Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

DEPARTMENT INFORMATION

Bachelorarbeit

Semantic Web:
Speicherung von Ontologien in einer relationalen Datenbank

vorgelegt von
Johannes-Arne Pauli

Studiengang Bibliotheks- und Informationsmanagement

erster Prüfer: Prof. Dr. Franziskus Geeb
zweiter Prüfer: Prof. Dr. Martin Gennis

Hamburg, September 2010

Abstract

In der vorliegenden Arbeit geht es um die Erstellung eines relationalen Datenbankmodells, in dem sich Ontologien speichern und sinnvoll abfragen lassen. Das Datenbankmodell soll später innerhalb der Chatbotprogrammierung verwendet werden. Als erstes werden einige Grundlagen aus dem Bereich der Wissensrepräsentation bis hin zu Ontologien erwähnt. Dann wird auf den Aufbau und auf die Funktion des Semantic Web eingegangen sowie auf die Grundlagen von relationalen Datenbanken, deren Normalisierung und auf die Abfragesprache SQL. Nach einer Beschreibung des verwendeten Datenbankmanagementsystems und der Ontologien, die als Beispiel in der Datenbank gespeichert wurden, wird der Aufbau des erstellten Datenbankmodells sowie dessen Funktionalität erläutert. Zuletzt wird dieses Modell anhand von beispielhaften Abfragen in beiden verwendeten Ontologien überprüft.

Schlagworte:

Relationale Datenbanken, SQL, Ontologien, OWL, Semantic Web, Wissensmodelle, Wissensrepräsentation

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

1NF	Erste Normalform
2NF	Zweite Normalform
3NF	Dritte Normalform
BCNF	Boyce-Codd-Normalform
DBMS	Datenbankmanagementsystem
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Manipulation Language
ER-Diagramm	Entity-Relationship-Diagramm
ER-Modell	Entity-Relationship-Modell
GPL	General Public License
OLS	Ontology Lookup Service
OWL	Web Ontology Language
PHP	PHP: Hypertext Preprocessor
RDBMS	Relationales Datenbankmanagementsystem
RDF	Resource Description Framework
RDFS	RDF Schema
SQL	Structured Query Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language

1 Einleitung

Im Jahr 2008 hat Andre Eisenmenger in seiner Diplomarbeit für das Departement Information der HAW Hamburg ein Konzept für die Nutzung von Ontologien im Bereich von Chatbots entwickelt. Dabei hat der Autor ein Online-Lexikon in eine OWL-Ontologie umgesetzt, die von einem Chatbot abgefragt werden sollte (vgl. EISENMENGER 2008, S. 29 ff.). In seiner Schlussbemerkung spricht er davon, eine kleine Verhaltensverbesserung des Chatbots durch die Nutzung einer Ontologie erreicht zu haben, was an der Unterscheidung von Synonymen und Stichworten, der Verweisung auf zugehörige Kategorien und der Verwendung von Tripeln liegt (vgl. EISENMENGER 2008, S. 51 f.).

Aus der Arbeit von Eisenmenger ergibt sich die Frage, ob es auch möglich ist, eine Ontologie in Form einer relationalen Datenbank abzubilden. Das würde den Vorteil bringen, dass man diese Ontologie innerhalb der Chatbotprogrammierung leichter abfragen könnte. Konkret würde das heißen, dass man z. B. eine Ontologie, die in einer mit MySQL erstellten Datenbank gespeichert wäre, mit Hilfe der Skriptsprache PHP abfragen könnte.

Die Recherchen zu diesem Thema in der Fachliteratur und im Internet ergaben wenige relevante Treffer. Zwei im Internet gefundene Möglichkeiten sollen an dieser Stelle erwähnt werden.

Der Ontologieeditor *Protégé* besitzt eine Exportfunktion, mit der sich in *Protégé* erstellte Ontologien in eine Datenbank exportieren lassen. Leider werden dabei alle Daten in eine einzige Tabelle geschrieben, die dadurch unübersichtlich wird und somit auch nicht abfragbar ist. Die Gründe für diese Vorgehensweise sind auf den Projektseiten von *Protégé* beschrieben (vgl. PROTÉGÉ 2010).

