Studienarbeit

Konzeption einer Datenbank am Beispiel von Sybase zur Archivierung semistrukturierter Daten der Technikgeschichte von MV

Lehrstuhl Datenbank- und Informationssysteme
Fachbereich Informatik
Universität Rostock

Lars Schneider

Betreuer
Prof. Dr. Andreas Heuer
Dr. Ing. Antje Düsterhöft

Rostock, den 28. November 2000

Inhaltsverzeichnis

Kapitel 1

Einleitung

Heute gibt es eine sehr große Menge an Dokumenten zu den unterschiedlichsten Themengebieten, welche in unterschiedlichsten Medien (Zeitung, Buch, Internet, ...) und in verschiedenen Formaten (gedruckt, elektronisch) publiziert werden. Viele interessante Dokumente schlummern auch noch in Archiven. Bei der Sammlung von Dokumenten zu einem Themenschwerpunkt, wie z. B. Dokumente der Technikgeschichte von Mecklenburg-Vorpommern, liegt es nun nahe, sich die Mühe zu machen, die Dokumente einerseits in einem bestimmten elektronischen Format zentral zu sammeln bzw. vorher noch elektronisch zu erfassen, um sie einer breiten Öffentlichkeit z. B. über das Internet zugänglich zu machen. Dabei ist die Informationsrückgewinnung ein wichtiges Kriterium und die Verwendung einer Datenbank zur Speicherung, Verwaltung sowie Manipulation der Daten liegt auf der Hand.

Im folgenden werden die an dieser Studienarbeit beteiligten Partner, das Ziel und die Problemstellung sowie der Inhalt dieser Studienarbeit vorgestellt.

1.1 Einordnung der beteiligten Partner

Die Ernst-Alban[1]-Gesellschaft für Mecklenburgisch-Pommersche Wissenschafts- und Technikgeschichte e. V. (EAG) wurde im Juni 1993 in Rostock als landesweite Gesellschaft im Bundesland Mecklenburg-Vorpommern gegründet. Ihr Ziel ist die Pflege, Bewahrung und Forschung der regionalen Bildungs-, Industrie-, Medizin-, Technik- und Wissenschaftsgeschichte, um damit dem Vorurteil, Mecklenburg und Pommern sind auf den zuvor genannten Gebieten rückständig gewesen, entgegenzuwirken. Es hatte sich im Laufe der Zeit herausgestellt, daß das Interesse an der historischen und aktuellen Entwicklung der Technik eines Landes, welches traditionell überwiegend Agrarland war, groß ist. In diesem Zusammenhang hat die EAG einen umfangreichen Datenbestand aus unterschiedlichen Quellen zusammengetragen und einiges bereits in Worddokumenten erfaßt. Ziel ist es, diese Informationen mit zusätzlichen touristischen Angaben einer breiten touristisch und technisch interessierten Öffentlichkeit zugänglich zu machen.

Die Firma MEDEOCOM Gesellschaft für Informations- und Kommunikationssysteme mbH [2] ist ein EDV-Systemhaus und Internet-Service-Provider mit Sitz in Rostock. Sie beschäftigt sich vorwiegend mit der Entwicklung, Herstellung und dem Vertrieb von Informations- und Kommunikationssystemen und ist u. a. Online-Partner des regionalen Online-Dienstes für Mecklenburg-Vorpommern "MVweb". Dieser Hintergrund ist auch die Plattform der Zusammenarbeit zwischen der EAG und der Firma MEDEOCOM, mit dem Ziel der Realisierung einer öffentlichen Datenbank z.B. im Internet zur technischen Entwicklung von Mecklenburg-Vorpommern.

[1] Benannt nach dem urmecklenburgischen Arzt, Ingenieur und Maschinenbauer Ernst Alban, geboren 1791 in Neubrandenburg und gestorben 1856 in Plau am See

[2] http://www.medeocom.de

1.2 Ziel und Problemstellung der Arbeit

Das Ziel dieser Studienarbeit ist der Entwurf einer Datenbank zur Speicherung von vorhandenen semistrukturierten multimedialen Dokumenten (Texte und Bilder) zur Technikgeschichte in Mecklenburg-Vorpommern. Als Datenbank steht das relationale Datenbanksystem Sybase SQL Anywhere 6.0 mit dem Entwurfstool SQL Modeler 6.1 zur Verfügung.

In diesem Zusammenhang sind zwei Problemstellungen zu betrachten. Zum einen sind die beim Datenbankentwurf relevanten Kriterien zu beachten und zum anderen sind aufgrund der besonderen Art der Dokumente Aspekte des Information Retrieval zu berücksichtigen.

1.3 Aufbau der Arbeit

In Kapitel 2 werden Konzepte des Information Retrieval (kurz IR) vorgestellt und mit denen der Datenbanken verglichen, soweit dies für die Studienarbeit notwendig erscheint, bevor in Kapitel 3 mit der Analyse der vorhandenen Daten begonnen wird. In Kapitel 4 wird der Datenbanksystem unabhängige konzeptionelle Datenbankentwurf mit dem EER-Modell und die Abbildung des EER-Modells auf das Relationenmodell dargestellt. Die Datenbankimplementation wird in Kapitel 5 vorgestellt, wobei hier auch das verwendete Case-Tool kurz vorgestellt wird und ein Anwendungsszenario beschrieben wird. Dieses Anwendungsszenario basiert auf dem in Kapitel 2 beschriebenen Modell für Informationssysteme mit Dateneingabe bzw. Wissensrepräsentation, Datenhaltung und Retrievalfunktionalitäten. Zum Abschluß werden die wichtigsten Punkte der Studienarbeit zusammengefaßt und ein Ausblick auf die Weiterführung dieser Arbeit gegeben.

Kapitel 2

Information Retrieval und Datenbanken

In Datenbanken sollen zunehmend multimediale Dokumente (vgl. Kapitel 1) gespeichert, verwaltet und manipuliert werden. Charakterisiert man multimediale Dokumente als unstrukturiert bis semistrukturiert, abstrahiert weiter nur auf Textdokumente, dann war die Verwaltung dieser Dokumente bisher nur IR-Systemen vorbehalten, die Datenbanken meist nur zur Speicherung der Dokumente nutzten. Die Speicherung von großen Datenmengen ist dabei nicht mehr das Problem, da die meisten Datenbanken dafür geeignete Datentypen (CLOB[1], BLOB[2]) und entsprechende Speichermechanismen bereitstellen. Vielmehr steht die Frage im Vordergrund ein geeignetes Datenmodell zu finden, um über strukturierte Attribute unstrukturierte Dokumente zu verwalten bzw. anfragen zu können, ohne dabei auf externe Retrievalwerkzeuge zurückgreifen zu müssen. Hierzu ist es notwendig, sich mit IR-Techniken zu befassen und diese den Datenbanktechniken gegenüberzustellen. Die folgende kurze thematische Einordnung und kurzen Erläuterungen zu IR sind ausführlich in [Fuh97], [Fer99] und zu Datenbanken in [SH97] nachzulesen.

2.1 Information Retrieval

Das klassische Anwendungsgebiet von IR-System basiert auf der inhaltlichen Suche in Texten. Allgemein sind Texte Dokumente in natürlicher Sprache und enthalten unformatierte und unstrukturierte bis semistrukturierte Daten. Ziel eines IR-Systems ist es, mit Hilfe von IR-Strategien Änderungen und Anfragen an den Datenbestand "möglichst genau" auszuführen. Problem dabei ist die Mehrdeutigkeit der natürlichen Sprache.

2.1.1 Daten-Wissen-Information

In der Informationsverarbeitung unterscheidet man die Begriffe Daten, Wissen und Information.

Daten sind auf der syntaktischen Ebene einzuordnen, wenn man von einer reinen Datensammlung ohne Semantik ausgeht.

Wissen sind Daten mit Semantik. Demzufolge kann man bei einer Datenbank davon ausgehen, daß in ihr Wissen enthalten ist, da ein Teil der Anwendung semantisch im Datenbankmodell enthalten ist. Das gleiche gilt für IR-Systeme, wobei hier spezielle Verfahren der Wissensrepräsentation verwendet werden.

[1] CLOB - Character Large Object
[2] BLOB - Binary Large Object

Information ist der pragmatische Aspekt und folgendermaßen definiert (aus [Fuh97]): „Information ist die Teilmenge von Wissen, die von jemandem in einer konkreten Situation zur Lösung von Problemen benötigt wird“. Ein Informationssystem wird dazu genutzt, aus gespeichertem Wissen Informationen zu extrahieren.

Oberflächlich betrachtet sind Datenbanken und IR-Systeme Informationssysteme, doch unterscheiden sich beide. Die Unterschiede zwischen einem IR-System und einem DB-System liegen im allgemeinen in der Art der zu speichernden Daten (strukturierte vs. unstrukturierte), in der Art der verwendeten Wissensrepräsentation (relational vs. Klassifikationen, Thesauri, Semantische Netze bzw. Ontologien, ...) und der darauf angewendeten Retrievalverfahren (SQL vs. Boolean-Retrieval, Fuzzy-Retrieval, ...) zur Informationsgewinnung.

2.1.2 Grundmodell des Information Retrieval

Information Retrieval kann als ein Kommunikationsprozeß aufgefaßt werden, bei dem die Daten vor der Speicherung durch Verfahren der Wissensrepräsentation analysiert werden, bevor sie als Wissen intern abgespeichert werden können. Informationen werden durch Transformationen aus dem gespeicherten Wissen erzeugt. Am Beispiel der Abbildung 2.1 vom Grundmodell des IR sollen einige ausgewählte Verfahren der Wissensrepräsentation und einige Retrievalmodelle kurz erläutert werden.

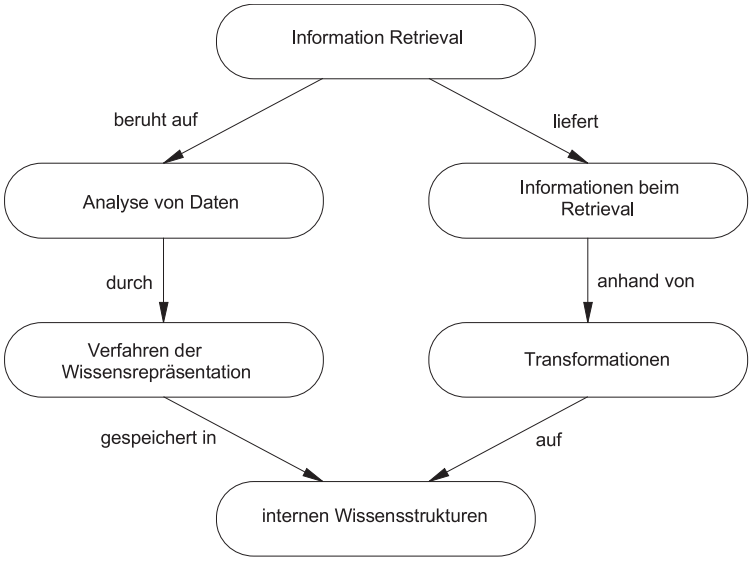

Abbildung 2.1: Grundmodell des Information Retrieval (aus [Fuh97])

2.1.2.1 Verfahren der Wissensrepräsentationen für Textdokumente

Da in IR-Systemen überwiegend eine inhaltliche Suche in Texten erfolgt, müssen die Textinhalte geeignet repräsentiert werden, um im Retrievalprozeß relevante Dokumente finden zu können. Die Repräsentation soll einerseits unterschiedliche Formulierungen auf die gleiche Repräsentation abbilden (Erhöhung des Recall[3]) und andererseits unklare Formulierungen durch die Mehrdeutigkeit einzelner Wörter präzisieren (Erhöhung der Precision[4]). Um die Dokumente semantisch zu ergänzen gibt es verschiedene Verfahren der Wissensrepräsentationen, von denen hier einige vorgestellt werden.

[3] Anteil der gefunden relevanten Dokumente an den gesamten relevanten Dokumenten (setzt Kenntnis der Anzahl relevanter Dokumente voraus)

[4] Anteil relevanter Dokumente an den gefunden Dokumenten

Freitextsuche

Bei der Freitextsuche wird keine zusätzliche Repräsentation des Textes erstellt, sondern es werden Suchfunktionen auf den Text angewendet. Zuerst wird der Text mit den folgenden Schritten zur anschließenden Freitextsuche vorbereitet:

1. Zerlegung des Textes in einzelne Wörter
2. Stoppwortentfernung wie Artikel, Füllwörter, Konjunktionen, da sie nicht bedeutungstragende Wörter sind
3. Satzendeerkennung, weil Freitextfunktionen sich auch auf Satzgrenzen beziehen

Die eigentliche Freitextsuche findet auf dem reduzierten Text statt, wobei zwischen zwei Ansätzen zu unterscheiden ist:

- informatischer Ansatz:
  Hier wird der reduzierte Text als eine Folge von Zeichenketten aufgefaßt und mit entsprechenden Zeichenkettenoperatoren durchsucht. Truncation- und Maskierungsoperatoren dienen zur Erkennung einzelner Wörter und deren Flexions- und Derivationsformen. Kontextoperatoren unterstützen die Suche nach mehrgliedrigen Ausdrücken.
- computerlinguistischer Ansatz:
  Hierbei wird versucht, mit morphologischen und syntaktischen Verfahren eine Normalisierung der einzelnen Wortformen zu erreichen, wobei der Text weiter reduziert wird. Die eigentliche Suche auf dem Text bezieht sich auf Wörter und erfolgt algorithmisch, teilweise auch mit Hilfe von Wörterbüchern.

Klassifikationen

Klassifikationen strukturieren Objekte einer speziellen Anwendungsdomäne entsprechend ihren Eigenschaften in Klassen. Aufgrund seiner Eigenschaft läßt sich dann jedes Objekt genau einer Klasse zuordnen, bzw. falls ein Objekt mehrere Eigenschaften besitzt, wird es mehreren Klassen zugeordnet. Der praktische Vorteil hierbei ist, daß zum Aufstellen von Klassifikationen nur Fachwissen der Domäne benötigt wird und diese Einteilungen durch Außenstehende leicht nachvollziehbar sind. Dieser Vorteil der Einfachheit hat aber auch seine Grenzen, wenn man Klassifikationen mit den semantischen Fähigkeiten anderer Wissensrepräsentationen vergleicht.

Ist die Klasseneinteilung disjunkt, so kann sie als Baumstruktur repräsentiert werden und ist monohierarchisch. Anderenfalls können auch mehrere Oberklassen zu einer Klasse gehören. Die Klasseneinteilung ist dann polyhierarchisch.

Sind bei der Klasseneinteilung auf einer Stufe mehrere Eigenschaften gegeben, nach denen weiter strukturiert werden soll und sind diese Eigenschaften orthogonal zueinander, so spricht man von Polydimensionalität, ansonsten von Monodimensionalität.

Allgemein verwendete Klassifikationen sind z.B. die internationale Dezimalklassifikation ([Fuh97] ab Seite 63 und [Fer99] ab Seite 29) und die CR-Klassifikation ([Fuh97] ab Seite 65).

Thesauri

Ein Thesaurus ist eine geordnete Zusammenstellung von Begriffen. Im Gegensatz zur hierarchischen Klassifikation mit der disjunkten Einteilung von Objekten liegt hier der Schwerpunkt auf der Erfassung von Wörtern, Termen und Ausdrücken einer Anwendungsdomäne. Zu jedem Begriff wird im Thesaurus eine kurze Begriffsdefinition gespeichert und werden die verschiedenen Relationen zu anderen Begriffen (Synonyme, Antonyme, verwandte Wörter, Oberbegriffe, speziellere Begriffe) erfaßt.

Semantische Netze - Ontologien

Ein semantisches Netz, im folgenden auch als Ontologie bezeichnet, dient zur besseren Darstellung inhaltlicher Zusammenhänge von Dokumenten einer speziellen Anwendungsdomäne. In der Ontologie werden Konzepte (Klassen - Begriffe) und Rollen (Beziehungen) dargestellt. Abstrakte Klassen besitzen im Gegensatz zu konkreten Klassen keine Instanzen. Jede Klasse enthält eine verbale Definition, Attribute und deren Instanzen und die Beziehungen zu anderen Klassen. Die Beziehungen zwischen den Klassen entsprechen einer hierarchischen Klassifikation, wobei zur Darstellung der Art der Beziehung mehr Möglichkeiten als die Spezialisierung existieren. Graphisch läßt sich eine Ontologie als ein Graph mit Knoten (Konzepte) und Kanten (Beziehungen) darstellen.

Vorteil hierbei ist die Möglichkeit einer sehr präzisen Darstellung von inhaltlichen Zusammenhängen, aber dafür ist der Aufwand zur Erstellung einer allgemeingültigen Ontologie für eine Anwendungsdomäne sehr hoch. Einerseits wird Domänenwissen und andererseits Wissen aus dem Fachgebiet der Computerlinguistik benötigt.

2.1.2.2 Retrievalverfahren

Auf der Seite der Anfragen gibt es ebenfalls unterschiedliche Verfahren, die zum Teil spezielle Wissensrepräsentation bedingen. Ziel ist hierbei die Erzeugung von Informationen auf eine formalisierte Anfrage, die gegebenenfalls erst im Retrievalprozeß erzeugt wird. Durch eine Retrievalfunktion werden die Frage-Dokument-Paare miteinander verglichen und bei Übereinstimmung als Informationen aus dem Retrievalprozeß geliefert. In diesem Zusammenhang sind das Ranking, Relevance Feedback und die Indexierung der Dokumente wichtig.

Ranking (siehe dazu [Fuh97] Kapitel 3.6 Rangordnungen) erzeugt eine Rangordnung der gefundenen Dokumente entsprechend eines im Retrievalprozeß erzeugten Meßwertes für die Güte der Dokumente.

Relevance Feedback (siehe dazu [Fuh97] Kapitel 5.5 Vektorraummodell) ist die Verbesserung der Retrieval-Qualität durch Änderung der Fragetermgewichtung aufgrund von Relevancebeurteilungen der bereits gefundenen Dokumente. Dies ist nur durchführbar mit geeigneter Dokumententerm- und Fragetermgewichtung, wie z.B. beim Vektorraummodell.

Indexierung (siehe dazu [Fuh97] Kapitel 8.4 Zugriffspfade) der Dokumente eines Retrievalsystems ist von großer Bedeutung. Im Index eines IR-Systems ist semantisches Wissen enthalten, somit dient der Index der Wissensrepräsentation. Weiterhin dient der Index als Zugriffspfad zu den im IR-System enthaltenen Dokumenten. Da die Suche meist auf einem sehr umfangreichen Datenbestand durchgeführt wird, erhöht die Indexierung die Performance im Retrievalprozeß entscheidend.

Boolean-Retrieval

Boolean Retrieval ist historisch betrachtet das erste und bekannteste Retrievalverfahren. Die Anfragen bestehen aus Schlüsselwörtern, die mit den logischen Operatoren AND, OR und NOT verknüpft werden. Aufgrund der binären Gewichtung der Anfrageterme kann die Retrievalfunktion nur 0 für nicht relevante und 1 für relevante Dokumente als Retrievalgewichte liefern. Durch diese strikte Zweiteilung der Dokumente ist es nicht möglich, ein Ranking der gefundenen Dokumente durchzuführen, da diese alle das gleiche Retrievalgewicht besitzen.

Vor- und Nachteile:

+ effizient zu implementieren
+ Mächtigkeit der Anfragesprache, da jede beliebige Teilmenge selektierbar ist

- keine Gewichtung der Anfrageterme und somit strenge Trennung in nicht gefundene und gefundene Dokumente (kein Ranking bzw. nur möglich durch Simulation von z.B. Vorkommenshäufigkeiten einzelner Terme)
- Größe der Antwortmenge schwer kontrollierbar
- schwierige Anfrageformulierungen

Fuzzy-Retrieval

Im Gegensatz zum booleschen Retrieval können den Dokumenten reelle Retrievalgewichte aus dem Intervall von [0,1] aufgrund der Verwendung Fuzzy-Logik zugeordnet werden. Damit ist ein Ranking der gefundenen Dokumente möglich.

Vor- und Nachteile:

+ Rangordnung auf gefundenen Dokumenten realisierbar
- schwierige Anfrageformulierungen, da ähnlich denen des booleschen Retrieval
- Fragetermgewichtung nicht möglich
- Retrievalqualität nicht zufriedenstellend

Vektorraummodell

Im Vektorraummodell werden Dokumente und Fragen als Punkte in einem Vektorraum aufgefaßt, der durch die Terme der Domäne aufgespannt wird. Der Retrievalprozeß sucht nach Dokumenten, deren korrespondierende Vektoren ähnlich dem des Fragevektors sind. Voraussetzungen sind, daß die Termvektoren orthogonal und normiert sind.

Vor- und Nachteile:

+ Verbesserung der Retrievalqualität gegenüber den vorher genannten Modellen, insbesondere wegen der Möglichkeit des Relevance Feedback
+ benutzerfreundlich aufgrund einfacher Frageformulierungen
- heuristischer Ansatz zur Berechnung der Indizierungsgewichte hat zur Folge, daß Dokumentrepräsentationen schlecht erweiterbar sind

2.1.3 Dokumentretrieval als klassisches IR-Modell

Wie bereits im Abschnitt 2.1.2 beschrieben, können wir beim IR-Modell zwischen der Eingabeseite und der Ausgabeseite unterscheiden. Das Grundmodell des IR in Abbildung 2.1 auf Seite 4 diente dabei zur Veranschaulichung und Einordnung der grundlegenden Verfahren des IR. Da IR meist auf der inhaltlichen Suche in Texten basiert, läßt sich das allgemeine Grundmodell zum Grundmodell für Dokumentretrieval spezialisieren. Die Spezialisierung wie in Abbildung 2.2 ist möglich, da die Art der Dokumente (Daten, Wissen) die Art des IR-Systems und somit auch die speziellen Verfahren bedingen. Die unterschiedlichen Möglichkeiten zur Wissensrepräsentation von Textdokumenten wurden bereits im gleichnamigen Abschnitt 2.1.2.1 ab Seite 4 und die Retrievalverfahren in Abschnitt 2.1.2.2 ab Seite 6 vorgestellt.

Die bisherige semantische Sicht auf die Dokumente, gemeint ist hierbei die Semantik in Bezug auf den Inhalt der Dokumente als Wissen, ist für ein IR-System elementar wichtig. Auf eine Anfrage an ein IR-System sollen Informationen, also relevante Dokumente, aufgrund ihrer Semantik gefunden werden. Dies bedingt die semantische Interpretation der Daten als Wissen zum einen und eine geeignete Retrievalmöglichkeit zum anderen. In der Regel wird deshalb im IR die semantische Sicht als ausreichend betrachtet, z.B. Textdokumente als unstrukturierte Dokumente zu betrachten und deren

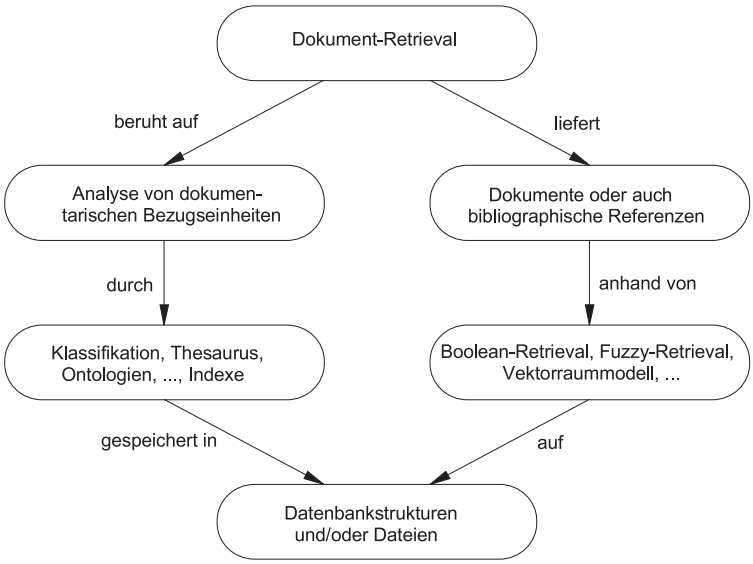

Abbildung 2.2: Grundmodell für Dokumentretrieval (aus [Fuh97])

Semantik auf Termebene zu erschließen, was für die eigentliche Informationsgewinnung zwar ausreichend wäre, aber in der Praxis anders aussieht. In der Praxis sind die Dokumente strukturiert und werden in Abhängigkeit der Anwendung speziell dargestellt. Diese unterschiedlichen Sichtweisen sind in Abbildung 2.3 am Beispiel von XML[5]-Dokumenten dargestellt.

- Die semantische Sicht, also mögliche Arten der Wissensrepräsentation, wurden bereits in Abschnitt 2.1.2.1 ab Seite 4 erläutert.
- Die als Daten bezeichneten Dokumente sind in der Regel nicht unstrukturiert, sondern besitzen eine logische Struktur. Diese Struktur soll im IR-System erhalten bleiben und ist notwendig, wenn die Dokumente aus der Datenbasis wieder rekonstruiert werden müssen.
- Das Layout ermöglicht eine einheitliche Ausgabe der Dokumente auf unterschiedlichen Medien. Diese Sicht soll aber nicht weiter interessieren, da es nicht unbedingt notwendig ist, Informationen zur Layout-Sicht zu speichern. Diese Sicht wird von der externen Anwendung bestimmt bzw. ist durch sie vorgegeben.

Um Dokumente eines IR-Systems austauschbar und wiederverwendbar zu machen, müssen die Dokumente in einem standardisierten Dokumentformat vorliegen. Im einfachsten Fall also ASCII-Texte, die aber den heutigen Anwendungsansprüchen nicht mehr genügen. Aus diesem Grund sind standardisierte Dokumentformate entwickelt worden, die zum einen die logische Struktur und zum anderen die Layoutstruktur beschreiben können. Als Beispiele seien hier das ODA - Dokumenten-Architektur-Modell[6] und SGML[7] genannt und auf [Fuh97] Kapitel 8.3 ab Seite 172 verwiesen.

