtcrosscinemat(id_title,id_person,id_aka,uncredited) )
      DBMapping( rdt_rdtcrosscinemat_id_person... mapto rdt_rdtcrosscinemat_id_person(id_title,id_person) )
  5   DBMapping( rdt_rdtcrosscinemat_id_aka...... mapto rdt_rdtcrosscinemat_id_aka(id_title,id_aka)...... )
      DBMapping( rdt_rdtcrosscinemat_uncredited.. mapto rdt_rdtcrosscinemat_uncredited(id_title,uncredited) )
      DBMapping( rdt_rdtcrosscinemat_uncredited_1 mapto rdt_rdtcrosscinemat_uncredited_10(id_title)..... )
      DBMapping( rdt_rdtcrosscinemat_uncredited_2 mapto rdt_rdtcrosscinemat_uncredited_21(id_title)..... )
      DBMapping( rdtcrosscountries............... mapto rdtcrosscountries(id_title,id_country)........... )
 10   DBMapping( rdt_rdtcrosscountries_id_country mapto rdt_rdtcrosscountries_id_country(id_title,id_country) )
      DBMapping( rdt_rdtcrosscountries_id_country_1 mapto rdt_rdtcrosscountries_id_country_1C2(id_title)... )
      DBMapping( rdt_rdtcrosscountries_id_country_2 mapto rdt_rdtcrosscountries_id_country_2C128(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_3 mapto rdt_rdtcrosscountries_id_country_3C5(id_title)... )
      DBMapping( rdt_rdtcrosscountries_id_country_4 mapto rdt_rdtcrosscountries_id_country_4C43(id_title).. )
 15   DBMapping( rdt_rdtcrosscountries_id_country_5 mapto rdt_rdtcrosscountries_id_country_5C15(id_title).. )
      DBMapping( rdt_rdtcrosscountries_id_country_6 mapto rdt_rdtcrosscountries_id_country_6C9(id_title)... )
      DBMapping( rdt_rdtcrosscountries_id_country_7 mapto rdt_rdtcrosscountries_id_country_7C161(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_8 mapto rdt_rdtcrosscountries_id_country_8C41(id_title).. )
      DBMapping( rdt_rdtcrosscountries_id_country_9 mapto rdt_rdtcrosscountries_id_country_9C56(id_title).. )
 20   DBMapping( rdt_rdtcrosscountries_id_country_10 mapto rdt_rdtcrosscountries_id_country_10C173(id_title) )
      DBMapping( rdt_rdtcrosscountries_id_country_11 mapto rdt_rdtcrosscountries_id_country_11C160(id_title) )
      DBMapping( rdt_rdtcrosscountries_id_country_12 mapto rdt_rdtcrosscountries_id_country_12C67(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_13 mapto rdt_rdtcrosscountries_id_country_13C177(id_title) )
      DBMapping( rdt_rdtcrosscountries_id_country_14 mapto rdt_rdtcrosscountries_id_country_14C80(id_title). )
 25   DBMapping( rdt_rdtcrosscountries_id_country_15 mapto rdt_rdtcrosscountries_id_country_15C68(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_16 mapto rdt_rdtcrosscountries_id_country_16C112(id_title) )
      DBMapping( rdt_rdtcrosscountries_id_country_17 mapto rdt_rdtcrosscountries_id_country_17C142(id_title) )
      DBMapping( rdt_rdtcrosscountries_id_country_18 mapto rdt_rdtcrosscountries_id_country_18C79(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_19 mapto rdt_rdtcrosscountries_id_country_19C115(id_title) )
 30   DBMapping( rdt_rdtcrosscountries_id_country_20 mapto rdt_rdtcrosscountries_id_country_20C84(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_21 mapto rdt_rdtcrosscountries_id_country_21C13(id_title). )
      DBMapping( rdt_rdtcrosscountries_id_country_22 mapto rdt_rdtcrosscountries_id_country_22C146(id_title) )
      DBMapping( rdtcrossdist.................... mapto rdtcrossdist(id_title,id_company)................. )
      DBMapping( rdt_rdtcrossdist_id_company..... mapto rdt_rdtcrossdist_id_company(id_title,id_company). )
 35   DBMapping( rdt_rdtcrossdist_id_company_1... mapto rdt_rdtcrossdist_id_company_1D34888(id_title).... )
```