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
frame	VARBINARY(310)	☐	☑	☐	☐	☐	☐	☐	
frame_type	SMALLINT(6)	☐	☑	☐	☐	☐	☐	☐	
slot	VARBINARY(310)	☐	☑	☐	☐	☐	☐	☐	
facet	VARBINARY(310)	☐	☑	☐	☐	☐	☐	☐	
is_template	BIT(1)	☐	☑	☐	☐	☐	☐	☐	
value_index	INT(11)	☐	☑	☐	☐	☐	☐	☐	
value_type	SMALLINT(6)	☐	☑	☐	☐	☐	☐	☐	
short_value	VARCHAR(310)	☐	☐	☐	☐	☐	☐	☐	NULL
long_value	MEDIUMTEXT	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 1.1: Tabelle des Datenbankexports von *Protégé*

Ein anderes Beispiel findet sich auf den Internetseiten des *European Bioinformatics Institute*. Hier werden Datenbankexporte der Ontologien zum Download angeboten, die dieses Institut im Rahmen des *Ontology Lookup Service (OLS)* zur Ansicht bereitstellt. Bei diesen Ontologien handelt es sich hauptsächlich um biologische Ontologien mit hierarchischen Strukturen. Die Datenbankexporte der Ontologien lassen sich nach Begriffen, Synonymen, Notationen, Querverweisen und nach hierarchischen Beziehungen durchsuchen (vgl. OLS 2010).

Tables (18 items)

Add Table	annotation	dbxref
dbxref_qualifier_value	ojb_dlist	ojb_dlist_entries
ojb_dmap	ojb_dmap_entries	ojb_dset
ojb_dset_entries	ojb_hl_seq	ojb_lockentry
ojb_nrm	ontology	relationship_type
stats_loaded_ontolo...	stats_loader_run	term
term_path		

Abbildung 1.2: Tabellen der Datenbank des OLS

In beiden Beispielen geht es darum, Ontologien in relationalen Datenbanken abzubilden. Die Datenbankexporte des OLS sind dabei ein gutes Muster in Bezug auf die Umsetzung der oben genannten Fragestellung. Sie sind jedoch ausschließlich für die Nutzung innerhalb eines bestimmten

Fachbereiches vorgesehen. Somit enthält die Datenbank des OLS nicht die Möglichkeit, Ontologien mit unterschiedlicher Struktur aufzunehmen. Um diesen Aspekt soll die Fragestellung erweitert werden, d. h. um die Flexibilität der zu erstellenden Datenbank in Bezug auf die Struktur der enthaltenen Ontologien.

1.1 Zielsetzung

In dieser Arbeit wird versucht, ein Datenbankmodell zu entwickeln, mit dem sich Ontologien unterschiedlicher Struktur in einer relationalen Datenbank abbilden lassen. Diese Datenbanken sollen später innerhalb der Chatbotprogrammierung verwendet werden. Chatbotprogrammierung und Chatbots sind jedoch nicht Thema der Arbeit. Auch nicht die Möglichkeiten, die Skriptsprachen wie PHP in Bezug auf das Abfragen und Bearbeiten von Datenbanken bieten. Vielmehr soll ein relationales Datenbankmodell erarbeitet werden, das die Möglichkeiten und die Funktionalität der Ontologiesprache OWL bietet. Die Untersuchung der Funktionalität des Modells wird im letzten Teil der Arbeit anhand von Abfragen in zwei beispielhaften Ontologien mit unterschiedlicher Komplexität gezeigt.