Interessant scheint in diesem Zusammenhang die Betrachtung von XML [BM98] als ein durch das World Wide Web Consortium (W3C) standardisiertes Dokumenten-Austauschformat. XML ist eine Teilmenge von SGML. In XML-Dokumenten wird die logische Struktur getrennt von der Layout-Struktur definiert. Die logische Struktur wird in der XML-DTD[8] beschrieben und das Layout wird

[5] XML - Extensible Markup Language
[6] ODA - Office Document Architekture
[7] SGML - Standard Generalized Markup Language
[8] DTD - Document Type Definition

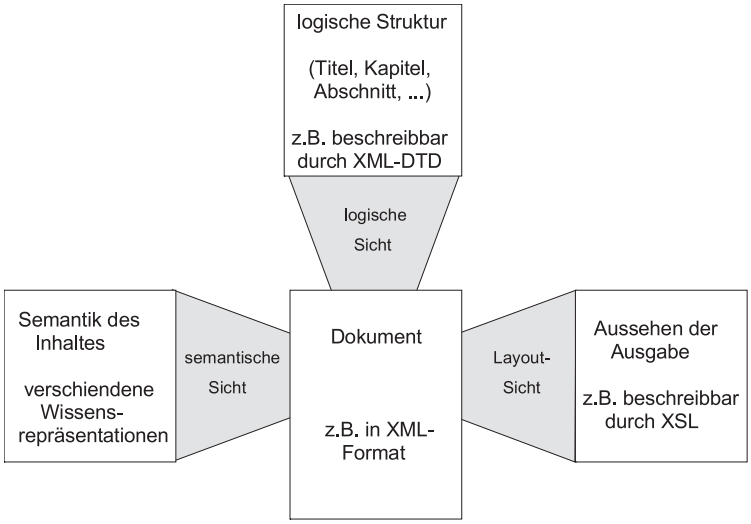

Abbildung 2.3: Sichten auf Dokumente (aus [Fuh97])

mit der XSL[9] festgelegt.Weiterhin wird zwischen logischer und physischer Struktur unterschieden. Die physische Struktur bestimmt die Zusammensetzung des Dokumentes aus verschieden separat gespeicherten Entities, wie z.B. externe Bilder. Die logische Struktur bestimmt die interne Struktur des Dokumentes, z.B. durch Abschnitte, Unterabschnitte, den sogenannten Elementen. Diese Eigenschaften machen XML zu einem bevorzugten Dokumenten-Austauschformat.

2.2 Datenbanken

Da Datenbanken mittlerweile nicht nur zur Speicherung von multimedialen Daten genutzt werden, sollen im folgenden kurz die wesentlichen Vorzüge bei der Nutzung von Datenbanken hervorgehoben werden. Weiterhin soll die Nutzung eines relationalen Datenbanksystems am Beispiel des Faktenretrieval erläutert werden. Im Vordergrund dabei steht der Vergleich von IR-Konzepten mit denen der Datenbank. Dies soll aber nicht als Versuch betrachtet werden, eine Datenbank als IR-System zu sehen, sondern eher dazu dienen, abschätzen zu können, ob der Einsatz eines relationalen Datenbanksystems ohne spezielle externe und interne IR-Werkzeuge bzw. -Erweiterungen ausreichende Retrievalmöglichkeiten bietet, um den speziellen Informationswunsch der Anwendungsdomäne Technikgeschichte-Mecklenburg-Vorpommern (kurz TGMV) gerecht zu werden.

2.2.1 Eigenschaften

Die folgenden Anforderungen haben sich als Basis-Funktionalitäten für Datenbanksysteme (vgl. [SH97]) erwiesen und werden von den meisten Datenbanksystemen erfüllt. Damit eigenen sich diese Datenbanksysteme zumindest zur Speicherung von großen Datenmengen.

Datenintegration fordert die einheitliche Verwaltung von Daten und unterstützt dadurch die redundanzfreie Datenhaltung.

Konsistenzerhaltung, auch bekannt als Integritätssicherung, übernimmt die Kontrolle der Korrektheit von Datenbankinhalten und korrekten Ausführung von Datenmanipulationen.

Datenschutz gewährleistet ausschließlich autorisierte Zugriffe auf die gespeicherten Daten.

[9] XSL - XML Style Language

Transaktionen sind als Einheit zusammengefaßte Änderungsoperationen, die entweder ganz oder gar nicht durchgeführt werden dürfen, um die Datenbankintegrität zu gewährleisten bzw. zu erhalten.

Synchronisation realisiert den konkurrierenden Zugriff auf die Datenbank.

Datensicherung soll die Wiederherstellung der Daten nach Systemfehlern gewährleisten, ersetzt aber keine externe Sicherung der Datenbank.

Katalog ermöglicht Zugriffe auf die Datenbeschreibungen der Datenbank, und enthält das semantische Modell in Form eines physischen Modells, also bei einer relationalen Datenbank die Tabellendefinitionen mit den Attributdefinitionen und den Integritätsbedingungen.

Datenunabhängigkeit wird durch die Drei-Ebenen-Architektur von Datenbanken gewährleistet. Die externe Ebene ist die Betrachtung aus Anwendungssicht. Die konzeptuelle Ebene beschreibt die Struktur der Datenbank basierend auf dem implementierungsunabhängigen und systemunabhängigen semantischen Datenmodell (z.B. EER-Modell). Die interne Ebene ist die systemspezifische Realisierung der Datenstrukturen z.B. durch Zugriffspfade, Speicherstrukturen, etc..

Datenbankoperationen ermöglichen mittels einer benutzergerechten Sprache, den Zugriff auf die Daten in der Datenbank.

Die Datenbanken haben sich aus Dateiverwaltungssystemen entwickelt und bieten heute eine vielfaches an Funktionalität, was man eigentlich von einem puren Datenspeicherungssystem erwartet. Ein Datenbanksystem ist in gewisser Weise ein Informationssystem, auch wenn es nie als isolierte Software eingesetzt wird, sondern meistens Teil eines komplexen Softwaresystems ist. Dies wird aufgrund der Drei-Ebenen-Architektur durch die damit realisierte Datenunabhängigkeit ermöglicht.

2.2.2 Datenbank und Faktenretrieval

Geht man bei einer relationalen Datenbank davon aus, daß in diesem Datenbanksystem nur strukturierte Daten (Fakten) gespeichert und angefragt werden sollen, so kann man dieses Datenbanksystem als ein Faktenretrieval-System (vgl. Abbildung 2.4) bezeichnen.

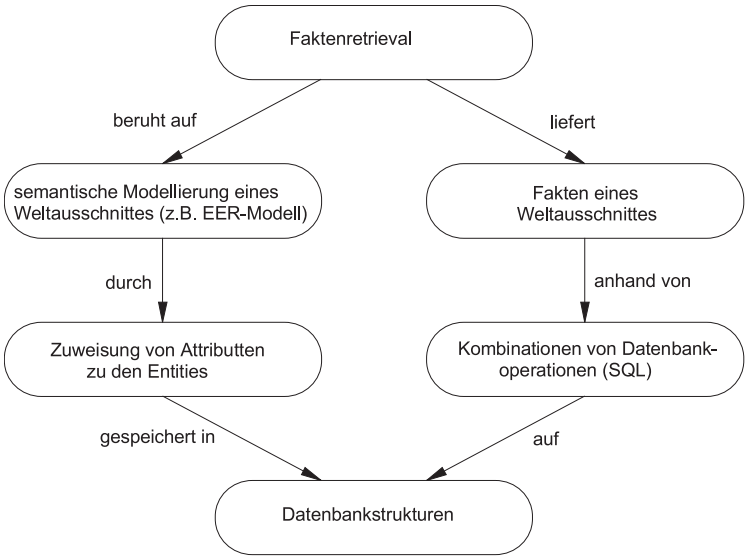

Abbildung 2.4: Grundmodell für Faktenretrieval (aus [Fuh97])

Ähnlich dem Grundmodell in Abbildung 2.1 auf Seite 4 erfolgt auf der Eingabeseite eine Analyse der Daten, deren Ergebnis hier ein semantisches Modell ist, z.B. EER-Modell. In diesem Modell werden die Entities mit ihren Eigenschaften (Attribute) und deren Beziehungen untereinander beschrieben und dienen als Grundlage für das in der Datenbank in der konzeptuellen Ebene gespeicherte physische Datenmodell. Durch das physische Datenmodell wird eine Struktur der Fakten in der Datenbank definiert, nämlich die Beschreibung von Entities und Beziehungen in Form von Tabellen, deren Spalten die Attribute sind. Hier steht die logische Struktur der Daten, also der Fakten, im Vordergrund. Im Gegensatz dazu stand beim Grundmodell des Dokument-Retrieval in Abbildung 2.2 auf Seite 8 die semantische Sicht über Wissensrepräsentationen auf den Inhalt der Dokumente im Vordergrund.

Auf der Seite der Anfragen steht die Suche nach der logischen (syntaktischen) Struktur im Vordergrund, was einerseits Kenntnisse über die Struktur der Daten und andererseits Kenntnisse der formalen Anfragesprache voraussetzt, falls direkt mit der Datenbank gearbeitet wird. Die mit der Anfrage exakt übereinstimmenden Fakten werden als Ergebnis geliefert, wobei sich mit Hilfe der Anfragesprache die Struktur (Tabelle) des Ergebnisses beeinflußen läßt.

Bei dieser Sichtweise wird ausschließlich von relationalen Datenbanken mit atomaren Attributen ausgegangen. Wenn diese Einschränkung in der Praxis auch gelten würde, dann wären Datenbanken nicht einmal zur Speicherung von Dokumenten beim Dokument-Retrieval einsetzbar. Datenbanken haben sich in Bezug auf ihre Funktionalität für IR unterschiedlich stark entwickelt. Grundvoraussetzung ist die Möglichkeit der Speicherung von unstrukturierten Dokumenten in entsprechenden Datentypen. Dies beherrschen die meisten Datenbanken. Wünschenswert ist aber weiterhin die Möglichkeit der gleichzeitigen Anfrage an strukturierte und unstrukturierte Datenbankinhalte. In diesem Punkt unterscheiden sich die Datenbanken voneinander.

2.3 Faktenretrieval vs. Information Retrieval

Bisher wurde IR, das Grundmodell und seine Arten der Spezialisierung, getrennt voneinander betrachtet. Ähnlich ist auch die Entwicklung von IR-Systemen und Datenbanken zu sehen, auch wenn es heute immer schwerer fällt, beide getrennt voneinander zu betrachten. Kaum ein IR-System verwendet keine Datenbank und umgekehrt spezialisieren sich die Datenbanken immer mehr und integrieren Konzepte des IR. Beide Systeme werden als Informationssysteme bezeichnet, deren Ziel es ist, durch geeignete Datenrepräsentationen Nutzeranfragen effizient auszuführen. So ist es auch nicht verwunderlich, daß es schon eine Vielzahl von Systemen (Suchmaschinen, Wissensbanken, ...) gibt, die unter Verwendung unterschiedlicher Kopplungsarten beide Systemarten ineinander integriert haben.

Aus der folgenden Tabelle 2.1 wird ersichtlich, welche Anforderungen bisher aufgrund der Daten an die beiden Retrievalsysteme gestellt wurden. Dies hat bisher auch die starke Spezialisierung der beiden Systeme bedingt.

	Faktenretrieval	Information Retrieval
Matching	exakt	partiell
Inferenz	deduktiv	induktiv
Modell	deterministisch	probabilistisch
Klassifikation	monothetisch	polithetisch
Anfragesprache	formal	natürlich
Fragespezifikation	vollständig	unvollständig
gesuchte Objekte	passende	relevante
Reaktion auf Fehler	sensitiv	insensitiv

Tabelle 2.1: Faktenretrieval vs. Informationretrieval aus [Rij79]

Im folgenden sollen nochmal die wesentlichen Unterschiede zwischen den beiden Retrievalsystemen in Bezug auf die einzelnen Komponenten des IR-Prozesses zusammengefaßt werden. Dies dient dazu, weil teilweise Wissen des IR benötigt wird, um Daten für inhalts- und auch strukturorientierte Anfragen geeignet repräsentieren zu können und später beurteilen zu können, ob das Datenbanksystem mit der gewählten Modellierung des Weltausschnittes den Informationswunsch der Anwendungsdomäne TGMV erfüllen kann.

Wissensrepräsentation
In einem IR-System steht die semantische Inhaltserschließung im Vordergrund, wobei die logische Struktur der Daten meistens unberücksichtigt bleibt. Diese Interpretation ist unsicher, da ausgehend von der Mehrdeutigkeit der natürlichen Sprache der Inhalt der Dokumente nur unsicher wiedergegeben werden kann, was sich auch auf das zu speichernde Wissen und die Anfrageergebnisse auswirkt.

In einem Datenbanksystem steht die logische Struktur der Daten im Vordergrund und das entworfene Datenbankmodell soll die Daten möglichst exakt und vollständig beschreiben. Durch das Datenbankmodell wird die Struktur der Daten beschrieben, die die Bedeutung der Objekte des Weltausschnittes wiedergeben soll. Dies bedingt eine Reduktion auf exakte, vollständige Werte, wobei ein möglicher Informationsverlust vermieden werden soll.

Wünschenswert ist hier auf Seite der Datenbanken die Speicherung unstrukturierter Daten (Textdokumente) und die Möglichkeit der semantischen Inhaltserschließung z.B. durch die Anfragesprache.

Datenverwaltung
Die Speicherung des Wissens eines IR-Systems in einer Datenbank ist nur eine mögliche Variante. Vielmehr steht die Wissensrepräsentation und der Zugriff über diese Wissensrepräsentation auf das Wissen im Vordergrund. So werden z.B. in einem IR-System (vgl. Abschnitt 2.1.2.2 auf Seite 6 oder auch [Fuh97] Kapitel 8.4 Zugriffspfade) Indexe als Wissensrepräsentation genutzt, aber auch gleichzeitig als Pfad für den effizienten Zugriff auf die Daten benötigt.

Die Indexierung in Datenbanken ermöglicht ausschließlich den effizienten Zugriff auf die Daten. In den Datenbanksystemen gehört der Index zur internen Ebene der Drei-Ebenen-Architektur, die auch die Datenunabhängigkeit gewährleistet. Dies bedeutet, das ein Datenbanksystem ohne eingerichteten Zugriffspfad die gleiche Ergebnismenge liefert, wie mit eingerichtetem Zugriffspfad. Weiterhin bieten Datenbanken bei der Speicherung der Daten Datenintegrität, das Transaktionskonzept, die Synchronisation, Datensicherung und Datenschutz. Diese Aspekte werden bei IR-Systemen nur im Bedarfsfall betrachtet.

Die Speicherung von Daten bzw. Wissen in einem Datenbanksystem bietet aufgrund der Eigenschaften von Datenbanken (vgl. auch Abschnitt 2.2.1 auf Seite 9) entscheidende Vorteile.

Retrievalfunktionalität
Bei Anfragen in einem IR-System werden relevante Dokumente gesucht, deren Bedeutung am besten zur Anfrage paßt. Diese inhaltlich basierten Anfragen an unstrukturierte Dokumente müssen geeignet transformiert werden, wodurch es auch möglich ist, den Benutzer durch z.B. natürlich sprachliche Anfragen, visualisierte und auch navigierende Anfragesysteme zu unterstützen. Vorteil sind die effizienten Funktionen zur Textanalyse bzw. auch die Möglichkeit der Nutzung der geeigneten Wissensrepräsentationen.

Das Relationenmodell der Datenbanken basiert auf einer formalen mathematisch basierten Semantik, dem auch die Semantik der Anfragesprache, in diesem Fall SQL, entspricht. Dies ermöglicht intern eine Optimierung der Anfrage. Auf eine Anfrage an die Datenbank werden passende Objekte gesucht, deren logische (syntaktische) Struktur exakt mit denen der Anfrage übereinstimmt. Die Anfragesprache setzt zum einen Kenntnisse über die logische Struktur der Daten und zum anderen der Sprache

selbst voraus. Sie ermöglicht aber auch Boolean-Retrieval, wie bereits in Abschnitt 2.1.2.2 auf Seite 6 beschrieben.

Die Betrachtungen in diesem Kapitel sind ziemlich allgemein gehalten. Wenn in der folgenden Arbeit spezielle Konzepte verwendet werden, so werden sie in den entsprechenden Kontext eingeordnet und gegebenenfalls erläutert, bzw. wird auf die entsprechende Literatur verwiesen. Im weiteren Verlauf wird auch ersichtlich, wieso auf Konzepte des IR zurückgegriffen wird bzw. daß die Art der Dokumente spezielle Überlegungen voraussetzt, auch wenn es nicht die Aufgabe dieser Studienarbeit ist, ein IR-System für die Daten (Fakten - strukturierte Daten, Dokumente - unstrukturierte bis semistrukturierte Daten) zu realisieren.

Kapitel 3

Analyse der vorhandenen Daten

Die folgenden Kapitel beschreiben den Entwurf bis zur Implementation der Datenbank für die Domäne TGMV. Die Entwurfsschritte entsprechen in etwa dem allgemeinen Phasenmodell des Datenbankentwurfs. Eine besondere Bedeutung wird im Entwurfsprozeß der Informations- und Konsistenzerhaltung der bereits vorhandenen Anwendungsdaten beigemessen.

In diesem Kapitel wird mit der Informationsanalyse als Voraussetzung für einen erfolgreichen Datenbankentwurf begonnen. In der Informationsanalyse erfolgt die informale Beschreibung des zu entwerfenden Realweltausschnitts. Bei dieser Analyse werden im Zusammenhang mit dem Datenbankentwurf nur die Daten (hier die Informationen aus dem Dateiverzeichnis, evtl. dem Dateinamen und dem Inhalt der Datei) betrachtet. Besondere Berücksichtigung sollen hierbei Daten finden, die entsprechend dem IR eine gewisse Wissensrepräsentation darstellen. Eine Funktionsanalyse wird nicht durchgeführt, da der Entwurf der Datenbank im Vordergrund steht.

3.1 Eigenschaften semistrukturierter Daten

Die zu betrachtenden Daten sind semistrukturiert und somit nicht direkt geeignet, um sofort mit dem Datenbankentwurf zu beginnen. Semistrukturierte Daten besitzen folgende charakteristischen Merkmale [Abi97]:

- Struktur der Daten ist heterogen
  Die innere Struktur der einzelnen Dokumente ist heterogen, da viele Informationen optional sind, teilweise zusätzliche Informationen enthalten sind und auch gleiche Informationen durch unterschiedliche Typen (Preise in DM oder Euro) repräsentiert werden.
- Struktur ist implizit in den Daten enthalten
  Die später beschriebenen Dokumente beinhalten abgesehen von ihrer inneren und äußeren Struktur einen Artikel, eine Anekdote, Chronik, etc. zu einem bestimmten Objekt. Diese Textbestandteile besitzen alle eine bestimmte Struktur, also Strukturinformationen und Beziehungen zwischen den Strukturelementen, die nicht mit den Mitteln der Standarddatenbanken interpretiert oder extrahiert werden können.
- Struktur der Daten ist unvollständig
  Eine vollständige Strukturierung der Daten ist aufgrund der unterschiedlichen Strukturierungsgrade nicht möglich. Beispielhaft seien hier die Öffnungszeiten genannt. Bei privaten Museen sind diese nur telefonisch vereinbar, dann gibt es saisonale Öffnungszeiten bei bestimmten Museen usw..
- Schema ist flexibel (keine strenge Typisierung)
  Aufgrund der Integritätssicherung wird in Datenbanken eine strenge Typisierung der Daten

überwacht. Dies ist für semistrukturierte Daten ziemlich einschränkend. Beispielhaft seinen hier die Zeitangaben der Chroniken genannt. Eine Typisierung auf Datum ist nicht möglich, da teilweise ein vollständiges Datum nicht vorhanden ist oder eine Zeitspanne angegeben wird.

- Schema ist groß (bedingt durch Heterogenität der Daten)
  Bei relationalen Datenbanken wird davon ausgegangen, daß es viele Objekte mit gleicher Struktur gibt und demzufolge die Anzahl der benötigten Schemata entschieden kleiner ist als die Anzahl der Objekte.

- Schema kann sich ändern
  Eine Änderung des Datenbankschema ist sehr aufwendig und deshalb wird davon ausgegangen, daß sich dieses im Verlauf der Anwendung kaum ändert.

- Unterscheidung zwischen Schema und Daten ist unscharf
  In Datenbanken beschreiben die Schemata die Struktur der Daten und Daten sind Instanzen der Schemata. Durch die häufige Änderung des Schema bei semistrukturierten Daten können Schemainformationen zu Daten mutieren und umgekehrt.

Relationale Datenbanksysteme sind geeignet für strukturierte, homogene Daten mit einem festen Schema. Auch wenn sich vielleicht die gegebenen Daten augenscheinlich nicht für relationale Datenbanksysteme eignen, wird im folgenden der Versuch unternommen, durch Trennung der strukturierbaren Informationen in und über die Dokumente von den semistrukturierten Informationen den Entwurf für das gegebene Datenbanksystem zu realisieren.

3.2 Analyse der Dokumentstruktur

Bisher wurden von den Mitarbeitern der EAG ca. 2000 Dokumente elektronisch erfaßt und noch viel mehr gesammelt. Dokumente sind im folgenden semistrukturierte Textdateien im Wordformat und Bilder in unterschiedlichen Dateiformaten sowie Qualitäten. Die Dateien sind in einer 259–teiligen hierarchischen Verzeichnisstruktur gespeichert. Informationen zu den im Entwurfsprozeß betrachteten Entities sind bereits in der Verzeichnisstruktur und dem Dateinamen, der äußeren Struktur enthalten. Die in der äußeren Struktur enthaltenen Informationen stellen in gewisser Weise schon eine Wissensrepräsentation in Form einer Klassifikation dar. Weiterhin existiert ein Schema für die innere Struktur. Diese beiden Strukturen und ihr Informationsgehalt sollen nun getrennt voneinander betrachtet werden.

3.2.1 Äußere Dokumentstruktur

Bisher wurden die Dokumente (Texte und Bilder) jeweils als separate Datei in einem hierarchisch gegliederten Dateiverzeichnisbaum gespeichert. Die allgemeinen Informationen hieraus sind der Dateiname, der den Objektnamen enthält und evtl. auf den Textdokumentinhalt bzw. Bildinhalt schließen läßt und der Verzeichnisname, der ein Klassifikationsschema der Domäne "Technikgeschichte" darstellt. Der globale Dateiverzeichnisbaum "DB-TGMV" ist, wie in Abbildung 3.1 zu sehen ist, bereits auf der ersten Stufe nach verschiedenen Objekten (Technik, Personen, Ereignisse) unterteilt. Hieraus werden später noch unterschiedliche Informationen wie z.B. Entities, Klassifikationsschemata, usw. abgeleitet. Diese Informationen werden im folgenden als äußere Struktur bezeichnet.

Wie bereits erwähnt, stellt diese Verzeichnishierarchie ein Klassifikationsschema dar. Entsprechend der in Abschnitt 2.1.2.1 auf Seite 5 beschriebenen Wissensrepräsentation durch Klassifikationen ist der Verzeichnisbaum monohierarchisch, also durch eine Baumstruktur darstellbar und monodimensional. Das bedeutet, daß auf jeder Stufe nur nach einem bestimmten Merkmal unterschieden wird. Weitere spezielle Eigenschaften ergeben sich aus der folgenden informalen Beschreibung.

Da die vorhandene Dateiverzeichnishierarchie monohierarchisch ist, kann sie als Baumstruktur aufgefaßt werden. In diesem Dateiverzeichnisbaum sind die Dateien in den Knoten und Blättern entsprechend ihrer Einordnung in das Klassifikationsschema abgespeichert.

Die Verzeichnisstruktur kann nicht als allgemeingültige Klassifizierung angesehen werden, wie z.B. die Bücherordnung einer Bibliothek, sondern es ist von eventuell seltenen Änderungen, Erweiterungen oder auch Löschungen auszugehen, wobei die Dateien in dem Verzeichnisbaum neu klassifiziert werden müssen.

Aktuell besteht der vorhandene Verzeichnisbaum aus einer maximalen Tiefe von 9 Knoten. Jeder Knoten hat eine unbestimmte Anzahl von Nachfolgern, wobei die maximale Anzahl von Knoten einer Tiefe 71 beträgt. Die Struktur der ersten Knoten bzw. auch Blätter des Dateiverzeichnisbaumes ist, wie in Abbildung 3.1 angedeutet, gegliedert.

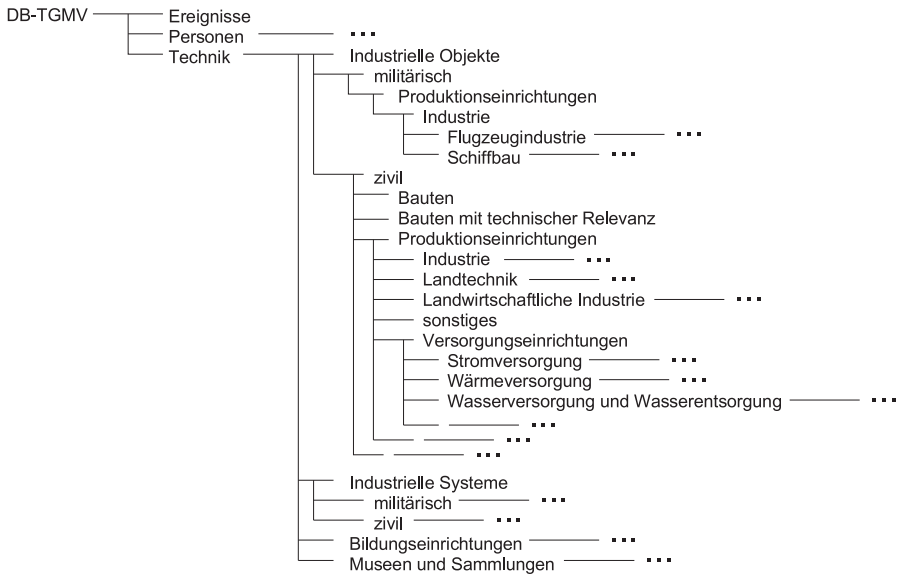

Abbildung 3.1: Ausschnitt aus dem Klassifikationsschema "Technikgeschichte"

Einige Dateien sind mehreren Knoten bzw. Blättern zugeordnet, da sie zu mehreren Klassen gehören. Der vorhandene Verzeichnisbaum enthält deswegen ca. 275 Duplikate. Beispielhaft sei hier genannt, daß ein industrielles Objekt genausogut ein industrielles System sein kann, bzw. kann ein industrielles Objekt auch Teil eines industriellen Systems sein. Außerdem ist es möglich, daß ein Bild gleichzeitig einer Person und einem industriellen Objekt zugeordnet wird. Dies begründet die hohe Anzahl von Duplikaten in den Verzeichnissen.

Eine Tatsache sei an dieser Stelle noch erwähnt. Die Menge der Objekte und somit die Anzahl der Dateien in den einzelnen Verzeichnissen war bisher überschaubar, und so wurden in den Knoten und Blättern des Dateiverzeichnisbaumes keine separaten Objektverzeichnisse angelegt. Dies wäre aber in absehbarer Zeit notwendig geworden, da zu jedem Objekt mehrere Dokumente existieren können und somit die Anzahl der Dateien im Knoten oder Blatt unüberschaubar geworden wäre. Wie bereits erwähnt, enthält der Dateiname zur Zeit die Information "Objektname" und "Dokumentinhalt". Durch Auftrennung des Dateinamen in "Objektname" und Verwendung als Objektverzeichnisname sowie "Dokumentinhalt" als neuer Dateiname im Objektverzeichnis könnte man an dieser Stelle die Dateiverzeichnishierarchie erweitern. Die Zuordnung der Bilder zu einem Objekt ist durch diese Erweiterung auch genauer.