```
      DBMapping( rdt_rdtcrossdist_id_company_2... mapto rdt_rdtcrossdist_id_company_2D21494(id_title).... )
      DBMapping( rdt_rdtcrossdist_id_company_3... mapto rdt_rdtcrossdist_id_company_3D93943(id_title).... )
      DBMapping( rdt_rdtcrossdist_id_company_4... mapto rdt_rdtcrossdist_id_company_4D116759(id_title)... )
      DBMapping( rdt_rdtcrossdist_id_company_5... mapto rdt_rdtcrossdist_id_company_5D81644(id_title).... )
40    DBMapping( rdt_rdtcrossdist_id_company_6... mapto rdt_rdtcrossdist_id_company_6D45565(id_title).... )
      DBMapping( rdt_rdtcrossdist_id_company_7... mapto rdt_rdtcrossdist_id_company_7D91558(id_title).... )
      DBMapping( rdt_rdtcrossdist_id_company_8... mapto rdt_rdtcrossdist_id_company_8D103205(id_title)... )
      DBMapping( rdt_rdtcrossdist_id_company_9... mapto rdt_rdtcrossdist_id_company_9D79213(id_title).... )
      DBMapping( rdt_rdtcrossdist_id_company_10.. mapto rdt_rdtcrossdist_id_company_10D97985(id_title)... )
45    DBMapping( rdt_rdtcrossdist_id_company_11.. mapto rdt_rdtcrossdist_id_company_11D49785(id_title)... )
      DBMapping( rdt_rdtcrossdist_id_company_12.. mapto rdt_rdtcrossdist_id_company_12D93253(id_title)... )
      DBMapping( rdt_rdtcrossdist_id_company_13.. mapto rdt_rdtcrossdist_id_company_13D14195(id_title)... )
      DBMapping( rdt_rdtcrossdist_id_company_14.. mapto rdt_rdtcrossdist_id_company_14D87604(id_title)... )
      DBMapping( rdt_rdtcrossdist_id_company_15.. mapto rdt_rdtcrossdist_id_company_15D74329(id_title)... )
50    DBMapping( rdt_rdtcrossdist_id_company_16.. mapto rdt_rdtcrossdist_id_company_16D117730(id_title).. )
      DBMapping( rdt_rdtcrossdist_id_company_17.. mapto rdt_rdtcrossdist_id_company_17D100348(id_title).. )
      DBMapping( rdt_rdtcrossdist_id_company_18.. mapto rdt_rdtcrossdist_id_company_18D98483(id_title)... )
      DBMapping( rdt_rdtcrossdist_id_company_19.. mapto rdt_rdtcrossdist_id_company_19D17306(id_title)... )
      DBMapping( rdt_rdtcrossdist_id_company_20.. mapto rdt_rdtcrossdist_id_company_20D91833(id_title)... )
55    DBMapping( rdt_rdtcrossdist_id_company_21.. mapto rdt_rdtcrossdist_id_company_21D98255(id_title)... )
      DBMapping( rdt_rdtcrossdist_id_company_22.. mapto rdt_rdtcrossdist_id_company_22D113569(id_title).. )
      DBMapping( rdt_rdtcrossdist_id_company_23.. mapto rdt_rdtcrossdist_id_company_23D49117(id_title)... )
      DBMapping( rdt_rdtcrossdist_id_company_24.. mapto rdt_rdtcrossdist_id_company_24D16657(id_title)... )
      DBMapping( rdt_rdtcrossdist_id_company_25.. mapto rdt_rdtcrossdist_id_company_25D76155(id_title)... )
60    DBMapping( rdt_rdtcrossdist_id_company_26.. mapto rdt_rdtcrossdist_id_company_26D122793(id_title).. )
      DBMapping( rdt_rdtcrossdist_id_company_27.. mapto rdt_rdtcrossdist_id_company_27D2395(id_title).... )

      [...]  [...]  [...]  [...]  [...]  [...]  [...]  [...]  [...]  [...]  [...]  [...]  [...]  [...]  [...]

65    DBMapping( rdt_rdtcrossgenres_id_genre_13.. mapto rdt_rdtcrossgenres_id_genre_13G4(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_14.. mapto rdt_rdtcrossgenres_id_genre_14G11(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_15.. mapto rdt_rdtcrossgenres_id_genre_15G24(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_16.. mapto rdt_rdtcrossgenres_id_genre_16G22(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_17.. mapto rdt_rdtcrossgenres_id_genre_17G5(id_title)..... )
70    DBMapping( rdt_rdtcrossgenres_id_genre_18.. mapto rdt_rdtcrossgenres_id_genre_18G28(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_19.. mapto rdt_rdtcrossgenres_id_genre_19G25(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_20.. mapto rdt_rdtcrossgenres_id_genre_20G14(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_21.. mapto rdt_rdtcrossgenres_id_genre_21G17(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_22.. mapto rdt_rdtcrossgenres_id_genre_22G18(id_title)..... )
75    DBMapping( rdt_rdtcrossgenres_id_genre_23.. mapto rdt_rdtcrossgenres_id_genre_23G1(id_title)..... )
      DBMapping( rdt_rdtcrossgenres_id_genre_24.. mapto rdt_rdtcrossgenres_id_genre_24G21(id_title)..... )
      DBMapping( rdtpersonentop.................. mapto rdtpersonentop(id_person,top)..................... )
      DBMapping( rdt_rdtpersonentop_top.......... mapto rdt_rdtpersonentop_top(id_person,top)............. )
      DBMapping( rdt_rdtpersonentop_top_1........ mapto rdt_rdtpersonentop_top_10(id_person).............. )
80    DBMapping( rdt_rdtpersonentop_top_2........ mapto rdt_rdtpersonentop_top_21(id_person).............. )
      DBMapping( rdtranking...................... mapto rdtranking(id_title,wrank,rank,top)............... )
      DBMapping( rdt_rdtranking_wrank............ mapto rdt_rdtranking_wrank(id_title,wrank).............. )
      DBMapping( rdt_rdtranking_rank............. mapto rdt_rdtranking_rank(id_title,rank)................ )
      DBMapping( rdt_rdtranking_top.............. mapto rdt_rdtranking_top(id_title,top)................... )
85    DBMapping( rdt_rdtranking_top_1............ mapto rdt_rdtranking_top_10(id_title).................... )
      DBMapping( rdt_rdtranking_top_2............ mapto rdt_rdtranking_top_21(id_title).................... )
      DBMapping( rdttitles....................... mapto rdttitles(titles,id_title)........................ )
      DBMapping( rdt_rdttitles_titles............ mapto rdt_rdttitles_titles(id_title,titles)............. )
      Mappings(574): STOP
```