1.2 Inhaltlicher Aufbau

Die Arbeit ist in acht Hauptkapitel unterteilt. Sie beginnt mit einer Übersicht über verschiedene Begriffsordnungen aus dem Bereich der Wissensrepräsentation. Es folgt ein Kapitel über die Funktion und den Aufbau des Semantic Web, sowie ein Kapitel zu Datenbanken, in dem Informationen zum relationalen Datenbankmodell, zu ER-Modellen und zum Erstellen und Abfragen von Datenbanken enthalten sind. Ferner werden das in der Arbeit verwendete Datenbankmanagementsystem und die verwendeten Ontologien erwähnt. Das sechste Kapitel enthält schließlich die Beschreibung des zu erstellenden Datenbankmodells. Dieses Modell wird im siebten Kapitel anhand von mehreren Abfragen in beiden Ontologien getestet.

2 Grundlagen aus der Wissensrepräsentation

In diesem ersten theoretischen Kapitel geht es um einige begriffliche Grundlagen aus der Wissensrepräsentation. Zuerst wird auf die Beziehung zwischen Symbolen, Objekten und Begriffen im semiotischen Dreieck eingegangen. Darauf folgen Erläuterungen des Aufbaus und der Anwendung von Taxonomien, Thesauri, Topic Maps sowie Ontologien, wobei auch auf deren semantische Stärke eingegangen wird.

2.1 Das semiotische Dreieck

„Das semiotische Dreieck (Bedeutungsdreieck) illustriert die Interaktion zwischen Worten (oder allgemeiner: Symbolen), Begriffen und realen Dingen in der Welt. Worte […] können die Essenz einer Referenz […] nicht vollständig erfassen. Dennoch gibt es eine Korrespondenz zwischen Wort, Begriff und Ding“ (EHRIG 2006, S. 471).

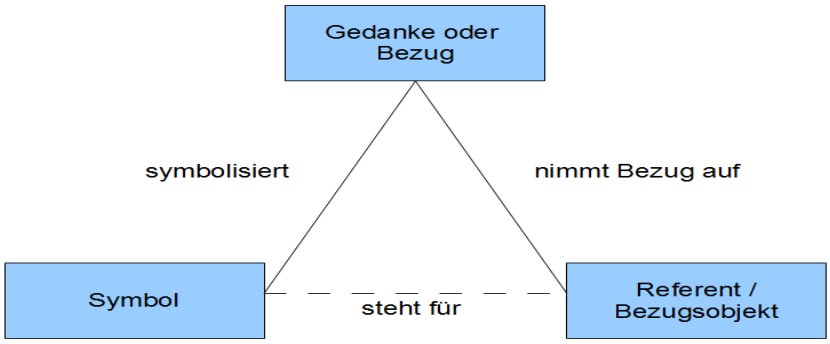


Abbildung 2.1: Das Semiotische Dreieck nach Ogden und Richards (STOCK 2008, S. 51)

Bei Ogden und Richards besteht diese Korrespondenz aus dem Symbol, dem Gedanken oder Bezug und dem Referenten oder Bezugsobjekt. Die Beziehung zwischen dem Gedanken und dem Symbol sind folgendermaßen: In der Richtung vom Bezug zum Symbol wird ein Gedanke ausgespro-

chen, während in umgekehrter Richtung ein Symbol in Gedanken gefasst wird, wobei ein Gedanke immer auch Bezug auf einen Referenten, ein Objekt, nimmt. Zwischen dem Symbol und dem Bezugsobjekt gibt es nur eine indirekte Verbindung, bei der das Symbol, vermittelt durch den Gedanken, für das Objekt steht (vgl. STOCK 2008, S. 51).

In der Informationswissenschaft wird an die Stelle des Gedankens der *Begriff* gesetzt, der wie bei Ogden und Richards durch *Benennungen* sprachlich repräsentiert wird. Der *Begriff* ist hier eine Klasse, unter die Objekte mit bestimmten Merkmalen fallen (vgl. STOCK 2008, S. 52).

In der Informationsverarbeitung dagegen werden die Begriffe *Gedanke* oder *Bezug* häufig durch den Begriff des *Konzeptes* ersetzt, um Klassen von Objekten und deren typische Eigenschaften zu beschreiben (vgl. STUCKENSCHMIDT 2009, S. 8).