3.2.2 Innere Dokumentstruktur

In der Anwendung gibt es bisher zwei prinzipielle Arten von Dokumenten. Zum einen sind es Textdokumente, gespeichert im Wordformat und zum anderen sind es Bilddokumente, die in unterschiedlichen Formaten und Qualitäten gespeichert sind.

Die Bilddokumente sollen hier erst einmal nicht weiter interessieren, da sie im weiteren Verlauf des Entwurfsprozesses nicht weiter strukturiert werden müssen bzw. direkt übernommen werden, da ihre innere Struktur durch das entsprechende Dateiformat vorgegeben ist und unverändert bleibt. Im Dateiverzeichnisbaum sind die zu einem Objekt gehörenden Bilddokumente im gleichen Knoten bzw. Blatt gespeichert. Meistens wurden die Bilddateien außerhalb der Worddokumente gespeichert, sind aber auch in Worddokumenten enthalten. Die Bilddateien sind häufig redundant, da sie unterschiedlichen Objekten zugeordnet werden können oder das zugeordnete Objekte zu mehreren Klassifikationen gehört.

Die Textdokumente beinhalten Informationen zu technischen Objekten (Gebäude, Fabriken, Fahrzeuge, ...), Personen, die im Zusammenhang mit der industriellen Entwicklung des heutigen Mecklenburg–Vorpommern stehen und Ereignissen, wobei hiermit technikgeschichtliche Zusammenhänge von Regionen oder auch Orten gemeint sind. Die Ereignisse werden im folgenden als Städte bezeichnet. Die Aufarbeitung erfolgte für Objekte, die seit etwa 1800 existieren, soweit sie heute noch zugänglich sind. Die Informationen zu den Objekten der Technikgeschichte unterliegen laufenden Veränderungen (neue Daten, Korrekturen, ...). Die bisherige innere Struktur wurde im Laufe der Erfassung mehrmals an die bestehenden Bedürfnisse angepaßt und unterliegt aktuell dem folgenden Schema:

1. Bezeichnung der Einrichtung
2. Postleitzahl, Straße, Ort, bzw. bei kleineren Orten exakte Beschreibung der Ortslage
3. Allgemeine Beschreibung der Sammlung/Anlage etc.
4. Bezeichnung der (wichtigsten) Bestandteile der Ausstellung/Sammlung
   - 4.1 Name/Bezeichnung
   - 4.2 Jahr, evtl. Ort der Konstruktion und/oder Inbetriebnahme
   - 4.3 Konstrukteur/Erbauer, Erfinder, soweit zugänglich mit Lebensdaten
   - 4.4 wesentliche technische Daten
   - 4.5 überwiegende Nutzung in der Vergangenheit
   - 4.6 wesentliche Veränderungen und Ergänzungen bis zur Gegenwart
   - 4.7 heutiger Zustand
   - 4.8 heutige Nutzung (einschließlich erfolgter Umnutzung)
5. Hinweise auf denkmalgeschützte Anlagen/Bauten
6. Touristisch verwertbare Informationen zum Umfeld (Landschaft, weitere Sehenswürdigkeiten, Zufahrt bzw. Wanderwege), Öffnungszeiten, Eintrittspreise
7. Literaturhinweise: Sekundärliteratur zum Spezialgebiet der Sammlung, Literatur von bzw. über in Frage kommende Persönlichkeiten, vorhandene Prospekte u.a. Informationsmaterial
8. Begebenheiten aus der Geschichte der Anlage/ Einrichtung (auch Historien, Anekdoten)
9. Quellennachweis für Fotos

Die Aspekte der Semistrukturierheit (vgl. Abschnitt 3.1) der Daten sind aus der inneren Struktur ersichtlich. Diese Struktur gilt für alle Dokumente unterschiedlichen Inhaltes, wie z.B. Museen, Städte, Personen und Technik. Dabei werden die eigentlichen Textdokumente zusätzlich um technik–geschichtlich und touristisch relevante Informationen ergänzt. Die jetzige Art und Weise der Speicherung der Dokumente in einem proprietären Dateiformat bringt zusätzliche Probleme mit sich, die bisher in Kauf genommen worden sind, bzw. sich auch nicht vermeiden lassen.

3.2.3 Zusammenfassung der Probleme aufgrund der Dokumentstruktur

Bei der Betrachtung der inneren und äußeren Dokumentstruktur wurden einige Probleme erkannt, die im folgenden zusammengefaßt werden. Dabei wird nicht auf Probleme eingegangen, die automatisch durch die Verwendung eines Datenbanksystems vermieden werden, wie z.B. Probleme mit dem Dateiformat etc..

- Zu jedem Objekt für Städte, Personen und Technik gibt es mehrere Anekdoten, verschiedene Artikel und Bilder und eventuell eine Chronik. Diese werden alle in separaten Dateien zu dem entsprechenden Objekt gespeichert und um weitere Informationen ergänzt. Ein Grund der separaten Speicherung ist die rechtliche Bedingung der Quellenangabe bei der Verwendung fremder Texte und Bilder.

  Jede separate Datei wird entsprechend der inneren Struktur um Informationen ergänzt. Diese Informationen werden als strukturierbare Informationen angesehen und sind überwiegend touristische Ergänzungen. Das größte Problem ist hier die Redundanz, die durch die separate Speicherung der Dateien entsteht.

  Weiterhin entsteht Redundanz durch eine mehrfache Zuordnung eines Objektes im Dateiverzeichnisbaum.

- Bei einer weiteren Erfassung von neuen Objekten, bzw. auch neuen Dokumenten zu bereits vorhanden Objekten, muß der Dateiverzeichnisbaum um den Objektnamen in den entsprechenden Knoten bzw. Blättern erweitert werden, da sonst die Übersicht verloren geht (vgl. Abschnitt 3.2.1).

- Beziehungen zwischen verschiedenen Objekten sind nicht darstellbar, wie z.B. zwischen Personen und technischen Objekten. Der Zusammenhang ergibt sich evtl. aus dem Inhalt des Textes oder könnte auch ähnlich dem Verweis auf zugehörige Bilder angegeben werden. Die Integritätsprüfung der zugeordneten Datei ist nicht realisierbar.

- Da die Bilddateien außerhalb der Textdokumente zum jeweiligen Objekt gespeichert werden, ist es z.Zt. notwendig, im Textdokument die zugehörigen Bilddokumente zu erfaßen und deren Quellenangaben zu vermerken. Andererseits kann man diese Informationen auch im Dateinamen unterbringen, was aber auch keine Vorteile bringt. Hier wäre die Erweiterung des Verzeichnisbaumes um den Objektnamen wie im vorvorherigen Punkt besser. Trotzdem bleibt das Problem mit der Quellenangabe. Eine sogenannte Integritätsprüfung, also eine Prüfung ob die in den Textdokumenten angegebenen Bilddateien existieren, ist nicht durchführbar.

- Die innere Struktur stellt nur ein allgemeingültiges Schema dar und nicht alle Dateien sind nach diesem Schema erstellt. Das jetzige Schema wurde bereits öfter den Bedürfnissen angepaßt. Weiterhin ist zu bemerken, daß dieses Schema für unterschiedliche Objekte (Städte, Personen, technische Objekte) benutzt wird, wodurch viele Punkte aus diesem Schema aus Nullwerten bestehen können bzw. optional sind.

- Die vorhandenen Daten enthalten technisch-geschichtlich relevante und touristische Informationen, aber in der jetzigen Dokumentstruktur ist keine Trennung zwischen den unterschiedlichen Sichten der Nutzergruppen auf die Domäne möglich. Die Sichten sind hier als Informationswunsch des jeweiligen Nutzers zu interpretieren.

Diese Probleme wurden in Kauf genommen, da es sich hierbei nur um eine Übergangslösung handeln sollte. Umso wichtiger ist es also bei der Analyse, keine der vorhandenen Informationen zu verlieren, auf inhaltliche Zusammenhänge zu achten sowie erkannte Probleme zu beseitigen. Vorteile für den Datenbankentwurf ergeben sich insofern, daß schon ausgereifte Schemata, Klassifikationen und natürlich auch Testdaten vorhanden sind.

3.3 Analyse des Dokumentinhaltes

Um mit dem EER–Entwurf zu beginnen, werden zunächst aus der Anwendungsdomäne die verschiedenen Sichten auf die Domäne beschrieben, um die interessierenden Entities und deren Beziehungen untereinander für den EER–Entwurf zu bestimmen. Weiterhin müssen die erkannten Entities und Beziehungen informal beschrieben werden, da sie bestimmte Eigenschaften haben, die später modelliert werden sollen. Die Entities hätte man auch aufgrund ihrer unterschiedlichen Eigenschaften gruppieren können, aber hier werden die Entities über die Sicht auf die Domäne bestimmt und anschließend ihre allgemeinen Eigenschaften bestimmt. Die im folgenden Abschnitt beschriebene Sichtenzentrierung basiert auf der Einteilung der Nutzergruppen nach ihren Interessenschwerpunkten wie in Abbildung 3.2 dargestellt.

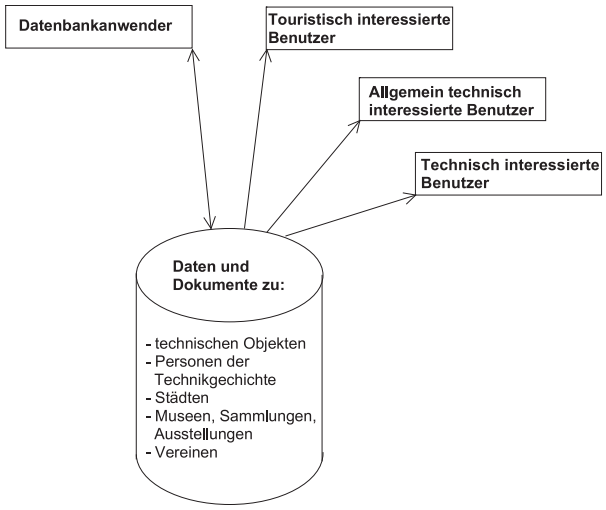

Abbildung 3.2: Nutzerstruktur der Technikdatenbank

Die Datenbankanwender sind Personen, die den Datenbestand administrieren, ändern und erweitern sollen. Sie können zwar differenzierte Rechte besitzen, aber im Gegensatz zu allen anderen Nutzern besitzen sie Schreibrechte zur Datenmanipulation in der Datenbank.

Die anderen Nutzergruppen besitzen ausschließlich Leserecht. Diese Nutzer können von ausschließlich touristisch und wenig technisch interessiert bis zu wenig touristisch und ausschließlich technisch interessiert eingeteilt werden. Berücksichtigung darf dabei natürlich nicht nur die eine oder andere Gruppe finden, indem wir touristische Daten ohne Bezug auf technische Daten speichern, sondern vielmehr sollen die Daten eine Einheit bilden. Das heißt, wenn man sich vorrangig für touristische Informationen interessiert, sollen auch dazu passende technische Hintergrundinformationen abrufbar sein und umgekehrt. Desweiteren sollen auch noch andere Informationen, die nicht direkt einer speziellen Sicht zugeordnet werden können, gespeichert werden. Diese Einteilung aus der Abbildung 3.2 mag zwar ziemlich grob erscheinen, ist aber ausreichend für die Sichtenzentrierung und Ableitung der späteren Entities und Beziehungen.

3.3.1 Sichtzentrierte Analyse

Für die zuvor genannten Nutzergruppen soll die Datenbank konzipiert werden. Aus ihrem Informationswunsch abgeleitet, soll nun der Unterschied zwischen touristischer und technischer Sicht auf die interessierenden Information beschrieben werden. Die Abbildung 3.3 stellt die beschriebene Sichtenzentrierung dar und dient gleichzeitig zur Vorbereitung und zum besseren Verständnis für die konzeptuelle EER-Modellierung im Kapitel 4.1 ab Seite 25.

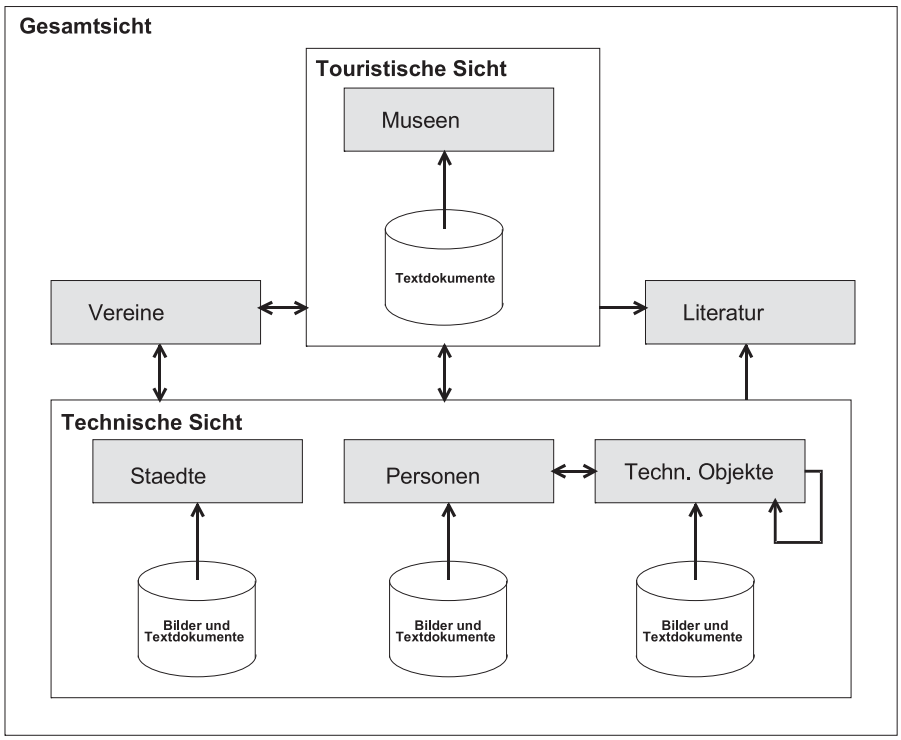

Abbildung 3.3: Sichtenzentrierung

Touristische Sicht

Touristen, Tourismusverbände, Reiseunternehmen etc. interessieren sich für öffentlich zugängliche Museen, Ausstellungen, Sammlungen, Gebäude, wie z. B. Kirchen, Windmühlen, Leuchttürme, technische Baudenkmäler, Stadtinformationszentren oder anderes und in diesem Zusammenhang vorrangig für Informationen wie Öffnungszeiten, Eintrittspreise, Adressen, Telefonnummern, Kurzbeschreibungen der touristischen Attraktionen und evtl. weitere Informationen.

Da jedes Museum eine Menge von technischen Objekten etc. enthält, ist es unmöglich, für alle Objekte technische Beschreibungen und evtl. Bilder in der Datenbank zu speichern bzw. von allen in der Datenbank enthaltenen Objekten die Beziehungen zum entsprechenden Museum herzustellen. Da aber, wie vorhin beschrieben, Beziehungen zwischen beiden Informationsmengen existieren sollen, muß es möglich sein, auf z. B. Hauptausstellungsstücke, die in der Datenbank gespeichert sind, zu verweisen. Wie in der Abbildung 3.3 zu sehen ist, soll die Möglichkeit bestehen, aus der touristischen Sicht von Museen auf Städte, Personen und technische Objekte zu verweisen, wobei dies auch in umgekehrte Richtung funktioniert. Objekte aus der technischen Sicht, die keine Beziehung zur touristischen Sicht haben, da sie z.B. keinem Museum zugeordnet werden können, besitzen trotzdem touristisch verwertbare Informationen, nämlich ihre Ortsangaben.

Die touristischen Informationen sind nach dem im vorigen Abschnitt beschriebenen Schema in un-

terschiedlicher Form enthalten. Zum einen gibt es einzelne Dokumente zu Ausstellungen, Sammlungen, Museen mit allen Informationen. Diese Dokumente enthalten teilweise auch komplette Aufstellungen von Ausstellungsstücken, Anekdoten, etc.. Zum anderen sind die Informationen auch in Dokumenten z.B. zu technischen Objekten in unterschiedlichen Punkten der inneren Struktur oder auch nur als Bemerkung (Referenz) enthalten. Daraus ergibt sich die Beziehung zwischen dem technischen Objekt und einem Museum.

Die Informationen sind semistrukturiert und müssen für den weiteren Datenbankentwurf strukturiert werden. Ausgenommen sind hiervon nur die Beschreibungen, Anekdoten, usw. zu den Museen.

Technische Sicht

Ähnlich wie in der touristischen Sicht sollen hier die interessierenden Objekte dargestellt werden. Weiterhin besteht hier genauso die Frage, welche Informationen die Nutzer vorrangig an der technischen Sicht interessiert. Städte besitzen alle eine Stadtchronik und es existieren eine Menge technisch-geschichtlich relevante Informationen zur Stadtentwicklung. Personen interessieren hier in zweifacher Hinsicht. Zum einen sind es Personen, die aus Mecklenburg-Vorpommern stammen und entweder hier oder wo anders etwas Bedeutendes geleistet haben. Andererseits sind es Personen, die nach Mecklenburg-Vorpommern kamen und hier Bedeutendes geleistet haben. Technische Objekte stellen nun die größte Kategorie dar. Was dazu alles zählt, wird später noch erläutert. Aus diesem Informationswunsch heraus ist die Einteilung der unterschiedlichen Objekte möglich.

Die Art der Beziehung zur touristischen Sicht wurde bereits beschrieben. Weiterhin bestehen innerhalb der technischen Sicht zwischen den Objekten interessierende Beziehungen. Technische Objekte können außer zu ihren Textdokumenten und Bildern untereinander verschiedene Beziehungen eingehen. Denkbare Beziehungen könnten z.B. sein, daß ein technisches Objekt in einem anderen enthalten ist oder auch daß von einer bestimmten Baureihe einer Eisenbahn noch verschiedene Exemplare existieren. Dies wird durch den rekursiven Pfeil in der Abbildung 3.3 ausgedrückt. Bei Personen sind natürlich nicht nur die eigenen Bilder und Textdokumente interessant, sondern vielmehr auch die Beziehung zu den technischen Objekten. Diese Beziehung sollte genauso umgekehrt existieren, da im Zusammenhang mit technischen Objekten oft Personen erwähnt werden.

Ähnlich wie bei der touristischen Sicht stellt sich auch hier weiterhin die Frage, woher die Informationen zu den technischen Objekten kommen und in welcher Form sie vorliegen.

Um die Informationen für die Datenbank zu erhalten, werden alle erdenklichen Quellen wie Zeitungsartikel, Bücher, Archive, u.s.w. genutzt. Es kann somit aufgrund der Vielzahl der Quellen zu einem Objekt (Stadt, Person, technisches Objekt) eine Menge an textuellen Dokumenten geben, die, falls inhaltlich richtig, auch unverändert in der Datenbank gespeichert werden sollen. Bisher wurden diese textuellen Beschreibungen gesammelt, klassifiziert (äußere Struktur) und mit Hilfe des allgemeinen Schema (innere Struktur) um zusätzliche Informationen (touristische, Literatur, Quellennachweis, ...) erweitert. Innerhalb der beiden genannten Strukturen sind starke Redundanzen vorhanden, die beim Datenbankentwurf unbedingt beseitigt werden müssen.

Abgesehen von den inhaltlichen Redundanzen bleiben die textuellen Informationen unverändert. Das hat hier die gleiche Konsequenz wie in der touristischen Sicht, nämlich daß wir semistrukturierte Daten behalten. Zur Darstellung der Objekte in der Datenbank benötigen wir aber strukturierte Daten, die aus den semistrukturierten Dokumenten abgeleitet werden müssen. Dies wird aus späterer Datenbanksicht notwendig, um die Objekte mit gleichen Eigenschaften (Attributen) in einer Relation speichern zu können und wird im Abschnitt 3.3.2 näher informal analysiert.

Gesamtsicht

Zusätzlich zu den speziellen Informationswünschen (touristisch, technisch) sind noch weitere Informationen interessant. Zu den Museen und den anderen Objekten gibt es unterschiedliche Veröffentlichungen, wie z.B. in Büchern, Zeitschriften, etc., die hier als Literatur in der Gesamtsicht aufgenommen werden. Die Literaturangabe dient nicht als Quellenangabe sondern als Bezugsquellenangabe. Viele

Vereine beschäftigen sich mit der Technikgeschichte oder dem Tourismus. Hier sind ihre Beziehungen zu den anderen Objekten interessant. Weiterhin dienen die Informationen zu den Vereinen der Kontaktinformation für Gleichgesinnte. Weitere Einzelheiten zu den beiden Objekten aus der Gesamtsicht und ihren Beziehungen zu den anderen Objekten werden noch genauer im Entwurf beschrieben.

3.3.2 Entityanalyse

Aus der vorhandenen Dokumentanalyse und Sichtenzentrierung wurden bereits unterschiedliche Objekte erkannt, die im formalen Datenbankentwurf Entities genannt werden. Hierzu ist natürlich auch die Einbeziehung der äußeren und inneren Sicht notwendig.

Entityanalyse aus der äußeren Sicht
Die äußere Sicht war der Dateiverzeichnisbaum und der Dateiname. Wie in Abbildung 3.1 auf Seite 16 zu sehen ist, enthalten die ersten Knoten nach dem Wurzelelement "DB-TGMV" die Klassifikationen für die Objekte Ereignisse (*Staedte*), *Personen* und Technik (*technische Objekte*). *Museen* sind dabei unterhalb der Klassifikation von Technik eingeordnet, was korrigiert werden muß. Die hervorgehobenen Objektnamen entsprechen den Objekten der Sichtenzentrierung und sind die interessierenden Objekte der Anwendungsdomäne TGMV.

Der Dateiverzeichnisbaum stellt bereits das Klassifikationsschema für die Anwendung "DB-TGMV" dar. Die folgend hervorgehobenen Wörter sind Klassifikationen einzelner Objekte bzw. Objekte zu denen Klassifikationen existieren und sollen in der Datenbank als Wissen übernommen werden. Ereignisse bzw. Städte wurden bisher nicht weiter klassifiziert. Im Prinzip macht dies auch keinen Sinn, denn aus technischer Sicht lassen sich Städte auch nicht großartig unterteilen, aber aus touristischer Sicht ergeben sich hier interessante Klassifikationen. Interessant ist zum einen die territorialpoltische Einteilung in *Kreise* und zum anderen die Einteilung in touristische *Regionen*. Die Klassifikation der *Personen* wurde bisher noch nicht vorgenommen, kann aber auch nach bestimmten Aspekten wie Qualifikation, Tätigkeit und Erfindung vorgenommen werden. *Technik* wurde bereits aufgrund von Erfahrungswerten ziemlich stark klassifiziert. Klassifikationsschemata sind in Abschnitt 2.1.2.1 auf Seite 5 beschrieben und als eine einfache Art der Wissensrepräsentation zu sehen. Die Einordnung der Objekte ergibt sich aus der Position der Speicherung innerhalb des Verzeichnisbaumes.

Der Dateiname wurde bisher aus dem entsprechenden Objektnamen und dem Inhalt (z.B. Überschrift des enthaltenen Artikels) gebildet. Hier ist zu beachten, daß der Dateiinhalt, sofern es sich um einen Text handelt, unverändert bleiben soll, also semistrukturiert. Aus diesem Grund erfolgt eine strukturelle Aufteilung zwischen den strukturierbaren Informationen zum Objekt und den unveränderbaren semistrukturierbaren Anteilen (Texte, Bilder) die dem Objekt zugeordnet werden. Dies ist auch aus der Abbildung 3.3 ersichtlich.

Entityanalyse aus der inneren Sicht
Entsprechend dem Schema der inneren Struktur (vgl. Seite 17) sind in den Dateien unterschiedliche Informationen enthalten wie touristische Zusatzinformationen, die eigentlichen strukturierbaren Objektinformationen (Name, Ort, ...) und die unstrukturierbaren Informationen (Texte). Desweiteren sind die Beziehungen zu anderen Objekten in unterschiedlicher Form enthalten, teilweise als direkte Objektreferenz oder indirekt innerhalb des Textes. Sofern die Bilder nicht innerhalb der Datei gespeichert wurden, ist die Zuordnung als Referenz innerhalb der Datei vermerkt. Die vorhandenen Informationen und Referenzen bzw. Beziehungen zwischen den Objekten müssen hier erkannt werden, damit sie beim formalen Entwurf berücksichtigt werden können.

Nun folgenden die informalen Beschreibungen der erkannten Entities und ihrer Beziehungen:

- **Technische Objekte**
  Diese Entitymenge umfaßt die größte Gruppe von Objekten. Zu den Objekten zählen z. B. Gebäude, Fabriken, Fabrikationsstätten, technische Bildungs- und Forschungseinrichtungen, techni-

sche Objekte wie z.B. Fahrzeuge, Uhren, etc.. Hierbei muß man berücksichtigen, daß technische Objekte untereinander Beziehungen eingehen können. Alle diese Objekte besitzen Eigenschaften wie Objektname, Objektadresse, Wegbeschreibung etc.. Weiterhin gibt es eine Menge an textuellen Beschreibungen, evtl. eine Historie, im weiteren Verlauf Chronik genannt und Bilddokumente zu jedem technischen Objekt. Alle technischen Objekte sind klassifiziert. Das Klassifikationsschema wurde bereits in Abschnitt 3.2.1 auf Seite 15 beschrieben.

- **Personen**
  Personen, die im Zusammenhang mit der technischen Entwicklung (Geschichte) in Mecklenburg-Vorpommern stehen, haben im Gegensatz zu den technischen Objekten spezielle identifizierende, strukturierbare Eigenschaften, wie z. B. die Namensangaben, Geburtsdaten, etc.. In dem bisherigen Datenbestand wurde aufgrund der geringen Anzahl von erfaßten Personen noch keine besondere Klassifizierung vorgenommen. Da zu den Personeneigenschaften auch ihre Qualifikation, Tätigkeit etc. zählen, wäre es möglich, diese als Klassifikationsschema aufzubauen. Zu beachten ist, daß ähnlich dem Klassifikationsschema der technischen Objekte auch eine mehrfache Klassifikation, so z.B. gleichzeitig mehrere Qualifikationen und Tätigkeiten, möglich sein können. Zu jeder Person können eine Menge von textuellen Beschreibungen, eine Chronik, hier eine Art Lebenslauf und eine Menge an Bildern existieren. Außerdem stehen Personen meist mit technischen Objekten in Beziehung. Zu beachten sind evtl. gewisse Ausnahmefälle, da im Zusammenhang mit der Technikgeschichte manchmal die Rede von Familien oder Geschwistern ist, also nicht von einzelnen Personen. Unberücksichtigt bleibt das Auftreten von Pseudonymen (z. B. bei Künstlern der Künstlername).

- **Städte**
  In den vorhandenen Dokumenten gibt es ein Dateiverzeichnis für Dokumente zu Ereignissen. Gemeint sind damit die technik–geschichtlich–relevanten Informationen zu Städten und Gemeinden. Beziehungen zu technischen Objekten und Personen bestehen nicht explizit, da diese Informationen bereits bei den entsprechenden Entities als Objekteigenschaften vorhanden sind. Charakterisierende Eigenschaften sind hier der Name, die Postleitzahl. Weitere Eigenschaften sind der Kreis und die Region. Diese Informationen könnten als Klassifikationsschema dargestellt werden. Dabei wäre die Klassifikation der Kreise nur einstufig und die der Regionen beliebig strukturierbar. Wenn dieses Klassifikationsschema in der Datenbank allgemein verfügbar ist, wäre über die Adressangaben der anderen Objekte eine weitere Klassifikation für diese Objekte möglich. Zu jeder Stadt können eine Menge von textuellen Beschreibungen, eine Stadtchronik und eine Menge an Bildern existieren.

- **Museen**
  Museen sind, wie bereits erwähnt, in der Dateiverzeichnishierarchie anderes zugeordnet als in der modellierten Sichtzentrierung. Der Begriff Museen steht hier nur allgemein für alles, was irgendwie zu besichtigen ist oder touristische Informationen liefert. Dazu zählen natürlich Museen, private Sammlungen, Ausstellungen, etc., aber auch Stadtinformationszentren. Beziehungen können zu jedem Objekt der technischen Sicht bestehen. Wenn z.B. ein Museum Informationen zu Städten, Personen oder Ausstellungsstücken (technischen Objekten) enthält, die in der Datenbank bereits gespeichert sind, sollte auch die Beziehung innerhalb der Datenbank dargestellt werden. Die charakterisierenden Eigenschaften wurden bereits bei der Beschreibung der touristischen Sicht genannt. Eine Klassifikation scheint hier weniger sinnvoll, eher eine Zuordnung zu einer Beschreibung der Museumsarten. Auch wenn diese z.B. einstufige Beschreibung ähnlich der Kreisklassifikation ist, soll diese bewußt nicht als Klassifikation aufgefaßt werden. Klassifikationen sind Wissensrepräsentationen, werden in diesem Zusammenhang angefragt und liefern im Retrievalprozeß Informationen. Aus diesem Grund seien die Museumsarten hier nur als Eigenschaften aufgefaßt. Zu den Museen existieren auch textuelle Beschreibungen. Das können z.B. Beschreibungen der Sammlung, Auflistung der Ausstellungsstücke, Veranstaltungshinweise, etc. sein.

Die bisher betrachteten Entities wurden bei der Sichtenzentrierung der touristischen oder technischen Sicht zugeordnet. Aus der Gesamtsicht ergeben sich noch weitere zu modellierende Entities.

- **Literatur**
  Die Literaturangaben sollen hier kein Quellennachweis von Textdokumenten sein, sondern mehr der Möglichkeit eines Literaturverweises oder einer Bezugsquellenangabe für Literatur zum jeweiligen Objekt dienen. Die Literatur kann ein Magazin, Zeitschrift, Buch, Tagebuch oder ähnliches sein. Einiges davon läßt sich evtl. noch bestellen, wiederum anderes existiert nur noch als Einzelexemplar oder womöglich auch gar nicht mehr, manches wird nicht mehr publiziert, z.B. ein Großteil der DDR-Literatur. In diesem Zusammenhang ist die Speicherung von Literaturverweisen mit ihren eventuellen Bezugsquellen nützlich. Zu den allgemeinen Eigenschaften der Literatur kommen hier noch zusätzliche Informationen aufgrund der beschriebenen Problematik, wie z.B. Bezugsquellenangaben und die Angabe der Literaturart. Beziehungen existieren zu allen Entities der touristischen und technischen Sicht aus Abbildung 3.3.

- **Vereine**
  Der weitere Datenbankentwurf kann noch um eine Relation für Vereine erweitert werden, da diese im Zusammenhang mit der Technikgeschichte eine wichtige Rolle spielen. Sie führen z. B. Ausstellungen, private Museen oder pflegen und erhalten einzelne technische Objekte. Tourismusverbände können hier auch aufgeführt werden. Im Prinzip können Vereine mit allen beschriebenen Entities, außer mit dem Entity Literatur, in Beziehungen stehen. Die charakterisierenden Eigenschaften sind der Vereinsname, die Adresse, evtl. Kontaktpersonen, Vereinsziel usw.. Zu manchen Vereinen existieren Veröffentlichungen in der Presse. Dies macht es notwendig, textuelle Beschreibungen zum Verein zu speichern.

Die folgende Tabelle soll noch einmal alle erkannten Entities mit ihren Beziehungen zusammenfassen. Nicht betrachtet werden hier die Zuordnung der Bilder und Textdokumente zu den Entities. Weiterhin ist angedeutet, durch welche Klassifikationsschemata das Entity repräsentiert werden kann. Wie bereits beschrieben, können Städte über Kreise und Regionen klassifiziert werden. Dabei wird ein Klassifikationsschema genau durch einen Hierarchiebaum repräsentiert. Zu Personen und technischen Objekten können gleichzeitig mehrere Hierarchiebäume existieren, da sie nach verschiedenen Aspekten klassifiziert werden können.

Entity	**Beziehung zu Entities**
Staedte	Museen, Vereine, Literatur, Klassifikationsschemata zu Staedte
Personen	Museen, Vereine, Literatur, technische Objekte, Klassifikationsschemata zu Personen, Klassifikationsschemata der Staedte
TObjekte	Museen, Vereine, Literatur, Personen, Klassifikationsschemata zu technischen Objekten, Klassifikationsschemata der Staedte
Museen	Staedte, Personen, technische Objekte, Vereine, Literatur Klassifikationsschemata der Staedte
Vereine	Staedte, Personen, technische Objekte, Museen
Literatur	Beziehung ist als Verweis vom Objekt aus zu sehen

Tabelle 3.1: Zusammenfassung der Entities und ihrer Beziehungen

Die hier informal beschriebenen Entities werden im folgenden Kapitel formal durch das EER-Modell[1] beschrieben.

[1] EER-Modell – erweitertes Entity-Relationship-Modell

Kapitel 4

Datenbankkonzeption

Nachdem im vorherigen Kapitel die zu modellierenden Daten gesammelt und informal analysiert wurden, soll in diesem Kapitel die Datenbank konzeptionell entworfen werden. Der konzeptionelle Datenbankentwurf basiert auf den Ergebnissen der Informationsanalyse und ist als unabhängig vom später verwendeten Datenbanksystem zu betrachten.

4.1 Konzeptioneller Entwurf

Ziel dieser Phase ist es, die Datenbank formal, aber unabhängig vom zu verwendenden Datenbanksystem zu beschreiben. Dies geschieht durch Beschreibung der Datenbank durch ein semantisches Datenmodell, wie dem verwendeten EER–Modell.

Weiterhin ist die Modellierung und Integration der verschiedenen anwendungsspezifischen Sichten auf die Daten des Datenmodells Teil des konzeptionellen Entwurfs, sollen aber hier nicht berücksichtigt werden. Beim Entwurf der Technikdatenbank wird von einem einzigen konzeptionellen Gesamtschema für die weiteren Entwurfsschritte ausgegangen, wobei evtl. benötigte Sichten (z.B. touristische, technik–geschichtliche, ...) sich später trotzdem einfach aus dem Gesamtschema ableiten lassen. Dies hat hier keinen Nachteil, denn der Informationswunsch und die eventuell dadurch zu realisierenden Sichten wurden bereits bei der Datenanalyse berücksichtigt. Mit Hilfe der Sichtenzentrierung im Abschnitt 3.3.1 auf Seite 20 wurde der Informationswunsch der zukünftigen Datenbanknutzer konkret analysiert und daraus auf die zu modellierenden Entities geschlossen.

Im Anschluß an den konzeptionellen Entwurf folgt der Verteilungsentwurf. Dieser legt fest, wie die Daten des Gesamtschema unter Berücksichtigung der Beziehungen auf unterschiedlichen Rechnern verteilt in Datenbanken gespeichert werden. Eine Verteilung der Daten, die hier unbedingt berücksichtigt werden muß, ist bisher nicht vorgesehen und wird hier deshalb auch nicht betrachtet. Eine spätere Verteilung der Daten aufgrund verschiedener Aspekte, wie z.B. der Nutzerstruktur (vgl. Abbildung 3.2 auf Seite 19) oder anderer Anwendungserfordernisse, wird im Kapitel 5 diskutiert.

4.1.1 EER-Modell

Zur Beschreibung der Semantik des Datenmodells wird das erweiterte ER–Modell (kurz EER–Modell) verwendet. Dieses Modell basiert auf dem Entity–Relationship-Modell (kurz ER–Modell). Das ER–Modell besitzt Basiskonzepte für Entities, Beziehungen, Attribute und deren Werte, sowie weitere Modellierungskonzepte für funktionale Beziehungen, Schlüsselattribute, abhängige Entities, Spezialisierungen (IST-Beziehung), Kardinalitäten und Optionalität von Attributen und Beziehungen. Die häufig benötigten Konzepte zur Modellierung von Datenbanken sind somit im ER–Modell enthalten, können aber z.B. spezielle Beziehungstypen nur ungenau modellieren. Im EER–Modell sind die unveränderten Basiskonzepte des ER–Modells enthalten, sowie erweiterte Konzepte zur Darstellung

verschiedener Beziehungstypen und Erweiterungen der Möglichkeiten für Attributdefinitionen. Beide Modelle werden ausführlich in [SH97] beschrieben. Hier sollen nur noch einmal kurz die benötigten Konzepte vorgestellt werden:

- **Entity** - Ein Entity ist ein Objekt der Anwendung, zu dem Informationen gespeichert werden sollen, wobei der Entity-Typ eine Gruppierung von Entities mit gleichen Eigenschaften ist und in der graphischen Notation des EER-Modells als Rechteck dargestellt wird.
- **Beziehung** - Die Beziehungen zwischen den Entities werden als Beziehungstypen zusammengefaßt. An einer Beziehung kann eine beliebige Anzahl $n \geq 2$ von Entities teilnehmen. Graphisch wird eine Beziehung im EER-Modell als Raute dargestellt.
  - **Kardinalität** - Die Kardinalität bestimmt die beliebige Teilnahme von Entities an Beziehungen und wird im folgenden Entwurf, wie im Standard–ER–Modell angegeben, in Intervallschreibweise verwendet, also nicht als Teilnahmezahl.
  - **Rollennamen** - Wenn ein Entitytyp mehrfach an einer Beziehung teilnimmt, so wird dies mit Rollennamen an den Kanten graphisch notiert.
- **Attribut** – Die Attribute modellieren Eigenschaften von Entities sowie von Beziehungen. Da alle Entities eines Entity-Typs die gleichen Arten von Eigenschaften haben, besitzen sie auch die selben Attribute, aber nicht die gleichen Attributwerte. Die Attribute werden in der graphischen Notation in Form eines abgerundeten Rechtecks den Entities bzw. Beziehungen direkt zugeordnet. Eine als Kante bezeichnete Linie verbindet die Attribute mit den zugehörigen Entitytypen und Beziehungstypen. Der Übersichtlichkeit halber wurden die Attribute im EER-Modell der Abbildung 4.1 nicht graphisch notiert, sondern sind im Anhang A auf Seite 46 programmiersprachlich notiert (siehe dazu auch Kapitel 4.1.2).
  - **Schlüsselattribute** - Schlüsselattribute sind identifizierende Attribute für Entities und werden beim EER–Modell in der graphischen Notation durch einen gefüllten Kreis am Beginn der Kante dargestellt, bzw. in der erwähnten programmiersprachlichen Notation als *key*-Attribut gesondert ausgewiesen.
  - **optionale Attribute** - Optionale Attribute müssen nicht unbedingt einen definierten Wert (Nullwert) besitzen. Diese Art von Attributen kann unterschiedlich graphisch notiert werden, wobei die Kennzeichnung dieser Attribute durch einen kleinen Kreis an der Kante am gebräuchlichsten ist. Bei der programmiersprachlichen Notation der Attribute ist die Optionalität von Attributen nicht berücksichtigt.
  - **strukturierte Attribute** - Strukturierte Attribute sind zusammengesetzte Attribute, welche aus einzelnen Attributen bestehen, wie z.B. die Adresse. Diese strukturierten Attribute wurden im EER-Entwurf als *record*-Attribute deklariert.
  - **objektwertige Attribute** - Objektwertige Attribute sind Attribute, die Entity-Typen als Wertebereich haben. Sie können nur in Abhängigkeit von den zugeordneten Entities existieren. Dies wird in der graphischen Notation mittels geeigneter Schlüsselnotationen dargestellt und ist in der programmiersprachlichen Notation als *list*- und *set*-Typ definiert.
- **Datenwerte** – Datenwerte sind die primitiven direkt darstellbaren Werte. Wertemengen werden durch Datentypen beschrieben, die die Wertemenge und die darauf gültigen Grundoperationen definieren.

Das EER-Modell enthält weitere Konzepte zur speziellen Darstellung von Beziehungen, wie z.B. die Spezialisierung, Generalisierung, Partitionierung und Aggregierung, die aber zur EER–Modellierung der Datenbank DB-TGMV nicht benötigt werden und hier aus diesem Grund auch nicht erwähnt werden.

4.1.2 EER-Entwurf

In diesem Abschnitt soll das EER–Modell aus Abbildung 4.1 der Anwendungsdomäne TGMV vorgestellt werden. Dazu werden die Entity-Typen, ihre Beziehungen sowie die Attribute der Entity-Typen und Beziehungen mit ihren Datentypen formal beschrieben. Die zu modellierenden Entities und Beziehungen wurden in Abschnitt 3.3 ab Seite 19 durch die informale Dokumentanalyse erkannt und sind in der Tabelle 3.1 auf Seite 24 zusammengefaßt. Die Bilder und Textdokumente zu den Entities wurden bis dato als Attribute der entsprechenden Entities gesehen und sind im EER-Modell geeignet zu modellieren. Die Klassifikationsschemata sind im vorherigen Abschnitt als Wissensrepräsentationen zu den Entities erwähnt und werden hier als Entity-Typen modelliert. Die schon erwähnten Besonderheiten dieses Datenbankentwurfes, speziell die Trennung zwischen semistrukturierten und strukturierbaren Informationen, gilt es ab jetzt zu berücksichtigen. Die Ähnlichkeit der Anordnung der Entities aus Abbildung 4.1 (EER-Modell) zu Abbildung 3.3 (Sichtenzentrierung) auf Seite 20 ist nicht zufällig gewählt und wird auch in Abbildung B.2 (Relationenmodell) auf Seite 63 beibehalten.

Entity- und Attributmodellierung

Zunächst sollen die Entities und ihre Attribute definiert werden. In Kapitel 3.3 wurden aus Sicht der Anwendungsdomäne TGMV die interessierenden Entities ermittelt. Die Entities sind in der Tabelle 3.1 auf Seite 24) zusammengefaßt und bereits entsprechend ihrer Eigenschaften gruppiert. Weiterhin müssen noch die sich aus der Attributmodellierung ergebenden Entities, die Klassifikationsschemata und die bisher noch nicht berücksichtigten Bilder als Entity-Typen modelliert werden.

Aufgrund der Komplexität des Entwurfsdokumentes wurde bei der graphischen Notation der Abbildung 4.1 auf die Angabe der Attribute der Entity-Typen verzichtet. Die Attribute sind im Anhang A auf Seite 46 in einer gebräuchlichen programmiersprachen-ähnlichen Notation für die Typdefinition angegeben. Die Bedeutung der jeweiligen Entity-Typen und Attribute klärt sich durch die Namenswahl, ist aber auch an einigen Stellen kurz erläutert. Jedem Entity-Typ wurden die sich aus der Analyse ergebenden Attribute und benötigten Datentypen zugeordnet.

Die mengenwertigen Attribute, im EER-Modell als strukturierte Attribute bezeichnet, werden hier als *set-*,*list-* und *record*-Typen definiert und vereinfachen die Attributmodellierung, verdeutlichen aber auch die konkreten Anforderungen an die Attribute und die Komplexität einiger Entity-Typen. Der *set*-Typ definiert endliche Mengen von Werten ohne Duplikate, der *list*-Typ definiert Listen von Werten und der *record*-Typ soll das imperative Pendant zum *tuple*-Konstruktor sein. Diese Modellierung wurde im EER-Modell nicht beibehalten, sondern durch objektwertige Attribute bzw. neue Entity-Typen semantisch ersetzt.

Jeder *record*-Typ der Attributmodellierung wird im EER-Modell bereits als separater Entity-Typ angegeben und enthält die selben Attribute, aus denen ein Schlüsselattribut gewählt werden muß. Semantisch bedeutet dies, daß zu jedem vorherigen strukturierten Attribut jetzt genau ein Tupel im neuen Entity gespeichert wird, vorausgesetzt, daß beim *record*-Typ kein Nullwert erlaubt war. Dies ist bei der Angabe der Kardinalitäten der Beziehung berücksichtigt.

Die Kombinationen aus dem *set-* oder *list-* und dem *record*-Typ werden im EER-Modell als objektwertige Attribute modelliert, was durch eine geeignete Schlüsselnotation im EER-Modell dargestellt wird. Objektwertige Attribute haben Entity-Typen als Wertebereich. An dieser Stelle wird die eigentliche Trennung zwischen strukturierten und semistrukturierten Daten vorgenommen, wobei dies auch nur eingeschränkt gilt, was in diesem Abschnitt noch beschrieben wird. Die Entity-Typen mit dem objektwertigen Attribut, z.B. *STAEDTE, PERSONEN,* etc., enthalten dabei die strukturierten Daten und die objektwertigen Attribute die semistrukturierten Daten, natürlich ergänzt um identifizierende "strukturierte" Attribute, was durch den *record*-Typ modelliert wird. Der *set*-Typ wird verwendet für Textdokumente und der *list*-Typ für Chroniken. Jedes Entity mit dem objektwertigen Attribut für Textdokumente kann beliebig viele Texte (ohne Duplikate) haben. Die Entities mit dem objektwertigen Attribut für Chroniken können jeweils nur eine Chronik besitzen. Die Chronik besteht aus einer chronologisch geordneten Liste von Werten. Die Mengenwertigkeit des jeweiligen objektwertigen At-

tributes wird im EER-Modell durch die Wahl der Kardinalitäten genau festgelegt.

Die Attribute mit den Datentypen *textstring* und *imagestring* enthalten die semistrukturierten Daten. Wie bereits erwähnt, gibt es trotzdem noch Attribute, die semistrukturierte Daten enthalten. Dies betrifft einige Attribute mit dem Datentyp *string*, die Informationen enthalten, wie z.B. Medienhinweis, Bezugsquelle, Touristinfo, Anfahrt, Öffnungszeiten, etc.. Die Informationen stellen aufgrund der thematischen Ausrichtung der Anwendungsdomäne TGMV hier nur Nebeninformationen zum Datenbestand dar und werden deswegen nicht weiter strukturiert. Die Probleme, die sich daraus ergeben, werden in Abschnitt 6.2 auf Seite 45 genannt.

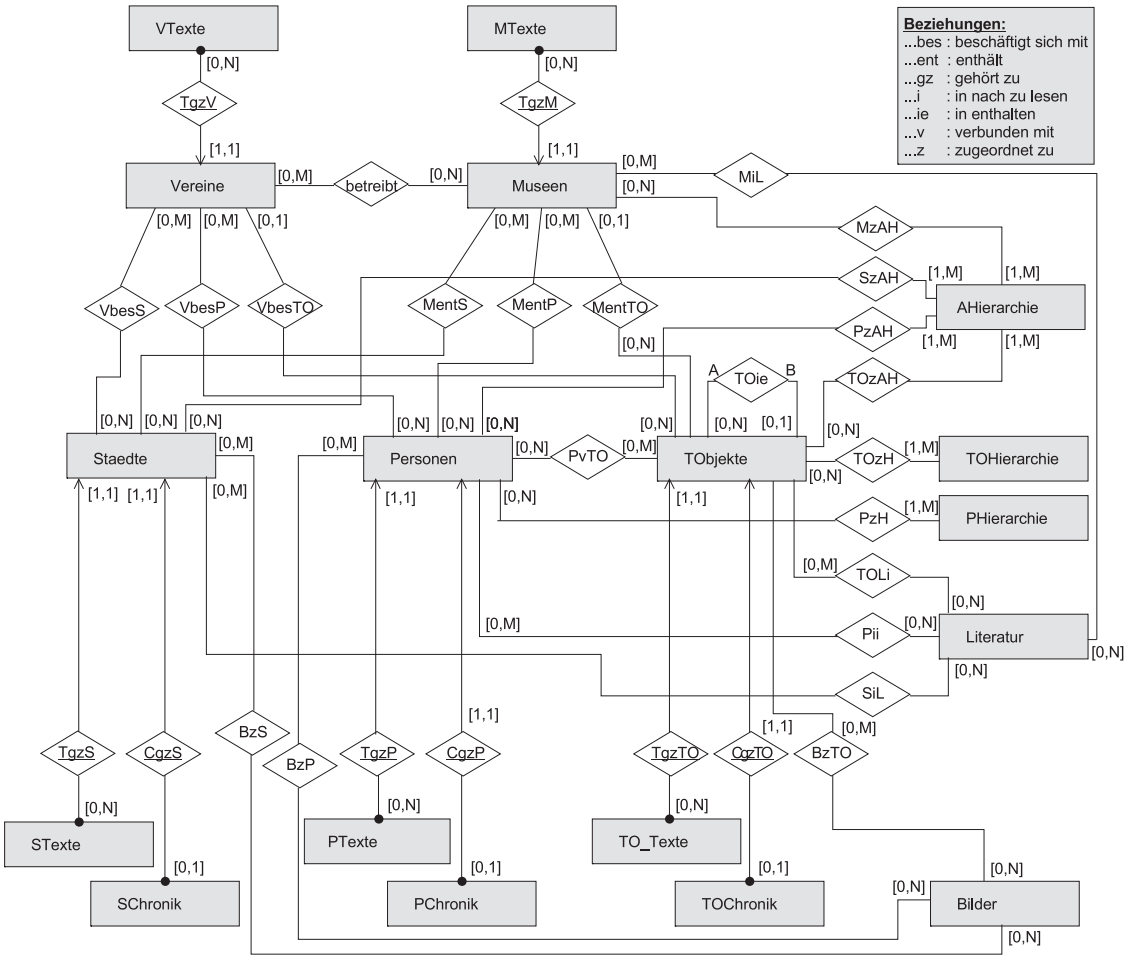

Abbildung 4.1: EER-Diagramm

Da die Klassifikationsschemata und Bilder bis jetzt noch nicht als Entity-Typen berücksichtigt wurden, sollen diese jetzt noch beschrieben werden.

Für jedes in der Tabelle 3.1 auf Seite 24 unter Beziehung zu Entities genannte Klassifikationsschema wird ein eigener Entity-Typ modelliert. In jedem Entity-Typ für ein Klassifikationsschema soll es möglich sein, mehrere Klassifikationsbäume abzuspeichern. Bei der Beziehungsmodellierung ist zu berücksichtigen, daß ein Entity jeweils innerhalb eines Hierarchie-Entity-Types mehreren Klassifikationen innerhalb eines Baumes und auch gleichzeitig mehreren Bäumen zugeordnet werden kann.

Bemerkt sei hier nochmal, daß jeweils für Adressangaben, Personen und technische Objekte ein separater Hierarchie Entity-Typ erstellt wird. Jedes dieser Objekte kann nach mehreren Aspekten klassifiziert werden, die als verschiedene Klassifikationsbäume im Entity-Typ Hierarchie enthalten sind. Die Attribute der Klassifikationsschemata sind der Klassifikationsname, die Klassifikationsstufe und ein manueller Schlüssel, der sich aus der Baumnummerierung ergibt. Der Schlüssel, auch Objektidentifier (kurz OID) genannt, ist eine Kombination aus der Baumwurzel-, der Baumknoten- und der Baumblattnummer.

Der Entwurf für die Bilder ist in dieser Studienarbeit nur als schemenhafter Ansatz angedacht. Bisher existieren zwar einige Bilder, aber diese liegen in ungünstigen Bildformaten und in ungenügender Qualität vor. Bilder werden im Gegensatz zu der objektwertigen Attributmodellierung von Texten als ein Entiy-Typ modelliert. Diese Modellierung ist bedingt durch die Anforderung, daß ein Bild mehreren Entity-Typen (*STAEDTE, PERSONEN, TOBJEKTE*) und zugleich mehreren Entities (aus diesen Typen) zugeordnet werden kann. Im Gegensatz dazu gehört ein Text immer nur zu einem Entity eines Entity-Types. Zu einem Bild werden "strukturierte" Attribute definiert, die später von der Anwendung zur Auswahl der Bilder benötigt werden. Dafür ist der Bildtyp notwendig. Die Bildgröße spielt eine Rolle, wenn das Bild nicht in Originalgröße angezeigt werden soll oder wenn eine bestimmte Bildgröße benötigt wird. Auf diese Probleme wird in Abschnitt 6.2 auf Seite 45 nochmal eingegangen.

Beziehungsmodellierung

Beziehungen zwischen Entities werden zu Beziehungstypen zusammengefaßt. Durch die Angabe von Kardinalitäten wird die Teilnahme der Entities am Beziehungstyp festgelegt, bzw. eingeschränkt. In der Tabelle 4.1 sind alle Beziehungen des EER-Modells aus Abbildung 4.1 nochmal zusammengefaßt. Wie in Abschnitt 4.1.1 bereits erwähnt, werden die Kardinalitäten hier wie im Standard EER-Modell in Intervallnotation verwendet und nicht als Teilnahmezahl.