2.2 Begriffsordnungen und deren semantische Ausdruckskraft

„Die Semantik (Bedeutungslehre) ist das Teilgebiet der Sprachwissenschaft (Linguistik), das sich mit Sinn und Bedeutung von Sprache beziehungsweise sprachlichen Zeichen befasst“ (BLUMAUER 2006, S. 9).

Die Semantik ist also mit der Syntax und der Pragmatik Teilgebiet der Semiotik. Während sich die Syntax mit den Beziehungen zwischen den Zeichen untereinander beschäftigt, geht es in der Semantik um die Beziehungen zwischen den Zeichen und Objekten bzw. Gegenständen der Außenwelt. Die Pragmatik handelt von den Beziehungen der Zeichen gegenüber Interpreten und Kontexten (vgl. BLUMAUER 2006, S. 10).

2.2.1 Taxonomie

Eine erste strukturell orientierte Form der Wissensrepräsentation ist die Taxonomie und in diesem Zusammenhang auch die Klassifikation. Eine Klassifikation wird definiert als eine „Einteilung eines Gegenstandbereiches in

Klassen, die systematische Einordnung in Klassen, entsprechend Gruppen gleicher Dinge“ (KIENREICH 2006, S. 362). Bei einer Klassifikation wird also eine Menge von Begriffen (Objekten) in eine meistens hierarchisch organisierte Menge von Klassen geordnet (vgl. KIENREICH 2006, S. 362).

Taxonomien werden als biologische Systematiken, aber auch im informationstechnischen Umfeld verwendet, z. B. bei der Organisation von Dateiobjekten in Betriebssystemen. Die Nachteile einer Taxonomie bestehen in der relativen Starrheit einer Klassifikationsstruktur, der Linearität der Zugangsmöglichkeiten zu Objekten und in den Einschränkungen von Relationen von Objekten untereinander (vgl. KIENREICH 2006, S. 362 f.).

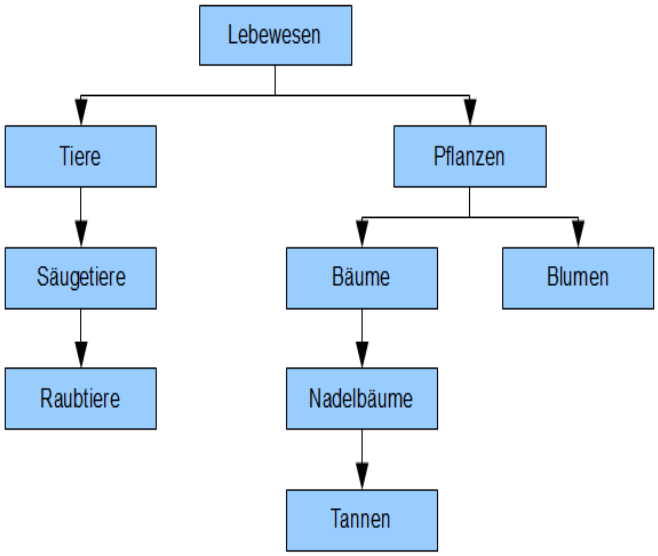


Abbildung 2.2: Beispiel einer Taxonomie

2.2.2 Thesaurus

Einen semantischen Mehrwert gegenüber einer Taxonomie bieten Thesauri.

„Ein Thesaurus im Bereich der Information und Dokumentation ist eine ge-

ordnete Zusammenstellung von Begriffen und ihren vorwiegend natürlichsprachlichen Bezeichnungen, die in einem Dokumentationsgebiet zum Indexieren, Speichern und Wiederauffinden dient“ (DIN1463/1 1987, S. 2).

Ein Thesaurus ist erstens durch eine terminologische Kontrolle gekennzeichnet, indem Synonyme möglichst vollständig erfasst werden, Homonyme und Polyseme besonders gekennzeichnet werden und jedem Begriff eine eindeutige Bezeichnung zugeordnet wird; zweitens dadurch, dass Beziehungen zwischen Begriffen dargestellt werden (vgl. DIN1463/1 1987, S. 2). Bei diesen Beziehungen handelt es sich grob gesagt um Hierarchierelationen, Äquivalenzrelationen und Assoziationsrelationen (vgl. DIN1463/1 1987, S. 5 f.).