Tabelle 4.1: Beziehungen und Kardinalitäten der Entity-Typen

Beziehungen	**beteiligte Entity-Typen**	**Kardinalitäten**
bes – beschäftigt sich mit	Vereine – Staedte	$[0,M]-[0,N]$
	Vereine – Personen	$[0,M]-[0,N]$
	Vereine – TObjekte	$[0,1]-[0,N]$
betreibt	Vereine – Museen	$[0,M]-[0,N]$
ent – enthält	Museen – Staedte	$[0,M]-[0,N]$
	Museen – Personen	$[0,M]-[0,N]$
	Museen – TObjekte	$[0,1]-[0,N]$
gz – gehört zu	Staedte – STexte	$[1,1]-[0,N]$
	Personen – PTexte	$[1,1]-[0,N]$
	TObjekt – TO_Texte	$[1,1]-[0,N]$
	Vereine – VTexte	$[1,1]-[0,N]$
	Museen – MTexte	$[1,1]-[0,N]$
	Staedte – SChronik	$[1,1]-[0,1]$
	Personen – PChronik	$[1,1]-[0,1]$
	TObjekt – TOChronik	$[1,1]-[0,1]$
i – in nach zu lesen	Staedte – Literatur	$[0,M]-[0,N]$
	Personen – Literatur	$[0,M]-[0,N]$
	TObjekte – Literatur	$[0,M]-[0,N]$
	Museen – Literatur	$[0,M]-[0,N]$
ie – in enthalten	TObjekt-A– TObjekte-B	$[0,N]-[0,1]$
v – verbunden mit	Personen – TObjekte	$[0,N]-[0,M]$

Fortsetzung der Tabelle auf der nächsten Seite

Tabelle 4.1: Fortsetzung der Tabelle 4.1

Beziehungen	**beteiligte Entity-Typen**	**Kardinalitäten**
z – zugeordnet zu	TObjekte – TOHierarchie	$[0,N]-[1,M]$
	Personen – PHierarchie	$[0,N]-[1,M]$
	TObjekte – AHierarchie	$[0,N]-[1,M]$
	Personen – AHierarchie	$[0,N]-[1,M]$
	Staedte – AHierarchie	$[0,N]-[1,M]$
	Museen – AHierarchie	$[0,N]-[1,M]$
	Bilder – Staedte	$[0,N]-[0,M]$
	Bilder – Personen	$[0,N]-[0,M]$
	Bilder – TObjekte	$[0,N]-[0,M]$

Aus der Tabelle 4.1 ergeben sich die folgenden unterschiedlichen Beziehungen, wobei R allgemein für die Beziehungen steht und E_1, E_2 die an der Beziehung teilnehmenden Entities darstellen.

- $R\big(E_1[0,M], E_2[0,N]\big)$

  Dieser Beziehungstyp folgt keiner Einschränkung.

- $R\big(E_1[0,N], E_2[1,M]\big)$

  E_1 wird mindestens einem Entity aus E_2 zugeordnet. Umgekehrt erfolgt keine Einschränkung von E_2 auf E_1. Dieser Beziehungstyp modelliert die Zuordnung von Entities (E_1) zu ihrem Klassifikationsschema (E_2).

- $R\big(E_1[0,1], E_2[0,N]\big)$

  Entity E_1 und E_2 können unabhängig voneinander existieren. Jedes Entity aus E_2 kann höchstens mit einem Entity aus E_1 in Beziehung stehen. Umgekehrt existiert keine Einschränkung. Dieser Beziehungstyp modelliert die Zuordnung von technischen Objekten (E_2) und Museen (E_1) bzw. Vereinen und außerdem die rekursive Beziehung von technischen Objekten A (E_2) zu technischen Objekten B (E_1). Hierbei kann Objekt A höchstens einmal in Objekt B enthalten sein, aber Objekt B mehrere Objekte A enthalten.

- $R\big(E_1[1,1], E_2[0,N]\big)$

  Jedem Entity E_2 ist genau ein Entity E_1 zugeordnet. Umgekehrt existiert keine Einschränkung. Dieser Beziehungstyp modelliert das objektwertige Attribut Texte. Hierbei kann ein Text (E_2) nicht ohne das zugehörige Entity (E_1) existieren und wenn, dann nur genau einem Entity zugeordnet werden.

- $R\big(E_1[1,1], E_2[0,1]\big)$

  Jedem Entity E_2 ist genau ein Entity E_1 zugeordnet. Umgekehrt kann jedem Entity aus E_1 höchstens ein Entity aus E_2 zugeordnet werden. Dieser Beziehungstyp modelliert das objektwertige Attribut Chronik. Hierbei ist klar, das zu jedem Entity E_1 höchstens eine Chronik E_2 existiert und wenn, dann kann diese nur genau zu einem Entity gehören.

Das EER-Modell in Abbildung 4.1 ist jetzt vollständig formal beschrieben und somit das konzeptionelle Gesamtschema für die Anwendungsdomäne TGMV. Dabei wurden alle in den Abschnitten 3.3, 3.3.1 und 3.3.2 erkannten Informationen berücksichtigt. An den konzeptionellen Entwurf schließt sich jetzt der logische Entwurf an, da wie bereits in Abschnitt 4.1 erwähnt, kein Verteilungsentwurf notwendig ist.

4.2 Relationaler Entwurf

Das EER-Modell diente der semantischen Analyse der Anwendungsdomäne. Die Modellierung ist unabhängig vom Datenbanksystem, das heißt, daß auf systemspezifische Aspekte noch nicht eingegangen wurde. Der logische Entwurf, hier relational aufgrund der Verwendung des relationalen Datenbanksystems Sybase, berücksichtigt nur den Aspekt der Art des Datenbanksystems und gilt insoweit auch noch als unabhängig vom Datenbanksystem. Das Ergebnis des relationalen Entwurfs sind Relationen. Um diese Relationen zu erhalten, wird zunächst das EER-Modell in das Relationenmodell transformiert. Anschließend müssen diese Relationen optimiert und die Transformationseigenschaften sowie Schemaeigenschaften validiert werden. In diesem Zusammenhang werden nur benötigte Konzepte kurz genannt, sofern dies zum Verständnis notwendig ist. Ausführliche theoretische Grundlagen sind in [SH97] nachzulesen.

4.2.1 Abbildung vom EER-Modell auf das Relationenmodell

Diese Transformation ist weitestgehend unproblematisch und im Prinzip auch automatisierbar. Aus diesem Grund ist es auch möglich, beginnend beim EER-Entwurf mit Case-Tools[1] zu arbeiten. Das bei dieser Studienarbeit benutzte und in Abschnitt 5.1 auf Seite 35 beschriebene Case-Tool von Sybase unterstützt den Datenbankentwurf nach dem logischen Entwurf, so daß die folgend beschriebenen Schritte noch manuell durchgeführt werden müssen. Aufgrund der Komplexität des hier enstehenden Relationenmodells wird dieses hier nicht separat in Form einer Grafik bzw. im Anhang in Tabellenform aufgeführt. Im Anhang in Abbildung B.1 ab Seite 64 ist das Physische Datenmodell (PDM) dargestellt. Dieses basiert auf dem folgenden Relationenmodell. In dem PDM sind spezifische Implementationsaspekte entsprechend dem verwendeten Datenbanksystem Sybase enthalten, die erst im nächsten Kapitel beschrieben werden. Abstrahiert man von diesen, so ist es das folgend aus dem EER-Modell transformierte Relationenmodell. Damit die Bezeichnung PDM zu keinen Irritationen aus Datenbanksicht entsprechend der Drei-Ebenen-Schema-Architektur führt, soll dies genauer in Kapitel 5 auf Seite 35 diskutiert werden.

Entity-Transformation

Jeder Entity-Typ des EER-Modells wird mit seinen Attributen jeweils auf ein Relationenschema abgebildet. Die Schlüssel werden übernommen und wenn mehrere vorhanden sind, einer als Primärschlüssel ausgewählt. Der Primärschlüssel soll möglichst aus einem Attribut bestehen, wobei ein *integer*-Attribut einem *string*-Attribut vorzuziehen ist.

Beziehungstransformation

Jeder Beziehungstyp des EER-Modells wird jeweils auf ein Relationenschema abgebildet. Wenn dieser Beziehungstyp eigene Attribute besitzt, was nur bei *TO_IN_TO_ENTHALTEN* der Fall ist, werden diese zu Attributen des Beziehungstyps. Zusätzlich werden die Primärschlüssel der beteiligten Entity-Typen als Attribute in den Beziehungstyp übernommen. Die Kardinalität der Beziehung bestimmt die Wahl des Primärschlüssels des Beziehungstyps. Die übernommenen Schlüssel werden als Fremdschlüssel gekennzeichnet. Später werden auf ihnen verschiedene Fremdschlüsselbedingungen definiert. Die Primärschlüssel der in Abschnitt 4.1 definierten Beziehungstypen werden wie folgt gewählt:

- $R\big(E_1[0,M], E_2[0,N]\big)$

  In dem neuen Relationsschema werden die beiden Primärschlüssel der beteiligten Entity-Typen als Schlüssel gewählt.

- $R\big(E_1[0,N], E_2[1,M]\big)$

  In dem neuen Relationsschema werden die beiden Primärschlüssel der beteiligten Entity-Typen als Schlüssel gewählt. Die zwingende Beziehung von einem Entity aus E_1 mit mindestens einem

[1] CASE-Tool – Computer Aided Software Engineering-Werkzeug

Entity aus E_2 ist hier nicht durch Wahl der Schlüssel oder durch verschmelzen von Relationsschemata lösbar. Dies muß später unbedingt als zusätzliche Integritätsbedingung durch die Anwendung realisiert werden.

- $R(E_1[0,1], E_2[0,N])$

  Der Primärschlüssel der N-Seite, hier also der vom Entity E_2, wird Schlüssel in dem neuen Relationsschema.

- $R(E_1[1,1], E_2[0,N])$ und $R(E_1[1,1], E_2[0,1])$

  Bei zwingenden Beziehungen wird das neue Relationsschema des Beziehungstyps mit dem Relationsschema der N- bzw. der optionalen Seite, hier also dem vom Entity E_2, verschmolzen.

4.2.2 Verbesserung der relationalen Schemata

Das bisher erhaltene Relationenmodell enthält noch nicht die notwendigen Eigenschaften, um es in ein bestimmtes relationales Datenbanksystem zu übertragen. Zunächst müssen die Schemaeigenschaften durch diverse Normalisierungen angepaßt werden. Der Schwerpunkt der Optimierung soll nicht die Erreichung aller Schemaeigenschaften sein, sondern eher ein Kompromiß zwischen Erreichung von notwendigen Schemaeigenschaften und einer guten Datenbankperformance. Weiterhin sind die Transformationseigenschaften zu validieren. Zuvor sollen aber noch einige Probleme und die sich daraus ergebenden Änderungen im Relationenmodell beschrieben werden.

Probleme und Änderungen im Relationenmodell

Die beschriebenen Änderungen sind in Abbildung B.1 auf Seite 64 bereits enthalten.

- Die rekursive Beziehung *TO_IN_TO_ENTHALTEN* enthält die beiden Primärschlüssel der beteiligten Entity-Typen, in diesem Fall zwei Primärschlüssel mit gleichem Namen. Dies ist nicht erlaubt und aus diesem Grund wird der vorhandene Name um den Rollennamen ergänzt.

- Die Entity-Typen *TO_IN_TO_ENTHALTEN*, *MUSEEN*, *LITERATUR*, *BILDER* enthalten ein Attribut, welches auf ein Unterscheidungsmerkmal, wie z.B. Museumsart, Bildtyp, etc., hinweist. Dies soll kein Klassifikationsmerkmal entsprechend einer Wissensstruktur sein, soll aber zur genauen Unterscheidung der vorhanden Entities dienen. Das Problem ist, wenn man zu jedem Entity das Merkmal als "String-Wert" eingibt, ist der Aufwand und die Gefahr von Tippfehlern hoch. Wenn man extern bestimmte Schlüssel für bestimmte Merkmale vergibt, muß man die Attributwerte überprüfen, so z.B. mit der Checkklausel, ob der Attributwert in dem festgelegten Schlüsselbereich liegt. Eine Erweiterung des Schlüsselbereiches kann dabei mit großem Aufwand verbunden sein. Die flexibelste Lösung ist, die Unterscheidungsmerkmale in einem separaten Relationsschema aufzunehmen und im Ausgangsentity per Fremdschlüsselbeziehung zu referenzieren.

- Die Entity-Typen *VEREINE*,*V_K_KONTAKTPERSON*,*MUSEEN* und *TOBJEKTE* besitzen alle Adressenangaben, die entweder bei dem Entity-Typ selbst oder besser jeweils in einem separaten Entity-Typ gespeichert werden könnten. Aufgrund der globalen Minimalitätseigenschaft (s.a. Schemaeigenschaften) ist es sinnvoll, alle Adressenangaben in einer Relation *ADRESSEN* zusammenzufassen. Dies hat folgende Vorteile und Nachteile:

  + globale Redundanzvermeidung, da z.B. technische Objekte teilweise die selbe Adresse wie ein Museum besitzen, das gleiche trifft z.B. für Heimatverein und Heimatmuseum zu, ...
  + globale Redundanzvermeidung, da weniger Relationen wegen einmalig notwendiger Zuordnung der Adressenklassifikation
  + Vereinfachung des Gesamtschema und dadurch einfachere Anfragen
  – unterschiedlich genaue Adressenangaben der einzelnen Entity-Typen und dadurch Probleme bei Überprüfung der lokalen Integritätsbedingungen (vgl. Abschnitt 5.2 ab Seite 36)

- Die Trennung der Adressenklassifikationen in Kreise und Regionen ist aufgrund der unterschiedlichen Kardinalitätsbedingungen zwischen territorialpolitischer und touristisch-regionaler Gliederung zu den Adressen notwendig und wurde bei der Zuordnung zu den Entity-Typen *ADRESSEN* und *STAEDTE* berücksichtigt.

Schemaeigenschaften

Das Entwurfziel ist die Erreichung der dritten Normalform als lokale Anforderung an die Datenbankschemata, welche die erste und zweite Normalform impliziert, und die Minimalität als globale Anforderung an die Datenbankschemata.

Die erste Normalform, also Speicherung von atomaren Attributen, wird von allen Relationen erfüllt. Die Attribute, die semistrukturierte Daten (Texte, Bilder) enthalten, sollen hier einfach als ein atomarer Attributwert betrachtet werden, da es keinen Sinn macht, z.B. einen zusammenhängenden Artikel zu zerlegen. Attribute, welche nicht vollständig oder nur mit sehr großem Aufwand strukturierbar sind (vgl. Abschnitt 3.1 auf Seite 14 und Abschnitt 6.2 auf Seite 45), bleiben semistrukturiert. Durch die Beseitigung von mehrwertigen Abhängigkeiten, z.B. in der Relation *CHRONIK* und *LITERATUR*, entstehen neue Redundanzen, die durch weitere Normalisierungsschritte beseitigt werden müssen.

Mit Hilfe der Theorie über funktionale Abhängigkeiten lassen sich lokale Redundanzen erkennen, die hier nur zum Teil beseitigt werden sollen.

Die zweite Normalform verbietet partielle Abhängigkeiten eines Nichtprimattributes von einem Schlüssel und entsteht unter anderem durch die Einführung der ersten Normalform. Diese sind durch Abtrennung des partiell abhängigen Schlüsselbestandteiles beseitigt worden. In der Abbildung B.1, auf die im Zusammenhang mit dem Relationenmodell verwiesen wird, sind nur noch einfache Schlüssel vorhanden. Mit dem natürlichen Schlüssel der jeweiligen Relation (vgl. Attributmodellierung in Anhang A ab Seite 46) wäre die zweite Normalform auch erfüllt.

Die dritte Normalform verbietet transitive Abhängigkeiten eines Nichtprimattributes von einem Schlüssel. Dies wird z.B. von den Relationen *ADRESSEN* und *LITERATUR* in der Abbildung B.1 nicht erfüllt. Eine weitere Zerlegung dieser Relationen ist aber aus Zugriffsgründen nicht erwünscht.

Durch Normalformen wurde versucht, innerhalb der einzelnen Relationen Redundanzen zu vermeiden. Die Minimalitätsanforderung soll unter Beibehaltung der vorherigen Schemaeigenschaften globale Redundanzen vermeiden. Einfach ausgedrückt heißt das, trotz lokaler Redundanzvermeidung möglichst wenige Relationen zu erhalten. Diese Eigenschaft ist erreicht. Vorstellbar wäre vielleicht eine andere Modellierung des Entity-Typs *CHRONIK* (vgl. Abschnitt 5.2 ab Seite 36), um die Anzahl der Relationen zu verringern, aber eine globale Redundanz existiert auch ohne diese Änderung nicht.

Transformationseigenschaften

Zu den Transformationseigenschaften beim logischen Datenbankentwurf gehören die Abhängigkeitstreue und die Verbundtreue. Eine vollständige Validation der Transformationseigenschaften würde hier den Rahmen sprengen und wird auch nicht verfolgt. Trotzdem sind einige Aspekte wichtig.

In der Datenbank sollen alle Daten und zugehörige mehrwertige und funktionale Abhängigkeiten semantisch äquivalent durch Integritätbedingungen, also Schlüssel und Fremdschlüssel, beschrieben werden. Wie bereits zuvor erwähnt, werden im Datenbankschema nicht die natürlichen Schlüssel verwendet, aber trotzdem sollen die gegebenen Integritäten gewährleistet werden. Dies muß bei der Datenbankimplementation in Abschnitt 5.2 ab Seite 36 berücksichtigt werden, z.B. durch geeignete Wahl von Alternativschlüsseln.

Bedingt durch Normalformen wurden Relationen teilweise in kleinere Relationen aufgespalten. Die Verbundtreue verlangt die Wiederherstellung der Originalrelationen durch den natürlichen Verbund. Dieses Kriterium kann als erfüllt betrachtet werden, da keine Verbund verletzenden Zerlegungen von Ausgangsrelationen durchgeführt wurden.

Das Ergebnis des logischen Entwurfes ist das Relationenmodell, das immer noch als unabhängiges Modell von einem relationalen Datenbanksystem betrachtet werden kann. Die nächsten Entwurfsschritte enthalten datenbanksystem-spezifische Aspekte und werden im folgenden Kapitel beschrieben.

Kapitel 5

Implementation der Technikdatenbank

In diesem Kapitel wird, basierend auf den Ergebnissen der Datenanalyse und der Datenbankkonzeption, die Datenbankimplementation beschrieben. Zur Datenbank Sybase wird das Case-Tool PowerDesigner® SQL Modeler for Sybase 6.1 (kurz SQL-Modeler) mitgeliefert, welches bei der Datenbankimplementation verwendet wurde. Der SQL-Modeler wird im nächsten Abschnitt vor der eigentlichen Datenbankimplementation kurz vorgestellt. Das Kapitel schließt mit einem Anwendungsszenario, in dem die hier entwickelte Datenbank in einem noch zu realisierendem Gesamtkonzept dargestellt wird. Zum Datenbanksystem war kaum Literatur zu finden und aktuelle schon gar nicht. Dafür liefert Sybase eine umfangreiche Dokumentation in Form von Onlinebooks mit dem Datenbanksystem, von denen [Syb99c], [Syb99b] und [Syb99a] genutzt wurden.

5.1 Case-Tool - SQL Modeler for Sybase 6.1

Die allgemeine Aufgabe der Case-Tools ist es, mit Hilfe von Werkzeugen den Entwurfsprozeß bei der Softwareentwicklung - hier Datenbankentwicklung - zu unterstützen. Dabei kann man zwischen Case-Tools und Case-Environments unterscheiden. Die Case-Environments sind eine integrierte Sammlung von Case-Tools, die alle Phasen des sogenannten Software-Lebenszyklus im Zusammenhang sehen und dafür Werkzeugunterstützung bieten. Demnach ist der SQL-Modeler ein Case-Tool, da er, wie bereits in Abschnitt 4.2.1 auf Seite 31 erwähnt, nicht alle Phasen der Datenbankentwicklung unterstützt, sondern nach dem Phasenmodell für den Datenbankentwurf (vgl. [SH97] Kapitel 4) erst nach dem logischen Entwurf einsetzbar ist. Viele Case-Tools für den Datenbankentwurf bieten bereits eine Unterstützung ab dem konzeptionellen Entwurf an, wie z.B. DBMAIN, ERwin, etc., aber ein Vergleich zwischen Case-Tools bzw. zu Case-Environments soll hier nicht durchgeführt werden. Vielmehr soll hier nur kurz auf den Funktionsumfang des SQL-Modeler eingegangen werden. Im Onlinebook [Syb99c] wird der SQL-Modeler beschrieben.

Das mit dem SQL-Modeler erstellte Modell wird als "Physisches Datenmodell" (s.a. Abschnitt 4.2.1 auf Seite 31) bezeichnet. Die Bezeichnung wird vom SQL-Modeler übernommen und soll auf die datenbank-spezifischen Aspekte - hier zuerst die Datendefinition nach dem logischen Entwurf - hindeuten. Vielfach wird mit Beginn der Datendefinition vom physischen Entwurf gesprochen. Aus Datenbanksicht (Drei-Ebenen-Architektur) gehören dieser Schritt sowie die Datenbankimplementation mit Hilfe der SQL-DDL[1] aber zur konzeptuellen Ebene und der physische Entwurf beginnt erst mit der Definition der internen Ebene der Datenbank durch die SQL-SSL[2] (siehe dazu auch [SH97] und [SH99]).

Bei der Datenbankimplementation der Datenbank DB-TGMV wurde der SQL-Modeler zur Datendefinition und Scriptgenerierung genutzt. Auch wenn das generierte Script letztendlich nicht direkt

[1] SQL-DDL – Data Definition Language
[2] SQL-SSL – Storage Structure Language

übernommen wurde, war die Verwendung des SQL-Modulers beim Ändern und Testen der Datenbank sehr hilfreich. Der verwendete graphische Editor unterstützt eine einfache und schnelle Modellierung und läßt sich in den wesentlichsten Eigenschaften anpassen. Das erstellte Modell läßt sich beliebig skaliert graphisch ausgeben. Eine automatische Versionskontrolle existiert nicht, ist aber mit etwas Aufwand manuell über ein Modelleigenschaftsfenster realisierbar. Die Systemanforderungen des SQL-Modelers sind gering. Da der SQL-Modeler noch mehr als die genutzten Funktionalitäten bietet, werden die wesentlichsten Funktionalitäten in der folgenden Tabelle nochmal zusammengefaßt.

Funktionalität	**Beschreibung**
Editor	graphisch mit – einstellbaren Farben und Schriften – beliebig skalierbarem Zoom
SQL-Scriptgenerator	vorhanden
ODBC-Schnittstelle	vorhanden
Konsistenzcheck	vorhanden
Versionskontrolle	nicht automatisch sondern manuell über Modelleigenschaftsfenster
Modellverwaltung	Untermodelle definierbar und Modellzusammenführung
Datenimport	nur ERwin
Datenexport	nur als Grafik
Reportgenerator	vorhanden (beliebig einstellbar)
Reverse Engineering	vorhanden
andere Notationen	nicht vorhanden

Tabelle 5.1: Funktionalitäten des SQL-Modeler

Aufgrund der Eigenschaften des SQL-Modeler ist er hauptsächlich für die Unterstützung der Datenbankimplementierung von Sybase-Datenbanken gedacht, aber prinzipiell auch für andere relationale Datenbanksysteme geeignet. Eine objektorientierte Datenbankentwicklung ist nicht möglich. Die beim Reverse Engineering unterstützten Datenbankmodelle sind Datenbanksysteme aus dem PC-Bereich, wie z.B. Paradox, Foxpro, Access, SQL-Server, etc.. Da bei der folgenden Datenbankimplementation der SQL-Modeler benutzt wurde, soll dies als kleine Übersicht zur Funktionalität des SQL-Modeler ausreichen.

5.2 Datenbankimplementation

Basierend auf dem Relationenmodell müssen jetzt die datenbank- spezifischen Eigenschaften berücksichtigt werden. Ziel dabei ist die Umsetzung des logischen Schemata in ein konkretes konzeptuelles Schema für die Datenbank Sybase Adaptive Server Anywhere Version 6.0.3.2747 und folgend Sybase genannt in der SQL-DDL und SQL-DML[3]. Zunächst sollen die Wertebereiche der Attribute definiert werden. Danach erfolgt die Beschreibung des konzeptuellen Schemas, konkret einige Änderungen des logischen Entwurfs sowie die Beschreibung der Festlegung der Integritätsbedingungen.

5.2.1 Attributdefinition der DB-TGMV

Die Definition der Attributwertebereiche legt die Typintegrität der Spalten fest. Ferner besteht die Möglichkeit der Angabe, ob Nullwerte erlaubt sind oder nicht, also die Definition von optionalen und nicht optionalen Attributen, sowie die Angabe von *default*-Werten. Die Optionalität der einzelnen

[3]SQL-DML – SQL-Data Manipulation Language

Attribute ist in der Abbildung B.1 auf Seite 64 ersichtlich. In der Anwendung war es nicht notwendig, *default*-Werte vorzugeben. Weiterhin lassen sich zu den vorgegebenen Wertebereichen neue Wertebereiche, sogenannte Domänen, basierend auf vorhandenen Wertebereichen, definieren. Die genaue Attributmodellierung in Form der SQL-DDL befindet sich im Anhang B.1 ab Seite 50 und auch in Abbildung B.1. Die folgenden Tabellen beschreiben die Eigenschaften der verwendeten Wertebereiche der Attribute und Domänen.

Wertebereich	**Beschreibung / Speicherung / Verwendung**
char(n)	String fester Länge n (max. 255 Zeichen)/ Speicherplatz entspricht n-Bytes/ Speicherung von atomaren Attributen
varchar(n)	String variabler Länge n (max. 32767 Zeichen)/ Speicherplatz entspricht Anzahl der gespeicherten Zeichen in Bytes und Offsetaddresse/ kurze stichwortartige Zusatzinformation (teils semistrukturiert)
long varchar	beliebig langer String (max. 2GB)/ siehe varchar/ längere textuelle Zusatzinformation (semistrukturiert)
text	BLOB (max. 2GB, in 2kB Pages))// Textdokumente (semistrukturiert)
image	BLOB (max. 2GB, in 2kB Pages)// Bilddokumente (semistrukturiert)
numeric(n,m)	n-stelliger numerischer Ausdruck mit m-Nachkommastellen/ 2-17 Bytes Zahl(Wertebereich $[10^{38}-1, 10^{-38}]$)/ manuelle und systemgenerierte Schlüssel
tinyint	1 Byte Integer (Wertebereich $[0, 255]$)// Klassifikationsstufe
date	Datumswert/ 4 Byte / genaue Zeiteinordnung (Datum) bei Chroniken durch Mittelwertbildung bei Zeitspannen, Jahresangaben, etc.

Tabelle 5.2: Attributwertebereiche

Die Attributwertebereiche wurden entsprechend einer Analyse der Testdaten und unter Beachtung der richtigen Datentypwahl (vgl. [Kir98]) entsprechend dem verwendeten Datenbanksystem Sybase festgelegt.