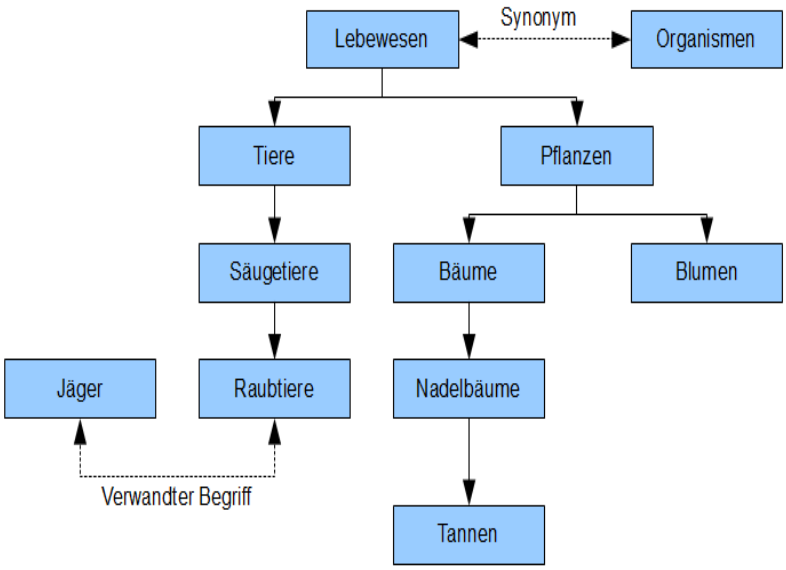


Abbildung 2.3: Beispiel eines Thesaurus

In Abbildung 2.3 lässt sich erkennen, dass sich der semantische Gehalt in einem Thesaurus im Vergleich mit einer Taxonomie erhöht. Über die Synonymrelation wird festgehalten, dass *Organismen* ein Synonym von *Lebewesen* ist, während über die Assoziationsrelation beschrieben ist, dass

Raubtiere auf die Jagd gehen, der Begriff *Jäger* also ein verwandter Begriff von *Raubtier* ist. Auch Hierarchien lassen sich ablesen. Ein Thesaurus beschränkt sich allerdings auf die drei genannten Relationen.

2.2.3 Topic Map

Topic Maps ist ein Modell um Dokumente und Teile von Dokumenten im Web semantisch zu beschreiben und zu kategorisieren. Eine Topic Map ist ein Set von verknüpften Themenfeldern und bietet einen inhaltsorientierten Index zu einer Dokumentensammlung (vgl. DACONTA 2003, S. 167).

Das Ziel von Topic Maps ist die bessere Navigation und Suche in Internetressourcen und anderen Dokumenten, und der Austausch von Metadaten (vgl. WIKIPEDIA 2010 a).

Dabei hat eine Topic Map den Grundstock von Taxonomien und Thesauri, geht aber in ihrer Ausdrucksstärke über diese hinaus. Sie besteht aus Topics (Themen), Relationen und zugeordneten Dokumenten und Instanzen (Occurences) (vgl. KOOS 2005, S. 74).