Domäne	**Wertebereich**	**Beschreibung**
MOID...	numeric(2,0) numeric(4,0) numeric(20,0) numeric(30,0)	manuell generierte Schlüssel entsprechender Länge für Klassifikationen und Unterscheidungsmerkmale
OID...	numeric(10,0) IDENTITY numeric(20,0) IDENTITY	systemgenerierte Schlüssel entsprechender Länge
ROID...	numeric(10,0) numeric(20,0)	Fremdschlüssel zu systemgenerierten Schlüsseln entsprechender Länge

Tabelle 5.3: Domänendefinition

Das Domänenkonzept wurde für die Definition der Schlüsselattribute genutzt, um anhand des Domänennamen die Schlüsselart und den Wertebereich zu erkennen. Dies hat technisch gesehen keinen besonderen Vor- bzw. Nachteil. Die manuellen und systemgenerierten Schlüsselbereiche sind ausreichend groß für die Anwendungsdomäne TGMV ausgelegt. Auf die manuelle Generierung der Schlüssel wird im nächsten Abschnitt eingegangen.

5.2.2 Beschreibung des konzeptuellen Schemas der DB-TGMV

Aufgrund der speziellen Daten und ihrer Beziehung untereinander ist die Konsistenzerhaltung der Daten besonders wichtig. Im vorigen Abschnitt wurde die Typintegrität als Überwachung der Wertebereiche von Attributen genannt und in diesem Zusammenhang auch die Kontrolle der Optionalität (`null`/`not null`) von Attributen. Als nächstes soll die gewählte Implementation (siehe Anhang B) von weiteren Integritäten (siehe auch [DD98]) beschrieben werden, bevor auf Besonderheiten der Implementation eingegangen wird.

Integritätswahl

Im Relationenmodell wurden die Schlüssel der Entity-Typen und der Beziehungstypen ausgewählt sowie die Fremdschlüssel gekennzeichnet. Für die meisten Entity-Typen wurde ein systemgenerierter Schlüssel implementiert. Das Schlüsselattribut ist atomar und numerisch und wurde aufgrund dieser Eigenschaften dem natürlichen Schlüssel vorgezogen. Hier ist die Schlüsselintegrität unter anderem bereits durch die Schlüsselvergabe gesichert. Andere Entity-Typen, wie die Klassifikationen und Unterscheidungsmerkmale, besitzen einen manuell vergebenen Schlüssel. Hier muß der Datenbanknutzer auf alle Fälle die Konsistenz seiner Klassifikationsschemata (Hierarchiebäume) und Unterscheidungsmerkmale (lineare Nummerierung) selbst überprüfen, denn nur die Schlüsselintegrität wird durch die Datenbank gewährleistet. Da der Änderungsaufwand und die Anzahl der Klassifikationen sehr gering sind, ist dies zumutbar. Diese Schlüssel haben auch einen numerischen Wertebereich.

Die referenzielle Integrität wird durch die Fremdschlüssel gewährleistet. Die Fremdschlüssel wurden bereits im Relationenmodell gekennzeichnet und werden jeweils um die referenziellen Aktionen `on update restrict` und `on delete restrict` erweitert. Dies bedeutet, daß jegliche Update- und Deleteoperationen auf referenzierten Schlüsseln zurückgewiesen werden.

Die meisten Entity-Typen mit dem systemgenerierten Schlüssel erhalten einen alternativen Schlüssel, der durch `UNIQUE` generiert wird. Hiermit soll die Überwachung des natürlichen Schlüssels gewährleistet werden, so daß es z.B. nicht möglich ist, jeweils zwei Personen, Städte, Technische Objekte, etc. mit dem gleichen Namen in der Datenbank zu speichern. Aus besonderem Grund wurde der alternative Schlüssel bei dem Entity-Typ *ADRESSEN* auf alle Attribute ausgedehnt, was später beschrieben wird.

Zusätzlich zu diesen Integritäten ist es innerhalb einiger Relationen weiterhin notwendig, einige attributspezifische Bedingungen (Zusicherungen) zu definieren. Diese werden mit der `check`-Klausel angegeben. Wenn diese Bedingungen unmittelbar mit der Spaltendefinition (s. B.1) definiert werden, möglich bei einzelner Attributbedingung und nicht bei lokaler Tupelbedingung, so sind sie später in Sybase jederzeit änderbar. Nachteil ist hierbei, daß bei einer Änderung der Zusicherung keine Integritätsüberprüfung der vorhandenen Attributwerte erfolgt und die Änderung gegebenenfalls verworfen wird. Dies wird problematisch, wenn zum Beispiel bei der DB-TGMV der Wert der Hierarchiestufen herabgesetzt werden soll, und vorhandene Werte größer als der neue Wert der Prüfbedingung sind. Dieser Fall sollte aber prinzipiell nicht vorkommen.

Wie eben erwähnt, ist die Attributzusicherung bei den Klassifikationsschemata bei dem Attribut *...STUFE* realisiert. Das Attribut *...STUFE* gibt die Hierarchiebaumtiefe des jeweiligen Knoten bzw. Blattes an, die u.a. aufgrund der Schlüssellänge begrenzt ist. Hierbei ist festgelegt, daß die Wurzel der Stufe 1 entspricht.

Eine weitere Bedingungsprüfung mußte in der Relation *TO_IN_TO_ENTHALTEN* imple-

mentiert werden. Durch die referenzielle Integrität wird hier nur ausgedrückt, daß jedes technische Objekt A höchstens einmal in einem beliebigen technischen Objekt B enthalten sein soll. Zusätzlich muß hier noch angegeben werden, daß A von B unterschiedlich sein muß und weiterhin, daß es zu keiner kreuzweisen Paarung (Schleife) kommt. Dies wurde mit einer lokalen Bedingungsprüfung auf Tupeln realisiert.

Weitere Integritätsbedingungen sind bisher nicht implementiert worden. Eine Bedingung muß im Zusammenhang mit den Entity-Typen, die einem Klassifikationsschema zugeordnet werden sollen, unbedingt noch erwähnt werden. In Abschnitt 4.2.1 auf Seite 31 wurde unter dem Punkt "Beziehungtransformation" bei der Modellierung des Beziehungstypen $R\big(E_1[0,N], E_2[1,M]\big)$ beschrieben, daß die zwingende Bedingung von E_1 auf E_2 nicht durch Schlüsselbedingung realisierbar ist. Eine Möglichkeit wäre hier z.B. die verzögerte Bedingungsprüfung mit der `check`-Klausel, die zusätzlich nach jeder Insertoperation überprüft, ob das Entity aus E_1 klassifiziert wurde. Das bedingt eine Kapselung mehrerer Insertoperationen als Transaktion. Da die Daten sowieso über eine Anwendung (vgl. Abschnitt 5.3) in die Datenbank eingefügt werden sollen, wird die verzögerte Bedingungsprüfung hier nicht realisiert. In der Anwendung muß für jedes Entity - hier aus *STAEDTE, ADRESSEN, PERSONEN* und *TOBJEKTE* - welches einer oder mehreren Klassifikationen - hier zu *REGIONEN, P_HIERARCHIE* und *TO_HIERARCHIE* - zugeordnet wird, unmittelbar bei der Insertoperation die Klassifikation zugeordnet werden. Diese zusammenhängenden Operationen sind als Transaktion auszuführen.

Die zwingende Kreiszuordnung jeder Adresse wird bereits durch die Fremdschlüsselmodellierung bei den Adressen gewährleistet. Das gleiche trifft für die Zuordnung von Unterscheidungsmerkmalen - hier aus *MUSEUMS_ART, LITERATUR_ART, TO_IN_TO_ART, BILDTYP* und *BILDFORMAT* - zu den jeweiligem Entity - hier auf *MUSEEN, LITERATUR, TO_IN_TO_ENTHALTEN* und *BILDER* - zu.

Wie bereits am Beginn dieses Abschnittes erwähnt, wurde bei der Modellierung der DB-TGMV vorrangig Wert auf die Konsistenzerhaltung der Daten gelegt. Dies steht natürlich im Gegensatz zu Performance, da jede Bedingungsprüfung Zeit kostet. Darauf wird im Abschnitt 5.3 unter dem Punkt Datenverwaltung näher eingegangen.

Besonderheiten der Implementation

Einige Erklärungen zum Verständnis der Implementation wurden bisher aufgeschoben und werden nun hier vorgenommen.

Die Klassifikationsschemata *KREIS, REGIONEN, P_HIERARCHIE* und *TO_HIERARCHIE* besitzen alle einen manuellen Schlüssel, der durch die Hierarchiebaumnummerierung vorgegeben ist. Die Nummerierung des Baumes erfolgt "preorder". Dabei werden stufenweise zuerst von links absteigend nach rechts jeweils die nachfolgenden Baumelemente durchnummeriert. Die Nummer von jedem Baumelement (Wurzel, Nummer, Blatt) ist zweistellig und demzufolge kann jedes Baumelement maximal 99 Nachfolger Elemente haben. Bei der Bildung des manuellen Schlüssels werden nun alle zweistelligen Nummern, beginnend von der Wurzel des Baumes, auf dem Pfad bis zum entsprechenden Knoten bzw. Blatt aneinandergereiht. Danach ist der Rest des Schlüssels entsprechend seiner Länge mit Nullen aufzufüllen. Diese Schlüssel sind eindeutig und der Baum ist nachträglich beliebig erweiterbar. Eine Umstrukturierung des Baumes ist sehr aufwendig, sollte aber im Normalfall selten vorkommen. Der größte Vorteil dieser Nummerierung ist die Möglichkeit von Bereichsanfragen über den Schlüssel bzw. die Suche nach allen Nachfolgerelementen ab einer bestimmten Baumposition. Dabei wird das Attribut *...STUFE* genutzt, um den relevanten Teil des Schlüssels zu erhalten. Alle nachfolgenden Baumelemente besitzen den gleichen Anfangsstring innerhalb des Schlüssels und können mittels der `like`-Bedingung selektiert werden. Eine Ausnahme hierbei ist, wie bereits schon mehrfach zuvor erwähnt, die Kreishierarchie. Die Kreise werden nur einfach durchnummeriert, was auch am zweistelligen Schlüssel zu erkennen ist. Auf den Klassifikationsschemata, wiederum bis auf Kreis, sind keine Alternativschlüssel vergeben, da es aufgrund der Klassifikationseigenschaften (vgl. Abschnitt 3.2.1 auf Seite 15) zu Mehrfachnennungen von Eigenschaften innerhalb des Baumes kommen kann.

Der Alternativschlüssel beim Entity-Typ *ADRESSEN* erstreckt sich über alle Attribute (*STRASSE, HNR, PLZ, ORT*). In der Datenbank existieren vier Entity-Typen, die Adressinformationen benötigen. Dabei kann es z.B. vorkommen, daß ein Verein, ein Museum und diverse enthaltene technische Objekte die selbe Adresse haben. Das gleiche gilt auch für technische Objekte, die in anderen technischen Objekten enthalten sind. Im Vergleich zur Anzahl von technischen Objekten existieren verhältnismäßig wenig Vereine mit Kontaktpersonen und Museen. Um Redundanzen in der Datenbank zu vermeiden, sind die Adressinformationen in einem Entity-Typ zusammengefaßt worden. Problematisch hierbei ist, daß manche technische Objekte keine vollständige Adresse besitzen, aber der Alternativschlüssel bedingt die Angabe einer vollständigen Adresse. Dies wird hier nicht durch den Nullwert der Datenbank ausgedrückt, sondern einfach durch einen leeren String. Die Adressangaben sollen zwingende Angaben sein, was auch in der Implementation in der Abbildung B.1 auf Seite 64 zu sehen ist. Dabei sind die Angabe eines Ortes mit Postleitzahl sowie die Einordnung zu den Klassifikationsschemata Mindestbedingungen. Falls keine Adresse existiert, was natürlich die Ausnahme sein soll, so kann dies z.B. durch eine sogenannte "Dummy-Adresse" simuliert werden. Wie schon in Abschnitt 4.2.2 unter dem Punkt Schemaeigenschaften beschrieben, ist eine weitere Normalisierung der Adresstabelle aus Zugriffsgründen nicht vorgesehen. Diese Adressinformationen sind als Zusatzinformation zu dem eigentlichen Datenbestand zu sehen.

Mit den Literaturangaben verhält es sich ähnlich wie mit den Adressangaben. Es soll nicht das Ziel sein, eine Literaturdatenbank aufzubauen. Aus diesem Grund ist die Literaturtabelle auch nicht vollständig normalisiert. Auflagen von Büchern bleiben in der Implementation unberücksichtigt. Die Literaturangaben sind nur Zusatzangaben zu den Entity-Typen *STAEDTE, PERSONEN, TOBJEKTE* und *MUSEEN* und dienen nicht der Quellenangabe gespeicherter Dokumente. Diese sind als zwingende Attributwerte im jeweiligen Entity-Typ modelliert.

Der Grund, daß Bilder in einem separaten Entity-Typ modelliert werden, ist in Abschnitt 4.1.2 ab Seite 27 unter dem Punkt Entity- und Attributmodellierung beschrieben worden. Weiterhin wird auf das Thema nochmal in Abschnitt 6.2 ab Seite 45 eingegangen.

Die Implementation der Entity-Typen ... *CHRONIK* soll hier nochmals durch eine andere Wahl der Modellierung in Frage gestellt werden. In der jetzigen Implementation sind die Entity-Typen ... *CHRONIK* ähnlich wie die ... *TEXTE* als objektwertige Attribute modelliert, wobei sie sich in der Beziehung zum zugehörigen Entity-Typ eigentlich nur in der Wahl der Kardinalität unterscheiden, wenn man von anderen Eigenschaften absieht. Der Aufbau des Entity-Typ ... *CHRONIK* könnte auch die Modellierung als mehrwertiges und mengenwertiges Attribut nahe legen. Dabei würde der Entity-Typ ... *CHRONIK* direkt als Attribut mit dem Datentyp `bit` im zugehörigen Entity-Typ aufgenommen. Der Datentyp `bit` kann `NULL`, 0 oder 1 sein. Die Bedeutung müßte geeignet festgelegt werden, bzw. `NULL` ausgeschlossen werden. Im Falle einer vorhandenen Chronik würde das entsprechende Attribut auf 1 gesetzt. Die Chronikdaten würden dann den Schlüssel des zugehörigen Entities zugeordnet bekommen und dies wäre zusammen mit ihrer OID (reihenfolgeabhängige manuelle Eintragsnummer der Chronikdaten zum jeweiligen Entity) der Schlüssel der Chronikdaten. Diese Modellierungsvariante hat keine besonderen Vorteile, aber auch keine Nachteile. Aus diesem Grund wurde die bisherige Sichtweise als objektwertiges Attribut beibehalten, da diese Modellierung besser zum Ausdruck bringt, daß es sich bei Chroniken um ein Dokument zu einem zugehörigen Entity handelt.

Auf die einzelnen Attributbedeutungen wurde bereits mehrfach eingegangen, unter anderem das Attribut ... *ZEITEINORDNUNG* in Tabelle 5.2 auf Seite 37.

Eine Besonderheit stellt der Schlüssel ... *MOID* der Entity-Typen ... *DATEN* dar. Er kann manuell vergeben oder durch die Anwendung generiert werden. Wenn er manuell vergeben wird, so könnte man mit dem Schlüssel eine bestimmte Reihenfolge der Chronikdaten simulieren, was durch das Attribut ... *ZEITPUNKT* nicht möglich ist, da es teilweise Bereichsangaben, teilweise nur Jahresangaben, teilweise ungenaue Zeitangaben etc. bezeichnet. Die Anwendung muß zusätzlich beim nachträglichen Einfügen bzw. Löschen von Chronikdaten eine Reorganisation der jeweiligen Chronik durchführen.

Durch das Attribut ... *ZEITEINORDNUNG* wird ein konkretes Datum manuell vergeben, welches dem Mittelwert der Zeitangabe aus dem Attribut ... *ZEITPUNKT* entspricht, so z.B. bei Zeitbereichsangaben, unvollständigen Datumswerten etc.. Durch dieses Attribut kann ebenfalls eine chronologischen Ordnung der Chronikdaten ohne die Verwendung des Schlüssels der Chronikdaten hergestellt werden. Die Anwendung sollte in diesem Fall den Schlüssel automatisch generieren. Eine Reorganisation der Schlüsselnummern ist nur beim Überschreiten des Wertebereiches des Schlüssels notwendig.

Die bisherigen Texte stammen überwiegend aus Zeitungsartikeln bzw. Buchausschnitten. Die Struktur dieser Texte besteht hauptsächlich aus Überschrift und Absätzen sowie evtl. bei den Artikeln der Autorenangabe. Aus diesem Grund werden die Texte bisher "vollkommen" unstrukturiert als ASCII-Texte im Attribut ... *TEXTE* der Entity-Typen ... *TEXTE* gespeichert. "Vollkommen" unstrukturiert sind sie natürlich nicht, denn einige Informationen werden gesondert gespeichert. So werden die Überschrift im Attribut ... *BEZEICHNUNG* und die Quellenangaben im Attribut ... *QUELLE* gespeichert. Der unveränderte Artikel wird dann im entsprechenden Attribut gespeichert. Da aus Artikeln nicht sonderlich viele Strukturinformationen zu gewinnen sind, außer vielleicht noch die Absätze, ist ein Zergliedern der Artikel nicht sinnvoll. Die vorhandenen Strukturinformationen, wie z.B. "Überschrift" - Erste Zeile, "Absatz" - Trennung durch Leerzeile, "Autor", falls vorhanden - letzte Zeile, wenn Absatz nicht länger als eine Zeile, könnte man auch aus den ASCII-Texten mit einem Parser selektieren, so daß auch das gegen ein Zergliedern der Artikel spricht. Sinnvoll ist die Integration von Strukturinformationen erst bei Texten, die Strukturbestandteile enthalten, wie z.B. Texte die in Kapitel, ähnlich der Studienarbeit, unterteilt sind, Texte die Tabellen enthalten u.s.w.. Vorstellbar sind auch Texte die multimediale Objekte enthalten, wie z.B. Bilder etc.. Dabei ist das Zergliedern von Texten, die in Kapitel strukturiert sind noch mit dem geringsten Aufwand realisierbar, z.B. durch Realisierung eines Unterscheidungsmerkmales für die Art des Textes. Bei einem Text mit Vorwort, Kapiteln etc. könnte dies rekursiv realisiert werden, ähnlich der Enthaltenseinsbeziehung bei den technischen Objekten. Wenn Texte nun aber andere strukturierte Informationen enthalten sollen, wie z.B. Tabellen, Aufzählungen, Bilder, deren Struktur oder Position im Text wichtig ist, so kann man diese nicht als reinen ASCII-Text speichern. An dieser Stelle wurde die Studienarbeit abgegrenzt, denn das Ziel bestand hier in der Implementation einer Datenbank für die vorhandenen Daten. Diese Aspekte werden aber nochmal im folgenden Abschnitt und im nächsten Kapitel weiter diskutiert. In diesem Zusammenhang erfolgt auch die Erklärung der Bedeutung der Attribute ... *SCHLAGWOERTER*.

Der Rest der Implementation bedarf an dieser Stelle erst einmal keiner weiteren Erklärung, da die meisten Zusammenhänge bereits in vorangegangen Kapiteln erklärt wurden. Im folgenden Abschnitt soll nun der Einsatz der Datenbank DB-TGMV beschrieben werden.

5.3 Anwendungsszenario

In Kapitel 2 wurden die Konzepte von IR-Systemen und Datenbanken vor- und gegenübergestellt. Dabei wurden prinzipielle Ähnlichkeiten herausgestellt, die Datenbanken und IR-Systeme zu Informationssystemen machen. In Abschnitt 2.3 wurden anschließend die jeweiligen Konzepte der Informationssysteme gegenübergestellt. Diese Gegenüberstellung soll jetzt wieder aufgegriffen werden, um das Anwendungsszenario der Datenbank TGMV zu beschreiben.

Wissensrepräsentation
Dieser Aspekt betrachtet die unterschiedliche Wissensrepräsentation, nämlich die semantische Wissenstransformation in IR-Systemen und die logische (syntaktische) Wissensrepräsentation in Datenbanken.

Abgesehen vom Datenbankschema, welches die Anwendungsdomäne semantisch repräsentiert, sind in der Datenbank die "strukturierten" Daten zu den Entities als logische Wissensrepräsentationen zu sehen. Die wichtigsten Kategorien sind hier der jeweilige Entity-Name und die Adressen zu den

touristisch- und technisch-relevanten Daten. Die Analyse und Strukturierung der Daten erfolgt manuell entsprechend dem Datenbankschema in Abbildung B.1 auf Seite 64.

Zusätzlich dazu wurden semantische Wissensrepräsentationen integriert. Die semantische Zuordnung von Entities wird manuell bei der Datenerfassung vorgenommen und setzt Erfahrungen der Anwendungsdomäne TGMV voraus.

Die Zuordnung von Entities zu den Klassifikationsschemata ist die wichtigste semantische Repräsentation. Sie erfordert einerseits eine gründliche Analyse der Entities und andererseits die Kenntnis der Klassifikationsschemata.

Eine weitere semantische Repräsentation wurde bei den Entity-Typen ... *TEXTE* über das Attribut ... *SCHLAGWOERTER* integriert. Hier ist der Hintergrund der, daß eine Klassifizierung der Entities zwar im allgemeinen reicht, aber keine genaue Inhaltserschließung vom Text darstellt und in der Anwendungsdomäne TGMV viel mehr Begriffe existieren, die nicht in den Klassifikationsschemata auftreten. Diese sollen genutzt werden, um die Semantik der Texte hervorzuheben. Der Aufwand zur Realisierung anderer Möglichkeiten ist sehr hoch, da die Anwendungsdomäne in Bezug auf die Wissensgebiete ziemlich breit gefächert ist.

Die Analyse und richtige Zuordnung der Daten ist elementar wichtig für den Retrievalprozeß. Unter dem Punkt Retrievalprozeß wird nachher beschrieben, wie die Wissensrepräsentationen zur Anfrage genutzt werden können. Es ist nämlich davon auszugehen, daß der Nutzer den Datenbankinhalt und die Anfragesprache nicht kennt, was durch IR-Systeme unterstützt wird.

Datenverwaltung

Die Vorteile der Nutzung einer Datenbank zur Speicherung von Daten sind ausgiebig im Kapitel 2 diskutiert sowie in Abschnitt 2.2.1 ab Seite 9 im einzelnen vorgestellt worden. Im Abschnitt 4.1 auf Seite 25 wurde darauf hingewiesen, daß beim Entwurf der Datenbank kein spezieller Sichtenentwurf durchgeführt wird, sondern von einer Gesamtsicht auf den Datenbestand ausgegangen wird. An dieser Tatsache hat sich nichts geändert. Weiterhin wurde erwähnt, daß kein Verteilungsentwurf vorgesehen ist, was prinzipiell noch stimmt, aber anwendungsbedingt näher erläutert werden soll.

Im vorherigen Abschnitt 5.2.2 am Ende unter dem Punkt Integritätswahl wurde bemerkt, daß es einerseits für die Konsistenz der Daten sehr wichtig ist, möglichst viele Integritäten durch die Datenbank und auch Anwendung bei der Dateneingabe zu prüfen. Dies steht anderseits im Gegensatz zu einer guten Datenbankanwendungsperformance. Da auf die Datenbank TGMV von der Anfrageseite, ähnlich eines Onlinelexikons, nur lesend zugegriffen wird, besteht hier die Möglichkeit, den Originaldatenbestand entsprechend dem Anwendungswunsch zu replizieren. Die Kopie der Datenbank kann so optimal dem Anwendungsverhalten angepaßt werden, z.B. durch Einrichtung von Zugriffspfaden, Speicherstrukturen etc., wodurch auch eine gute Anwendungsperformance erreicht wird. Wann und wie ein Datenabgleich durchgeführt werden muß, läßt sich flexibel handhaben und ist mit Sybase SQL Remote realisierbar.

Retrievalfunktionalität

Im Abschnitt 2.3 ab Seite 11 wurde unter dem gleichnamigen Punkt bereits auf die wesentlichen Unterschiede zwischen Datenbankenretrieval und Informationretrieval eingegangen. Hier soll nun beschrieben werden, welche Möglichkeiten aus Anwendungssicht existieren, um mit den vorhandenen Datenbank-Retrievalfunktionalitäten den Informationswunsch der Anwender zu erfüllen.

Prinzipiell soll ein IR in der DB-TGMV über die Klassifikationsschemata laufen, um darüber geeignete Entities zu finden und über diese z.B. interessierende Informationen zu selektieren. Die sich daraus ergebenden Bereichsanfragen über die Hierarchiestufen sind im vorherigen Kapitel unter Besonderheiten der Implementation beschrieben.

Da der Aufbau der Klassifikationen übersichtlich ist, bisher nicht mehr als 10 Nachfolger pro Knoten und die gleiche Baumtiefe, ist ein navigierender Zugriff aus einer Anwendung über die Klassi-

fikationen auf die Entities denkbar. Unterstützt wird dies auch durch die selbsterklärende Bedeutung der einzelnen Klassifikationen. Das Ergebnis dieser Bereichsanfrage ist eine Menge von Entities, die durch Auswahl selektiert werden können. Nach der Selektion eines Entity kann über einen natürlichen Verbund das Entity mit allen seinen Beziehungen zu anderen Entities und objektwertigen Attributen dargestellt werden. Die Navigation zu einem anderen Entity, z.B. von einem technischen Objekt zu einer Personen, ist dabei wieder eine konkrete Selektion eines Entity, für welches dann wieder über den natürlichen Verbund die vorhandenen Beziehungen aufgelistet werden könnten. Solche Navigationen werden auch von PC-Lexika unterstützt, bzw. sind den Linkstrukturen im Internet ähnlich. So könnte eine touristische Anfrage z.B. durch eine Navigation über ein sogenanntes `IMAGE MAP`, wie in HTML verwendet, entweder über die Kreise oder Regionen gezielt zu Informationen führen. Die beispielhaft genannten Anfragen entsprechen einer Suche auf den strukturierten Informationen in der Datenbank und sind durch als Bereichsanfragen mit SQL realisierbar.

Eine weitere Suche auf den strukturierten Informationen ist über allgemein bekannte Entity-Attribute möglich, wie die Personennamen und Stadtnamen, da hier davon auszugehen ist, daß diese in den meisten Fällen richtig geschrieben werden und somit die Anfrage auch zum Erfolg führt. Eine andere Variante des navigierenden Zugriffs könnte auch über die Postleitzahlen bzw. Postleitzahlenbereiche laufen. Die Postleitzahl und der Stadtname waren bei der Datenbankimplementation eine Mindestbedingung bei den Adressenangaben zu den Entity-Typen.

Da in der Datenbank auch semistrukturierte Informationen - hier Textdokumente - enthalten sind, ist es natürlich wünschenswert auch nach diesen Informationen gezielt zu suchen. Diese Suche kann z.B. über eine Stichwortsuche mit Hilfe der `like`-Bedingung simuliert werden. Diese Suche sollte auf dem unter dem Punkt Wissensrepräsentationen genannten Attribut ... *SCHLAGWOERTER* durchgeführt werden, da dieses wie beschrieben den Text genauer klassifiziert. Diese Suche ist einerseits stark abhängig von der genauen Klassifizierung des Textes durch die Schlagwortwahl und andererseits von der Kenntnis der Begriffswelt der Anwendungsdomäne TGMV, also eher für die technisch interessierten Benutzer (vgl. Abschnitt 3.3 ab Seite 19 und Abbildung 3.2) geeignet. Außerdem wird hier auch eine gewisse Grundkenntnis der Anfragesprache vorausgesetzt, was als Nachteil des Datenbankretrieval gesehen wird. Sind diese Kenntnisse vorhanden, so ist es mit Hilfe der `like`-Bedingung möglich, reguläre Ausdrücke zu formulieren und auf den Schlagwörtern bzw. in den Texten zu suchen.

Die Bilder in der Datenbank sind den Entity-Typen zugeordnet. Auf ihnen soll kein Retrieval stattfinden. Die strukturierten Daten zu den Bildern sind notwendig, da in der Datenbank keine Imageretrievalfunktionen vorhanden sind. Das heißt, die Bilder lassen sich nur über ihre strukturierten Informationen unterscheiden und können, wie z.B. Bildtyp und Bildgröße, so auch als Eigenschaften zur Auswahl bzw. Darstellung in der Anwendung und beim Replizieren der Daten genutzt werden.

Fazit der Retrievalmöglichkeiten ist, daß die Informationen in der Datenbank zwar mit Hilfe der Anfragesprache SQL gesucht werden, aber das man dies über eine Anwendung basierend auf den Wissensrepräsentationen durchführen kann. Dies erfordert keine Kenntnis der Anfragesprache, nutzt aber deren Mächtigkeit. Bessere und vielleicht auch einfachere Retrievalmöglichkeiten auf den Texten setzen einen hohen Arbeitsaufwand zur Erstellung einer Wissensrepräsentation voraus, wie z.B. einen Thesaurus oder eine Ontologie und diese wiederum sind auch nur nutzbar, sofern im Datenbanksystem entsprechende Volltext-Retrieval-Funktionalitäten (vgl. [Por99]) enthalten sind.

Kapitel 6

Schlußbetrachtung

Abschließend soll hier die Studienarbeit zusammengefaßt werden. Beim Datenbankentwurf und der -implementation wurden einige Themen abgegrenzt. Diese werden im Abschnitt Probleme noch einmal aufgegriffen und erläutert, bevor ein Ausblick auf weiterführende Arbeiten gegeben wird.

6.1 Zusammenfassung

Die Implementation einer Datenbank aus vorhandenen semistrukturierten Daten machte es in dieser Studienarbeit notwendig, sich am Beginn mit Konzepten zur Textverarbeitung auseinanderzusetzen, bevor mit der eigentlichen Analyse der Daten begonnen wurde.

Basierend auf den Kenntnissen des IR wurden die vorhandenen Daten umfangreich analysiert, bevor ein strukturierter Entwurf durchgeführt werden konnte. Bei diesem Entwurf wurden alle vorhandenen Informationen der Daten, einer äußeren und inneren Dateistruktur, in den Entwurf einbezogen, um die Daten entsprechend der Anwendungsdomäne semantisch äquivalent im EER-Modell abzubilden.

Die Struktur der Daten und die Vorgabe des relationalen Datenbanksystems Sybase machten es notwendig, strukturierbare Daten von den semistrukturierten Daten zu trennen, ohne dadurch Redundanzen zu erzeugen. Dabei werden viele in den Ausgangsdaten vorhandenen Redundanzen vermieden. Die strukturierbaren Informationen werden zu Entity-Typen zusammengefaßt. Semistrukturierte Daten - hier Texte und Bilder - werden um strukturierte bzw. beschreibende Informationen ergänzt, um auch als Entities gespeichert werden zu können. In den Ausgangsdaten direkt enthaltene semantische Informationen, wie die Dateiverzeichnishierarchie, aber auch indirekt enthaltene, wie die territorialpolitische und regionale Gliederung sowie Klassifizierungen von Personen, werden als Klassifikationsschemata im Datenbanksystem integriert. Die implementierte Datenbank DB-TGMV bildet alle erkannten Informationen und ihre Beziehungen ab.

Über diese Klassifikationen kann ein navigierender Zugriff aus einer noch zu entwickelnden Anwendung erfolgen. Dies ist notwendig, da nicht unbedingt davon auszugehen ist, daß die künftigen Nutzer den Inhalt der Datenbank und die Struktur kennen, um gezielt Informationen zu suchen. Die Klassifikation der Adressen dient dabei eher einer touristisch orientierten Navigation und die der Personen und technischen Objekte sind technisch orientiert, was aber hier nicht einschränkend auf die Sicht der Daten gemeint ist, sondern eher den Informationswunsch der zukünftigen Nutzer zum Ausdruck bringen soll.

Die vorher erwähnte Trennung in strukturierbare und semistrukturierte Daten ist aus Datenbanksicht notwendig, was in Abschnitt 3.1 auch erläutert wurde. Eine Suche auf den semistrukturierten Datenbankinhalten ist prinzipiell über Pattern-Matching möglich, setzt aber gewisse Grundkenntnisse der Anwendungsdomäne und der Anfragesprache voraus. Um das besser zu unterstützen, wurden die Entity-Typen ... *TEXTE* um das Attribut für Schlagwörter erweitert.

Die Klassifikation als Wissensrepräsentation wurde hier gewählt, da sie in den Ausgangsdaten teilweise implizit vorhanden war, einfach im relationalen Datenbanksystem zu integrieren ist und

ein Retrieval für Nutzer, die unvertraut mit der Anwendungsdomäne sind, einfacher zu handhaben ist. Andere Wissensrepräsentationen erfordern zusätzlich einen hohen Implementationsaufwand, aber andererseits werden Volltext-Retrieval-Funktionalitäten durch das verwendete Datenbanksystem noch nicht unterstützt, bzw. standen zur Implementation nicht zu Verfügung.

6.2 Probleme

Bei dem konzeptuellen Entwurf und der Datenbankimplementation sind Abgrenzungen vorgenommen worden, die hier noch einmal zusammenfassend dargestellt werden sollen.

Im konzeptuellen Entwurf sind einige `string`-Attribute semistrukturiert geblieben, da sich keine geeigneten Strukturierungen angeben lassen. Dies ist bedingt durch die Eigenschaften semistrukturierter Daten (vgl. Abschnitt 3.1 auf Seite 14). Bei der Dateneingabe dieser Attribute, wie z.B. Öffnungszeiten, Anfahrtsbeschreibungen, Touristeninformationen, Medienhinweise, usw., sollte aber eine möglichst gleiche Semistruktur wegen einer späteren Ausgabe erzwungen werden. Diese Attributwerte lassen sich nicht präzise anfragen.

Bilder können in einem `image`-Datentyp gespeichert werden, aber es stehen keine speziellen Retrievalfunktionalitäten im Datenbanksystem zu Verfügung. Dies macht es natürlich notwendig, Bilder um strukturierte Informationen zu erweitern, die teilweise schon im speziellen Bildtyp enthalten sind. Bisher wurden nur strukturierte Informationen angelegt, die aus einer eventuellen Anwendungssicht zur Auswahl benötigt werden. Für eine Bilddatenbank sind diese Attribute natürlich nicht ausreichend, aber das war auch nicht das Ziel der Bildspeicherung. Nachteilig ist hier auch, daß anwendungsabhängig eventuell verschiedene Bildtypen und -formate gespeichert werden, was zu einer starken Redundanz führt.

Eine strukturierte Speicherung der Texte ist bisher nicht vorgesehen, kann aber aus Anwendungssicht zur Darstellung der Texte benötigt werden. Die Problematik wurde im Abschnitt 2.1.3 ab Seite 7 und im Abschnitt 5.2.2 unter dem Punkt Besonderheiten der Implementation bereits ausführlich erklärt.

Da zum jetzigen Zeitpunkt noch nicht abzuschätzen ist, wieviel Objekte jemals in der Datenbank gespeichert werden, soll hier nochmals auf die Klassifikationsschemata eingegangen werden. Die Struktur der Klassifikationsschemata sollte bewußt flach und nicht zu breit gehalten werden, da sonst ein Navigieren zu langwierig ist. Die Zuordnung von Entities im Klassifikationsschema ist ungleich verteilt. Einige spezielle Bereiche prägten die Technikgeschichte von Mecklenburg-Vorpommern besonders. Aus diesem Grund ist möglichst eine Kombination mehrerer Klassifikationen zu den Entity-Typen notwendig, da sonst das Suchergebnis zu groß wird. Das selbe kann auch zwischen Entity-Typen und ihren zugehörigen Texten bzw. Bildern auftreten.

6.3 Ausblick

Zu der in dieser Studienarbeit vorgestellten Datenbankimplementation muß noch eine Anwendungsoberfläche implementiert werden, zu der schon einige Anregungen in den vorherigen Abschnitten und Kapiteln gegeben worden sind.

Entsprechend der späteren Verwendung der Daten ist es bestimmt noch einmal überlegenswert, die unstrukturierten Texte logisch zu strukturieren, z.B. mit XML als Dokumentenaustauschformat, damit die Informationen in der Anwendung besser präsentiert werden können. In diesem Zusammenhang ist auch eine direkte Zuordnung der Bilder in die Texte überlegenswert, was aber dann eine Erweiterung der Datenbankimplentation bedingt.

Anhang A

Attributmodellierung für EER-Modell

```
entity type Staedte                          {vorrangig für Stadtchronik und -geschichte}
   attributes
      SPLZ digit(5),
      SName string,
      SMedienhinweis string,
      SChronik list( record( SCZeiteinordnung date,
                             SCZeitpunkt string,
                             SCEreignis string ) ),
      STexte set( record( STBezeichnung string,
                          STSchlagwoerter string,
                          STDokument textstring,
                          STQuelle string ) ),
   key
      SPLZ
end entity type;

entity type AHierarchie                      {Klassifikationsschema für touristische Anfragen
   attributes                                bei Entities mit Adressangaben (Kreise, Regionen)}
      AHOID digit,                           {selbstdefinierte OID}
      AHStufe digit,                         {Klassifikationsstufe}
      AHName string,
   key
      AHOID
end entity type;

entity type Personen                         {Personen der Technikgeschichte}
   attributes
      PVorname string,
      PNachname string,
      PGeburtsort string,
      PGeburtsdatum date,
      PWirkungsort string,
      Pgestorben-in string,
      Pgestorben date,
      PMedienhinweis string,
      PChronik list( record( PCZeiteinordnung date,
                             PCZeitpunkt string,
                             PCEreignis string ) ),
```

```
      PTexte set( record( PTBezeichnung string,
                          PTSchlagwoerter string,
                          PTDokument textstring,
                          PTQuelle string ) ),
   key
      PVorname, PNachname
end entity type;

entity type PHierarchie                    {Klassifikationsschema für Personen}
   attributes
      PHOID digit,                         {selbstdefinierte OID}
      PHStufe digit,                       {Klassifikationsstufe}
      PHName string,
   key
      PHOID
end entity type;

entity type TObjekte                       {technische Objekte}
   attributes
      TOOID digit,
      TOName string,
      TOAdresse record( TOStraße string,
                        TOHNR string,
                        TOPLZ digit(5),
                        TOOrt string ),
      TOAnfahrt string,                    {zu Fuß, mit Auto, öffentlicher Nahverkehr}
      TOTouristinfo textstring,            {touristische Informationen zum Umfeld}
      TONutzung string,
      TOMedienhinweis string,
      TOChronik list( record( TOCZeiteinordnung date,
                              TOCZeitpunkt string,
                              TOCEreignis string ) ),
      TO_Texte set( record( TO_TBezeichnung string,
                            TO_TSchlagwoerter string,
                            TO_TDokument textstring,
                            TO_TQuelle string ) ),
   key
      TOOID
end entity type;

relation type TOie                         {Enthaltenseinsbeziehung zw. technischen Objekten}
   attributes
      TO-A-OID digit,                      {Entity-A enthalten in
      TO-B-OID digit,                      Entity B}
      TOenthaltenArt string,               {A Teil von B, A Exemplar von B, ...}
   key
      TO-A-OID
end entity type;

entity type TOHierarchie                   {Klassifikationsschema für technische Objekte}
   attributes
      TOHOID digit,                        {selbstdefinierte OID}
      TOHStufe digit,                      {Klassifikationsstufe}
      TOHName string,
   key
      TOHOID
end entity type;
```

```
entity type Bilder                          {Bilder zu Staedte, Personen, TObjekte}
   attributes
      BOID digit,
      BName string,
      BKuerzel string,
      BBeschreibung string,
      BTyp string,
      BBreite string,
      BHoehe string,
      BBild imagestring,
      Quellennachweis string,
   key
      BOID
end entity type;

entity type Vereine                         {Vereine, Verbände mit touristischer und
   attributes                                technikgeschichtlicher Relevanz}
      VName string,
      VKuerzel string,
      VAdresse record( VStraße string,        {Vereinsadresse}
                       VHNR string,
                       VPLZ digit(5),
                       VOrt string,
                       VTNr set( string ) ),
      KPerson record( KVorname string,        {Adresse der Kontaktperson}
                      KName string,
                      KStraße string,
                      KHNR string,
                      KPLZ digit(5),
                      KOrt string,
                      KTNr string,
                      KEMAIL string ),
      Vereinsziel string,
      VEmailadresse string,
      VHomepage string,
      VTexte set( record( VTBezeichnung string,
                          VTSchlagwoerter string,
                          VTDokument textstring,
                          VTQuelle string ) ),
   key
      VName
end entity type;

entity type Museen                          {Museum, Ausstellungen, private Sammlungen, ...
   attributes                                aber auch Touristinformationen, ...}
      MName string,
      MArt string,                          {Sammlung, Ausstellung, Heimatmuseum, ...}
      MAdresse record( MPLZ digit(5),
                       MOrt string,
                       MStraße string,
                       MHNr string,
                       MTNr set(string) ),
      MEmail string,
      MHomepage string,
      MÖffnungszeiten string,               {hoher Bearbeitungsaufwand, unstrukturiert }
      MAnfahrt string,                      {zu Fuß, mit Auto, öffentlicher Nahverkehr}
      MVeranstaltungen textstring,
      MInformationsmaterial textstring,
```

```
      MTouristinfo textstring,
      MTexte set( record( MTBezeichnung string,
                          MTSchlagwoerter string,
                          MTDokument textstring,
                          MTQuelle string ) ),
   key
      MName
end entity type;

entity type Literatur                          {Literaturempfehlung, -verweis, ... }
   attributes
      LOID string,                             {systemgenerierte OID}
      ISBN string,                             {falls vorhanden, ansonsten andere Buchkennzeichnung}
      LArt string,                             {Buch, Zeitschrift, Tagebuch, Archivaufzeichnung, ... }
      Verlag string,
      Titel string,
      LDatum date,
      LORT string,
      LBezugsquelle string,                    {z.B. bei Literatur die nicht mehr verlegt wird}
      Autor set( record( AVorname string,
                         AName string) ),
   key
      LOID
end entity type;
```