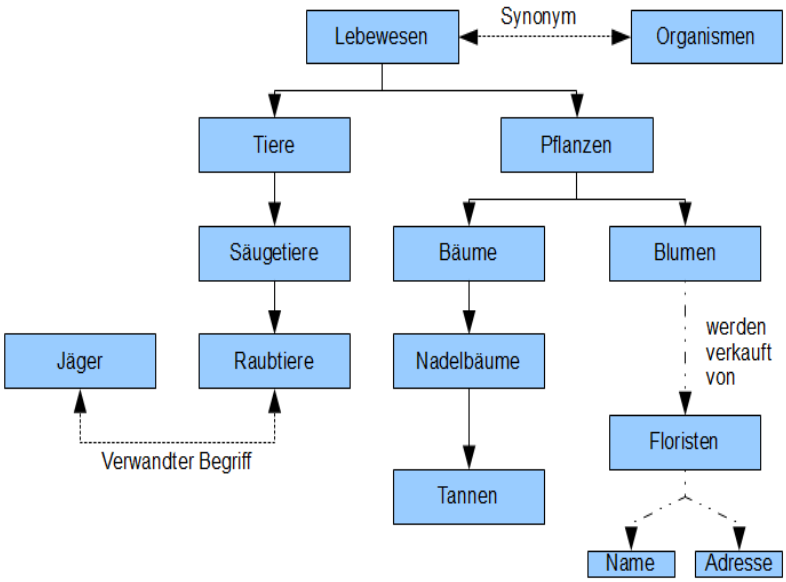


Abbildung 2.4: Beispiel einer Topic Map

Mit einer Topic Map lassen sich zusätzliche Relationen abbilden, wodurch sich die Ausdrucksstärke gegenüber dem Thesaurus erhöht. Es kann jetzt dargestellt werden, dass Blumen bei Floristen verkauft werden. Außerdem lassen sich Namen und Adressen von Floristen zuordnen.

2.2.4 Ontologie

Der Begriff *Ontologie* stammt aus der Philosophie und bedeutet seit der Metaphysik von Aristoteles die *Lehre vom Sein* (vgl. STOCK 2008, S. 255). In der Informationswissenschaft wurde dieser Begriff durch die Definition von Tom Gruber aus dem Jahr 1993 geprägt:

“An ontology is a specification of a conceptualization“ (GRUBER 1993, S. 1).

Demnach ist eine Ontologie eine formale, explizite Spezifikation einer Terminologie, die den Zweck hat, Begriffe eines Wissensbereiches gemeinsam zu nutzen. Dabei ist sie sowohl für die Mensch-Maschine-Interaktion als auch für die Zusammenarbeit von Computersystemen geschaffen (vgl. STOCK 2008, S. 255). In Bezug auf die Konzeptualisierung sei noch einmal an das semiotische Dreieck und den Begriff des *Konzeptes* erinnert. Die Rolle einer Ontologie besteht darin, diese Konzepte zu formalisieren, die im semiotischen Dreieck die Verbindung zwischen Symbolen einer informationstechnischen Darstellung und den Erwartungen an das dadurch dargestellte Objekt herstellen (vgl. STUCKENSCHMIDT 2009, S. 8).

Die potentielle Bedeutung von Ontologien in der Informationsverarbeitung liegt darin, bestimmte Ausschnitte der realen Welt in eine Darstellungsform zu überführen, die mit Computern manipuliert werden kann, um Veränderungen in der realen Welt abzubilden (vgl. STUCKENSCHMIDT 2009, S. 5).

Anwendungsfelder für Ontologien in der Informatik sind Kommunikation, automatisches Schließen und Repräsentation sowie Wiederverwendung von Wissen. Relevante Bereich sind dabei die künstliche Intelligenz, Datenbanken und Informationssysteme, aber auch Anwendungsbereiche aus der

Medizin, dem Rechtswesen oder der Wirtschaftsinformatik (vgl. HESSE 2002, S. 477 f.).

In einer Begriffsordnung müssen folgende Aspekte vorhanden sein, um sagen zu können, dass eine Ontologie vorliegt:

- es muss eine standardisierte Ontologiesprache verwendet werden
- unter Einsatz von terminologischer Logik muss automatisch geschlussfolgert werden können
- es muss nach Allgemeinbegriffen und Instanzen unterschieden werden, bzw. diese müssen vorkommen
- neben Hierarchierelationen müssen spezifische Relationen verwendet werden (vgl. STOCK 2008, S. 256 f.).

Eine Ontologie hat demzufolge folgende Bestandteile:

- *Begriffe* (engl.: concepts), die auch Klassen genannt werden und hierarchisch angeordnet werden können
- *Instanzen* (engl.: individuals), die das zur Verfügung stehende Wissen einer Ontologie in Form von Objekten darstellen
- *Relationen* zwischen Instanzen gleichen Typs
- *Axiome*, d. h. immer wahre Aussagen in einer Ontologie, die nicht aus anderen Begriffen abgeleitet werden können.