Anhang B

Konzeptuelles Datenbankschema der DB-TGMV

B.1 SQL-DDL der DB-TGMV

```
/* ========================================================== */
/*   Database name:  "DB - TGMV"                              */
/*   DBMS name:      Sybase AS Anywhere 6                     */
/*   Content:        SQL-DDL                                  */
/*   Created on:     15.08.00  10:14                          */
/*   Modified:       30.10.00  10:08                          */
/*   Version:        0.23                                     */
/* ========================================================== */

/* ========================================================== */
/*   DOMAENEN                                                 */
/*   MID?? - manuelle vergebene OID's                        */
/*   OID??  - automatisch generierte OID's                    */
/*   ROID?? - auf automat. generierte OID's ref. Spalte       */
/* ========================================================== */
create datatype MOID20 numeric(20,0)          ;
create datatype MOID30 numeric(30,0)          ;
create datatype MOID2 numeric(2,0)            ;
create datatype MOID4 numeric(4,0)            ;
create datatype OID10 numeric(10,0) IDENTITY  ;
create datatype OID20 numeric(20,0) IDENTITY  ;
create datatype ROID10 numeric(10,0)          ;
create datatype ROID20 numeric(20,0)          ;

/* ========================================================== */
/*   Table: KREIS                                             */
/* ========================================================== */
create table KREIS
(
    K_OID                 MOID2                not null,
    K_NAME                char(40)             not null,
    primary key (K_OID)                                 ,
    unique (K_NAME)
);
comment on table KREIS is 'verwaltungspolitischer Kreis';
```

```
/* ========================================================== */
/*   Table: REGIONEN                                          */
/* ========================================================== */
create table REGIONEN
(
    R_OID                  MOID20               not null,
    R_STUFE                tinyint              not null
     check (R_STUFE between 1 and 10)                    ,
    R_NAME                 varchar(100)         not null,
    primary key (R_OID)
);
comment on table REGIONEN is 'touristisches Gebiet';

/* ========================================================== */
/*   Table: ADRESSEN                                          */
/*   Redundanzpruefung der Adresse notwendig!                 */
/* ========================================================== */
create table ADRESSEN
(
    ADR_OID                OID20                not null,
    STRASSE                char(30)                      ,
    HNR                    char(3)                       ,
    PLZ                    numeric(5,0)         not null,
    ORT                    char(25)             not null,
    K_OID                  MOID2                not null,
    primary key (ADR_OID)                               ,
    foreign key (K_OID) references KREIS
    on update restrict on delete restrict               ,
    unique (STRASSE, HNR, PLZ, ORT)
);
comment on table ADRESSEN is 'Adressen von Vereinen, Museen, technischen Objekten';

/* ========================================================== */
/*   Table: ADR_R_ZUORDNUNG                                   */
/* ========================================================== */
create table ADR_R_ZUORDNUNG
(
    ADR_OID                ROID20               not null,
    R_OID                  MOID20               not null,
    primary key (ADR_OID, R_OID)                        ,
    foreign key (ADR_OID) references ADRESSEN
    on update restrict on delete restrict               ,
    foreign key (R_OID) references REGIONEN
    on update restrict on delete restrict
);
comment on table ADR_R_ZUORDNUNG is 'Zuordnung einer Adresse zu einer touristischen Region';

/* ========================================================== */
/*   Table: STAEDTE                                           */
/* ========================================================== */
create table STAEDTE
(
    S_OID                  OID10                not null,
    S_NAME                 char(30)             not null,
    K_OID                  MOID2                not null,
    S_MEDIENHINWEIS        varchar(512)                  ,
    primary key (S_OID)                                 ,
    foreign key (K_OID) references KREIS
    on update restrict on delete restrict               ,
    unique (S_NAME)
);
comment on table STAEDTE is 'technikgeschichtliche Information zur Stadt';
```

```
/* ========================================================== */
/*   Table: S_R_ZUORDNUNG                                     */
/* ========================================================== */
create table S_R_ZUORDNUNG
(
    S_OID                  ROID10              not null,
    R_OID                  MOID20              not null,
    primary key (S_OID, R_OID)                         ,
    foreign key (S_OID) references STAEDTE
    on update restrict on delete restrict              ,
    foreign key (R_OID) references REGIONEN
    on update restrict on delete restrict
);
comment on table S_R_ZUORDNUNG is 'Zuordnung von Staedten zu touristischen Regionen';

/* ========================================================== */
/*   Table: S_TEXTE                                           */
/* ========================================================== */
create table S_TEXTE
(
    S_T_OID                OID20                 not null,
    S_OID                  ROID10                not null,
    S_T_BEZEICHNUNG        varchar(50)           not null,
    S_T_SCHLAGWOERTER      varchar(255)                  ,
    S_T_TEXTE              text                  not null,
    S_T_QUELLE             varchar(255)          not null,
    primary key (S_T_OID)                              ,
    foreign key (S_OID) references STAEDTE
    on update restrict on delete restrict              ,
    unique (S_T_BEZEICHNUNG, S_OID)
);
comment on table S_TEXTE is 'Textdokumente zur technikgeschichtlichen Stadtentwicklung';

/* ========================================================== */
/*   Table: S_CHRONIK                                         */
/* ========================================================== */
create table S_CHRONIK
(
    S_C_OID                OID10                 not null,
    S_OID                  ROID10                not null,
    primary key (S_OID)                                ,
    foreign key (S_OID) references STAEDTE
    on update restrict on delete restrict
);
comment on table S_CHRONIK is 'technikgeschichtliche Stadtchronik';

/* ========================================================== */
/*   Table: S_C_DATEN                                         */
/* ========================================================== */
create table S_C_DATEN
(
    S_C_D_MOID             MOID4                 not null,
    S_C_OID                ROID10                not null,
    S_C_D_ZEITEINORDNUNG   date                  not null,
    S_C_D_ZEITPUNKT        char(25)              not null,
    S_C_D_EREIGNIS         varchar(1024)         not null,
    primary key (S_C_D_MOID, S_C_OID)                  ,
    foreign key (S_C_OID) references S_CHRONIK(S_C_OID)
    on update restrict on delete restrict
);
comment on table S_C_DATEN is 'Daten zur technikgeschichtlichen Stadtchronik';
```

```
/* ========================================================== */
/*   Table: PERSONEN                                          */
/* ========================================================== */
create table PERSONEN
(
    P_OID                  OID10                not null,
    P_VORNAME              char(35)             not null,
    P_NAME                 char(35)             not null,
    P_GEBURTSORT           char(100)                    ,
    P_GEBURTSDATUM         char(10)                     ,
    P_WIRKUNGSORT          char(35)                     ,
    P_GESTORBEN_IN         char(100)                    ,
    P_GESTORBEN            char(10)                     ,
    P_MEDIENHINWEIS        varchar(512)                 ,
    primary key (P_OID)                                 ,
    unique(P_VORNAME, P_NAME)
);
comment on table PERSONEN is 'Personen mit technikgeschichtlicher Relevanz';

/* ========================================================== */
/*   Table: P_TEXTE                                           */
/* ========================================================== */
create table P_TEXTE
(
    P_T_OID                OID20                not null,
    P_OID                  ROID10               not null,
    P_T_BEZEICHNUNG        varchar(100)         not null,
    P_T_SCHLAGWOERTER      varchar(255)                 ,
    P_T_TEXTE              text                 not null,
    P_T_QUELLE             varchar(255)         not null,
    primary key (P_T_OID)                               ,
    foreign key (P_OID) references PERSONEN
    on update restrict on delete restrict               ,
    unique (P_OID, P_T_BEZEICHNUNG)
);
comment on table P_TEXTE is 'Texte und Anekdoten zu Personen mit technikgeschichtlicher Relevanz';

/* ========================================================== */
/*   Table: P_CHRONIK                                         */
/* ========================================================== */
create table P_CHRONIK
(
    P_C_OID                OID10                not null,
    P_OID                  ROID10               not null,
    primary key (P_OID)                                 ,
    foreign key (P_OID) references PERSONEN
    on update restrict on delete restrict
);
comment on table P_CHRONIK is 'Chronik von Personen mit technikgeschichtlicher Relevanz';

/* ========================================================== */
/*   Table: P_C_DATEN                                         */
/* ========================================================== */
create table P_C_DATEN
(
    P_C_D_MOID             MOID4                not null,
    P_C_OID                ROID10               not null,
    P_C_D_ZEITEINORDNUNG   date                 not null,
    P_C_ZEITPUNKT          char(25)             not null,
    P_C_EREIGNIS           varchar(1024)        not null,
    primary key (P_C_D_MOID, P_C_OID)                   ,
    foreign key (P_C_OID) references P_CHRONIK(P_C_OID)
    on update restrict on delete restrict
);
comment on table P_C_DATEN is 'Daten zur Chronik von Personen mit technikgeschichtlicher Relevanz';
```

```
/* ========================================================== */
/*   Table: P_HIERARCHIE                                      */
/* ========================================================== */
create table P_HIERARCHIE
(
    P_H_MOID               MOID20               not null,
    P_H_STUFE              tinyint              not null
     check (P_H_STUFE between 1 and 10)                 ,
    P_H_NAME               char(50)             not null,
    primary key (P_H_MOID)
);
comment on table P_HIERARCHIE is 'Klassifikation von Personen';

/* ========================================================== */
/*   Table: P_H_ZUORDNUNG                                     */
/* ========================================================== */
create table P_H_ZUORDNUNG
(
    P_OID                  ROID10               not null,
    P_H_MOID               MOID20               not null,
    primary key (P_OID, P_H_MOID)                       ,
    foreign key (P_OID) references PERSONEN
    on update restrict on delete restrict               ,
    foreign key (P_H_MOID) references P_HIERARCHIE
    on update restrict on delete restrict
);
comment on table P_H_ZUORDNUNG is 'Zuordnung von Person zur Klassifikation';

/* ========================================================== */
/*   Table: TOBJEKTE                                          */
/* ========================================================== */
create table TOBJEKTE
(
    TO_OID                 OID20                not null,
    ADR_OID                ROID20               not null,
    TO_NAME                varchar(255)         not null,
    TO_ANFAHRT             long varchar                 ,
    TO_TOURISTINFO         long varchar                 ,
    TO_NUTZUNG             varchar(50)                  ,
    TO_MEDIENHINWEIS       varchar(512)                 ,
    primary key (TO_OID)                                ,
    foreign key (ADR_OID) references ADRESSEN
    on update restrict on delete restrict               ,
    unique (TO_NAME)
);
comment on table TOBJEKTE is 'Technisches Objekt (Gebaeude,Fabrik,Fahrzeug, ...)';

/* ========================================================== */
/*   Table: TO_TEXTE                                          */
/* ========================================================== */
create table TO_TEXTE
(
    TO_T_OID               OID20                not null,
    TO_OID                 ROID20               not null,
    TO_T_BEZEICHNUNG       varchar(100)         not null,
    TO_T_SCHLAGWOERTER     varchar(255)                 ,
    TO_T_TEXTE             text                 not null,
    TO_T_QUELLE            varchar(255)         not null,
    primary key (TO_T_OID)                              ,
    foreign key (TO_OID) references TOBJEKTE
    on update restrict on delete restrict               ,
    unique (TO_OID, TO_T_BEZEICHNUNG)
);
comment on table TO_TEXTE is 'Texte und Anekdoten zu technischen Objekten';
```

```
/* ========================================================== */
/*   Table: TO_CHRONIK                                        */
/* ========================================================== */
create table TO_CHRONIK
(
    TO_C_OID               OID20                not null,
    TO_OID                 ROID20               not null,
    primary key (TO_OID)                                ,
    foreign key (TO_OID) references TOBJEKTE
    on update restrict on delete restrict
);
comment on table TO_CHRONIK is 'Chronik von technischen Objekten';

/* ========================================================== */
/*   Table: TO_C_DATEN                                        */
/* ========================================================== */
create table TO_C_DATEN
(
    TO_C_D_MOID            MOID4               not null,
    TO_C_OID               ROID20               not null,
    TO_C_D_ZEITEINORDNUNG  date                 not null,
    TO_C_D_ZEITPUNKT       char(25)             not null,
    TO_C_D_EREIGNIS        varchar(1024)         not null,
    primary key (TO_C_D_MOID, TO_C_OID)                ,
    foreign key (TO_C_OID) references TO_CHRONIK(TO_C_OID)
    on update restrict on delete restrict
);
comment on table TO_C_DATEN is 'Daten zur Chronik von technischen Objekten';

/* ========================================================== */
/*   Table: TO_IN_TO_ART                                      */
/*          In_ENTHALTEN_ART '1' - A ist in B enthalten       */
/*                               (A Teil von B)               */
/*                           '2' - B ist Modell,              */
/*                               A ist Exemplar von B         */
/*                           ...                              */
/* ========================================================== */
create table TO_IN_TO_ART
(
    TO_E_OID               MOID2               not null,
    TO_E_ART               varchar(50)         not null,
    primary key (TO_E_OID)                             ,
    unique (TO_E_ART)
);
comment on table TO_IN_TO_ART is 'Art der Enthaltenseinsbeziehung von technischen Objekten';

/* ========================================================== */
/*   Table: TO_IN_TO_ENTHALTEN                                */
/* ========================================================== */
create table TO_IN_TO_ENTHALTEN
(
    TO_A_OID               ROID20              not null,
    TO_B_OID               ROID20              not null,
    TO_E_OID               MOID2               not null,
    primary key (TO_A_OID)                             ,
    foreign key (TO_A_OID) references TOBJEKTE
    on update restrict on delete restrict              ,
    foreign key (TO_B_OID) references TOBJEKTE
    on update restrict on delete restrict              ,
    foreign key (TO_E_OID) references TO_IN_TO_ART
    on update restrict on delete restrict              ,
    check ( TO_A_OID <> TO_B_OID AND
            NOT(TO_A_OID IN (select to_b_oid from to_in_to_enthalten) AND
            TO_B_OID IN (select to_a_oid from to_in_to_enthalten)))
);
comment on table TO_IN_TO_ENTHALTEN is 'technisches Objekt kann Teil eines technischen Systems sein';
```

```
/* ========================================================== */
/*   Table: TO_HIERARCHIE                                     */
/* ========================================================== */
create table TO_HIERARCHIE
(
    TO_H_MOID              MOID30               not null,
    TO_H_STUFE             tinyint              not null
     check (TO_H_STUFE between 1 and 15)                ,
    TO_H_NAME              varchar(50)          not null,
    primary key (TO_H_MOID)
);
comment on table TO_HIERARCHIE is 'Klassifizierung der technischen Objekte
(manuelle Vergabe der OID)';

/* ========================================================== */
/*   Table: TO_H_ZUORDNUNG                                    */
/* ========================================================== */
create table TO_H_ZUORDNUNG
(
    TO_OID                 ROID20               not null,
    TO_H_MOID              MOID30               not null,
    primary key (TO_OID, TO_H_MOID)
);
comment on table TO_H_ZUORDNUNG is 'Zuordnung des technischen Objektes zu n Klassifizierungen';


/* ========================================================== */
/*   Table: P_TO_BEZIEHUNG                                    */
/* ========================================================== */
create table P_TO_BEZIEHUNG
(
    P_OID                  ROID10               not null,
    TO_OID                 ROID20               not null,
    primary key (P_OID, TO_OID)                        ,
    foreign key (P_OID) references PERSONEN
    on update restrict on delete restrict              ,
    foreign key (TO_OID) references TOBJEKTE
    on update restrict on delete restrict
);
comment on table P_TO_BEZIEHUNG is 'Person und technisches Objekt stehen in Beziehung
(z.B. erbaut von, entwickelt von, gearbeitet in, ...)';

/* ========================================================== */
/*   Table: BILDTYP                                           */
/* ========================================================== */
create table BILDTYP
(
    BT_MOID                MOID2             not null,
    BT_NAME                char(3)              not null,
    primary key (BT_MOID)
);
comment on table BILDTYP is 'Dateityp des Bildes';

/* ========================================================== */
/*   Table: BILDFORMAT                                        */
/* ========================================================== */
create table BILDFORMAT
(
    BF_MOID                MOID2                not null,
    BF_BREITE              numeric(4,0)         not null,
    BF_HOEHE               numeric(4,0)         not null,
    BF_BEMERKUNG           char(50)             not null,
    primary key (BF_MOID)
);
comment on table BILDFORMAT is 'Groessenangabe zum Bild (Breite x Hoehe in Pixel)';
```

```
/* ========================================================== */
/*   Table: BILDER                                            */
/* ========================================================== */
create table BILDER
(
    B_OID                  OID20                not null,
    BT_MOID                MOID2                not null,
    BF_MOID                MOID2                not null,
    B_NAME                 varchar(50)          not null,
    B_KUERZEL              char(8)                      ,
    B_BESCHREIBUNG         varchar(255)                 ,
    B_BILD                 IMAGE                not null,
    B_QUELLE               varchar(255)         not null,
    primary key (B_OID)                                 ,
    unique (B_NAME, BF_MOID, BT_MOID)                   ,
    foreign key (BT_MOID) references BILDTYP
    on update restrict on delete restrict               ,
    foreign key (BF_MOID) references BILDFORMAT
    on update restrict on delete restrict
);
comment on table BILDER is 'Bilder zu Staedten, Personen und technischen Objekten';

/* ========================================================== */
/*   Table: S_B_ZUORDNUNG                                     */
/* ========================================================== */
create table S_B_ZUORDNUNG
(
    S_OID                  ROID10             not null,
    B_OID                  ROID20             not null,
    primary key (S_OID, B_OID)                       ,
    foreign key (S_OID) references STAEDTE
    on update restrict on delete restrict            ,
    foreign key (B_OID) references BILDER
    on update restrict on delete restrict
);
comment on table S_B_ZUORDNUNG is 'Bilder zur Stadt';

/* ========================================================== */
/*   Table: P_B_ZUORDNUNG                                     */
/* ========================================================== */
create table P_B_ZUORDNUNG
(
    P_OID                  ROID10               not null,
    B_OID                  ROID20               not null,
    primary key (P_OID, B_OID)                          ,
    foreign key (P_OID) references PERSONEN
    on update restrict on delete restrict               ,
    foreign key (B_OID) references BILDER
    on update restrict on delete restrict
);
comment on table P_B_ZUORDNUNG is 'Bilder zur Person';

/* ========================================================== */
/*   Table: TO_B_ZUORDNUNG                                    */
/* ========================================================== */
create table TO_B_ZUORDNUNG
(
    TO_OID                 ROID20             not null,
    B_OID                  ROID20             not null,
    primary key (TO_OID, B_OID)                      ,
    foreign key (TO_OID) references TOBJEKTE
    on update restrict on delete restrict            ,
    foreign key (B_OID) references BILDER
    on update restrict on delete restrict
);
comment on table TO_B_ZUORDNUNG is 'Bilder zu technischem Objekt';
```

```
/* ========================================================== */
/*   Table: MUSEUMS_ART                                       */
/* ========================================================== */
create table MUSEUMS_ART
(
    M_A_OID                MOID2               not null,
    M_A_ART                varchar(128)        not null,
    primary key (M_A_OID)                                ,
    unique(M_A_ART)
);
comment on table MUSEUMS_ART is 'Unterscheidungsmerkmale von Museen (siehe Testdaten)';

/* ========================================================== */
/*   Table: MUSEEN                                            */
/* ========================================================== */
create table MUSEEN
(
    M_OID                  OID10                 not null,
    M_NAME                 varchar(255)          not null,
    ADR_OID                ROID20                not null,
    M_A_OID                MOID2                 not null,
    M_OEFFNUNG             varchar(255)                   ,
    M_ANFAHRT              varchar(255)                   ,
    M_TELEFON              char(25)                       ,
    M_FAX                  char(20)                       ,
    M_EMAIL                char(30)                       ,
    M_HOMEPAGE             char(50)                       ,
    M_VERANSTALTUNGEN      text                           ,
    M_TOURISTINFO          text                           ,
    M_INFOMATERIAL         text                           ,
    primary key (M_OID)                                   ,
    foreign key (ADR_OID) references ADRESSEN
    on update restrict on delete restrict                 ,
    foreign key (M_A_OID) references MUSEUMS_ART
    on update restrict on delete restrict                 ,
    unique(M_NAME)
);
comment on table MUSEEN is 'oeffentliche Museen, Ausstellungen, Sammlungen, ...';

/* ========================================================== */
/*   Table: M_TEXTE                                           */
/* ========================================================== */
create table M_TEXTE
(
    M_T_OID                OID10                 not null,
    M_OID                  ROID10                not null,
    M_T_BEZEICHNUNG        varchar(100)          not null,
    M_T_SCHLAGWOERTER      varchar(255)                   ,
    M_T_TEXTE              text                  not null,
    M_T_QUELLE             varchar(255)          not null,
    primary key (M_T_OID)                                 ,
    foreign key (M_OID) references MUSEEN
    on update restrict on delete restrict                 ,
    unique(M_OID, M_T_BEZEICHNUNG)
);
comment on table M_TEXTE is 'Textdokumente, Pressemitteilungen etc. von bzw. zu Museen';

/* ========================================================== */
/*   Table: VEREINE                                           */
/* ========================================================== */
create table VEREINE
(
    V_OID                  OID10                 not null,
    V_NAME                 varchar(255)          not null,
    ADR_OID                ROID20                not null,
    V_KUERZEL              char(15)                       ,
```

```
    V_ZIEL                varchar(255)                ,
    V_TELEFON             char(25)                    ,
    V_FAX                 char(20)                    ,
    V_EMAIL               char(30)                    ,
    V_HOMEPAGE            char(40)                    ,
    primary key (V_OID)                               ,
    foreign key (ADR_OID) references ADRESSEN
    on update restrict on delete restrict             ,
    unique(V_NAME)
);
comment on table VEREINE is 'Vereine, die sich mit technikgeschichtlichen Aufgaben beschaeftigen';

/* ========================================================== */
/*   Table: V_K_PERSONEN                                      */
/* ========================================================== */
create table V_K_PERSONEN
(
    V_K_P_OID             OID10               not null,
    V_K_VORNAME           char(15)            not null,
    V_K_NAME              char(20)            not null,
    ADR_OID               ROID20                      ,
    V_K_TELEFON           char(25)                    ,
    V_K_EMAIL             char(30)                    ,
    primary key (V_K_P_OID)                           ,
    foreign key (ADR_OID) references ADRESSEN
    on update restrict on delete restrict             ,
    unique (V_K_VORNAME, V_K_NAME)
);
comment on table V_K_PERSONEN is 'Kontaktperson zum Verein';

/* ========================================================== */
/*   Table: V_K_P_ZUORDNUNG                                   */
/* ========================================================== */
create table V_K_P_ZUORDNUNG
(
    V_OID                 ROID10               not null,
    V_K_P_OID             ROID10               not null,
    primary key (V_OID, V_K_P_OID)                      ,
    foreign key (V_OID) references VEREINE
    on update restrict on delete restrict               ,
    foreign key (V_K_P_OID) references V_K_PERSONEN
    on update restrict on delete restrict
);
comment on table V_K_P_ZUORDNUNG is 'Zuordnung von Kontaktpersonen zu Vereinen';

/* ========================================================== */
/*   Table: V_TEXTE                                           */
/* ========================================================== */
create table V_TEXTE
(
    V_T_OID               OID10               not null,
    V_OID                 ROID10              not null,
    V_T_BEZEICHNUNG       varchar(100)        not null,
    V_T_SCHLAGWOERTER     varchar(255)                ,
    V_T_TEXTE             text                not null,
    V_T_QUELLE            varchar(255)        not null,
    primary key (V_T_OID)                             ,
    foreign key (V_OID) references VEREINE
    on update restrict on delete restrict             ,
    unique(V_OID, V_T_BEZEICHNUNG)
);
comment on table V_TEXTE is 'Textdokumente, Pressemitteilungen etc. von bzw. zu Vereinen';
```

```
/* ========================================================== */
/*   Table: V_BETREIBT_M                                      */
/* ========================================================== */
create table V_BETREIBT_M
(
    V_OID                  ROID10                not null,
    M_OID                  ROID10                not null,
    primary key (V_OID, M_OID)                          ,
    foreign key (V_OID) references VEREINE
    on update restrict on delete restrict               ,
    foreign key (M_OID) references MUSEEN
    on update restrict on delete restrict
);
comment on table V_BETREIBT_M is 'Verein betreibt Museum, Ausstellung oder hat Sammlung etc.';

/* ========================================================== */
/*   Table: V_BES_S                                           */
/* ========================================================== */
create table V_BES_S
(
    V_OID                  ROID10                not null,
    S_OID                  ROID10                not null,
    primary key (V_OID, S_OID)                          ,
    foreign key (V_OID) references VEREINE
    on update restrict on delete restrict               ,
    foreign key (S_OID) references STAEDTE
    on update restrict on delete restrict
);
comment on table V_BES_S is 'Verein beschaeftigt sich mit Stadtgeschichte';

/* ========================================================== */
/*   Table: V_BES_P                                           */
/* ========================================================== */
create table V_BES_P
(
    V_OID                  ROID10                not null,
    P_OID                  ROID10                not null,
    primary key (V_OID, P_OID)                          ,
    foreign key (V_OID) references VEREINE
    on update restrict on delete restrict               ,
    foreign key (P_OID) references PERSONEN
    on update restrict on delete restrict
);
comment on table V_BES_P is 'Verein beschaeftigt sich mit technikgeschichtlich relevanter Person';

/* ========================================================== */
/*   Table: V_BES_TO                                          */
/* ========================================================== */
create table V_BES_TO
(
    V_OID                  ROID10                not null,
    TO_OID                 ROID20                not null,
    primary key (TO_OID)                                ,
    foreign key (V_OID) references VEREINE
    on update restrict on delete restrict               ,
    foreign key (TO_OID) references TOBJEKTE
    on update restrict on delete restrict
);
comment on table V_BES_TO is 'Verein beschaeftigt sich mit technischem Objekt (pflegt, bewahrt, ...)';
```

```
/* ========================================================== */
/*   Table: M_ENT_S                                           */
/* ========================================================== */
create table M_ENT_S
(
    M_OID                 ROID10               not null,
    S_OID                 ROID10               not null,
    primary key (M_OID, S_OID)                          ,
    foreign key (M_OID) references MUSEEN
    on update restrict on delete restrict               ,
    foreign key (S_OID) references STAEDTE
    on update restrict on delete restrict
);
comment on table M_ENT_S is 'Museum enhaelt technikgeschichtliche Informationen zur Stadt';

/* ========================================================== */
/*   Table: M_ENT_P                                           */
/* ========================================================== */
create table M_ENT_P
(
    M_OID                 ROID10               not null,
    P_OID                 ROID10               not null,
    primary key (M_OID, P_OID)                          ,
    foreign key (M_OID) references MUSEEN
    on update restrict on delete restrict               ,
    foreign key (P_OID) references PERSONEN
    on update restrict on delete restrict
);
comment on table M_ENT_P is 'Museum enthaelt Informationen zur technikgeschichtlich relevanten Person';

/* ========================================================== */
/*   Table: M_ENT_TO                                          */
/* ========================================================== */
create table M_ENT_TO
(
    M_OID                 ROID10               not null,
    TO_OID                ROID20               not null,
    primary key (TO_OID)                                ,
    foreign key (M_OID) references MUSEEN
    on update restrict on delete restrict               ,
    foreign key (TO_OID) references TOBJEKTE
    on update restrict on delete restrict
);
comment on table M_ENT_TO is 'Museum enthaelt technisches Objekt';

/* ========================================================== */
/*   Table: LITERATUR_ART                                     */
/*          L_ART  '1' - ISBN Buchnummer                      */
/*                 '2' - ISSN Zeitschriftennummer             */
/*                 '3' - DDR  Buchnummer                      */
/*                 ...                                        */
/* ========================================================== */
create table LITERATUR_ART
(
    L_A_OID               MOID2              not null,
    L_A_ART               varchar(50)        not null,
    primary key (L_A_OID)                             ,
    unique (L_A_ART)
);
comment on table LITERATUR_ART is 'Unterscheidungsmerkmale von Literatur';
```

```
/* ========================================================== */
/*   Table: LITERATUR                                         */
/* ========================================================== */
create table LITERATUR
(
    L_OID                  OID10               not null,
    KENNZEICHNUNG          char(20)            not null,
    L_A_OID                numeric(2,0)        not null,
    VERLAG                 char(20)            not null,
    TITEL                  char(100)           not null,
    L_DATUM                char(20)                    ,
    L_ORT                  char(50)                    ,
    L_BEZUGSQUELLE         varchar(255)                ,
    primary key (L_OID)                                ,
    foreign key (L_A_OID) references LITERATUR_ART
    on update restrict on delete restrict              ,
    unique(TITEL,VERLAG)
);
comment on table LITERATUR is 'Literaturempfehlung zu Museum, Stadt, Person, technischem Objekt';


/* ========================================================== */
/*   Table: AUTOREN                                           */
/* ========================================================== */
create table AUTOREN
(
    A_OID                  OID10                 not null,
    A_VORNAME              char(15)              not null,
    A_NAME                 char(20)              not null,
    primary key (A_OID)                                  ,
    unique (A_VORNAME, A_NAME)
);
comment on table AUTOREN is 'Autoren der Buecher';

/* ========================================================== */
/*   Table: B_VON_A_GESCHRIEBEN                               */
/* ========================================================== */
create table B_VON_A_GESCHRIEBEN
(
    L_OID                  ROID10                 not null,
    A_OID                  ROID10                 not null,
    primary key (L_OID, A_OID)                             ,
    foreign key (L_OID) references LITERATUR
    on update restrict on delete restrict                  ,
    foreign key (A_OID) references AUTOREN
    on update restrict on delete restrict
);
comment on table B_VON_A_GESCHRIEBEN is 'Autorenzuordnung';

/* ========================================================== */
/*   Table: S_IN_L_NACHZULESEN                                */
/* ========================================================== */
create table S_IN_L_NACHZULESEN
(
    S_OID                  ROID10                not null,
    L_OID                  ROID10                not null,
    primary key (S_OID, L_OID)                             ,
    foreign key (S_OID) references STAEDTE
    on update restrict on delete restrict                  ,
    foreign key (L_OID) references LITERATUR
    on update restrict on delete restrict
);
comment on table S_IN_L_NACHZULESEN is 'Literatur zur Stadt';
```

```
/* ========================================================== */
/*   Table: P_IN_L_NACHZULESEN                                */
/* ========================================================== */
create table P_IN_L_NACHZULESEN
(
    P_OID                  ROID10               not null,
    L_OID                  ROID10               not null,
    primary key (P_OID, L_OID)                          ,
    foreign key (P_OID) references PERSONEN
    on update restrict on delete restrict               ,
    foreign key (L_OID) references LITERATUR
    on update restrict on delete restrict
);
comment on table P_IN_L_NACHZULESEN is 'Literatur zur Person';

/* ========================================================== */
/*   Table: TO_IN_L_NACHZULESEN                               */
/* ========================================================== */
create table TO_IN_L_NACHZULESEN
(
    TO_OID                 ROID20               not null,
    L_OID                  ROID10               not null,
    primary key (TO_OID, L_OID)                         ,
    foreign key (TO_OID) references TOBJEKTE
    on update restrict on delete restrict               ,
    foreign key (L_OID) references LITERATUR
    on update restrict on delete restrict
);
comment on table TO_IN_L_NACHZULESEN is 'Literatur zu technischem Objekt';

/* ========================================================== */
/*   Table: M_IN_L_NACHZULESEN                                */
/* ========================================================== */
create table M_IN_L_NACHZULESEN
(
    M_OID                  ROID10               not null,
    L_OID                  ROID10               not null,
    primary key (M_OID, L_OID)                          ,
    foreign key (M_OID) references MUSEEN
    on update restrict on delete restrict               ,
    foreign key (L_OID) references LITERATUR
    on update restrict on delete restrict
);
comment on table M_IN_L_NACHZULESEN is 'Literatur zum Museum, Ausstellung, Sammlung';
```

B.2 Physisches Datenmodell der DB-TGMV

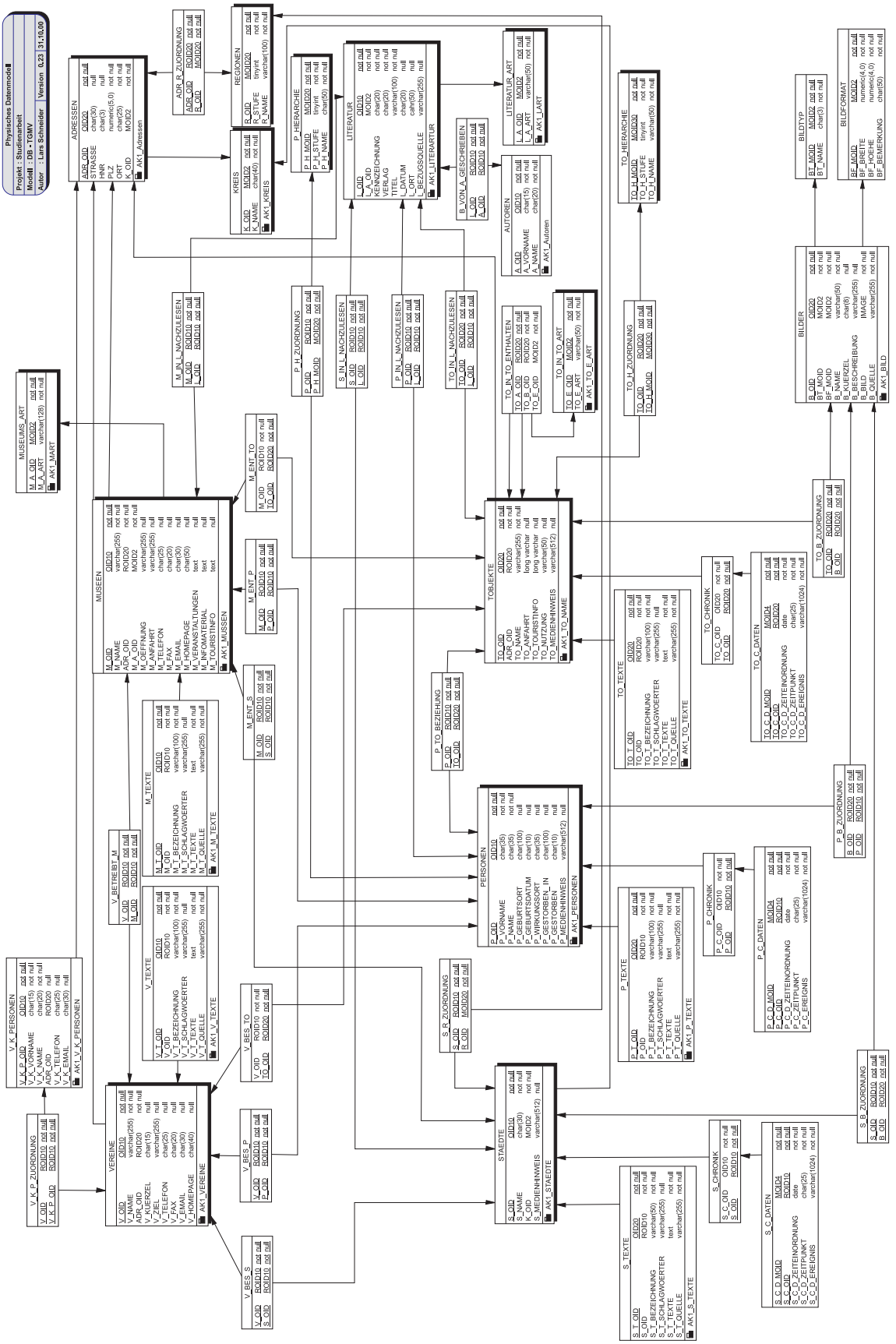

Abbildung B.1: Physisches Datenmodell

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Abi97] Serge Abiteboul. Querying semi-structured data. In *Proceedings of the International Conference on Database Theory*, 1997.

[BM98] Henning Behme and Stefan Mintert. *Extensible Markup Language (XML) — in der Praxis.* Addison Wesley Longman Verlag GmbH, Bonn, 1998.

[DD98] Chris J. Date and Hugh Darwen. *SQL – Der Standard: SQL/92 mit den Erweiterungen CLI und PSM.* Addison-Wesley, Bonn, 1998.

[Fer99] Reginald Ferber. Data Mining und Information Retrieval. Vorlesungsskript, Technische Universität Darmstadt, Darmstadt, 1998/99.

[Fuh97] Norbert Fuhr. Information Retrieval, Skriptum zur Vorlesung. Vorlesungsskript, Universität Dortmund, Dortmund, Mai 1997.

[Kir98] John Kirkwood. *Official Sybase Internals: Designing and Troubleshooting for High Performance.* ITP Company, London, 1998.

[Por99] Beate Porst. Untersuchungen zu Datentyperweiterungen für XML-Dokumente und ihre Anfragemethoden am Beispiel von DB2 und Informix. Diplomarbeit, Universität Rostock, Fachbereich Informatik, 1999.

[Rij79] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, London, second edition, 1979.

[SH97] Gunter Saake and Andreas Heuer. *Datenbanken: Konzepte und Sprachen.* MITP–Verlag GmbH, Bonn, 1997.

[SH99] Gunter Saake and Andreas Heuer. *Datenbanken: Implementierungstechniken.* MITP–Verlag GmbH, Bonn, 1999.

[Syb99a] Sybase, Inc. and its subsidiaries. *Adaptive Server Anywhere Version 6.0.3 - Reference Manual*, June 1999.

[Syb99b] Sybase, Inc. and its subsidiaries. *Adaptive Server Anywhere Version 6.0.3 - User's Guide*, June 1999.

[Syb99c] Sybase, Inc. and its subsidiaries. *SQL Modeler Version 6.1.3 - User's Guide*, January 1999.