Außerdem ist es in einer Ontologie möglich, Relationen und Eigenschaften von Begriffen zu vererben (vgl. WIKIPEDIA 2010 b).

Bei dem Begriff *Ontologie* muss eine Unterscheidung zwischen *lightweight*- und *heavyweight* Ontologien gemacht werden. Während bei einer *heavyweight* Ontologie alle oben genannten Bestandteile vorhanden sind, fehlt bei einer *lightweight* Ontologie die Möglichkeit, Axiome oder Einschränkungen zu definieren (vgl. STOCK 2008, S. 256).

Von den vier genannten Begriffsordnungen weisen Ontologien die größte semantische Ausdruckskraft auf. Aus diesem Grund sind sie auch eine der Grundlagen des Semantic Webs.

3 Das Semantic Web

Im folgenden Kapitel geht es um das Semantic Web, das eine Erweiterung des heutigen Web darstellt. Es wird die Funktion des semantischen Webs beschrieben sowie der Aufbau des Semantic Web erläutert.

3.1 Die Funktion des Semantic Web

Die Idee des Semantic Web basiert auf zwei Zielen: Zum einen soll das Web kollaborativer werden, zum anderen soll es für Maschinen verständlich und somit verarbeitbar gemacht werden (vgl. DACONTA 2003, S. 1).

Das heutige Web birgt drei Probleme: Zum ersten gibt es eine unüberschaubare Menge von Informationen, die im Netz nur schwer gefunden werden können und deren Repräsentation auf den Menschen ausgerichtet ist. In diesem Zusammenhang wäre eine semantische, d. h. inhaltliche, Suche der stichwortbasierten Suche, wie sie von heutigen Suchmaschinen ausgeführt wird, vorzuziehen. Zum zweiten sind die vorhandenen Informationen durch die dezentrale Struktur und Organisation des Web heterogen, z. B. durch verschiedene Dateiformate und Kodierungstechniken oder durch verschiedene Strukturen von Homepages. Zum dritten ist es möglich, dass Informationen nicht explizit im Web verfügbar sind, sondern nur implizit erschlossen werden können (vgl. HITZLER 2008, S. 10).

Der Ansatz des Semantic Web ist eine Möglichkeit zur Lösung dieser Probleme, da Informationen von vornherein so zur Verfügung gestellt werden, dass sie von Maschinen verarbeitet werden können. Dazu sind einheitliche und offene Standards zur Beschreibung von Informationen nötig. Mit diesen sollen Informationen zwischen verschiedenen Anwendungen und Plattformen ausgetauscht und zueinander in Beziehung gesetzt werden können. Weiterhin müssen mit dem Semantic Web neue Informationen aus gegebenen geschlussfolgert werden können, was Logiken bedingt, um aus spezifiziertem Wissen implizite Informationen zu extrahieren (vgl. HITZLER 2008, S. 11 f.).

In Bezug auf das Semantic Web spielen Ontologien eine wichtige Schlüsselrolle, da sie das Vokabular eines bestimmten Wissensgebietes in Form von Begriffen, Instanzen und Relationen liefern (vgl. GRIMM 2007, S. 78 f.) und Maschinen dabei unterstützen, „Inhalte im Web interpretieren zu können, anstatt sie einfach darzustellen und damit sämtliche Vernetzungstätigkeiten dem Menschen zu überlassen“ (BLUMAUER 2006, S. 12). Dabei wird zwischen Ontologien auf drei verschiedenen Ebenen unterschieden: der *top level*, *middle level* und *lower domain level* Ontologie. Während eine *top level* Ontologie allgemeine Informationen enthält, die alle anderen Ontologien umfasst, ist in einer *lower domain* Ontologie nur noch spezifisches Wissen eines bestimmten Wissensbereiches enthalten (vgl. DACONTA 2003, S. 230 f.).

3.2 Aufbau des Semantic Web

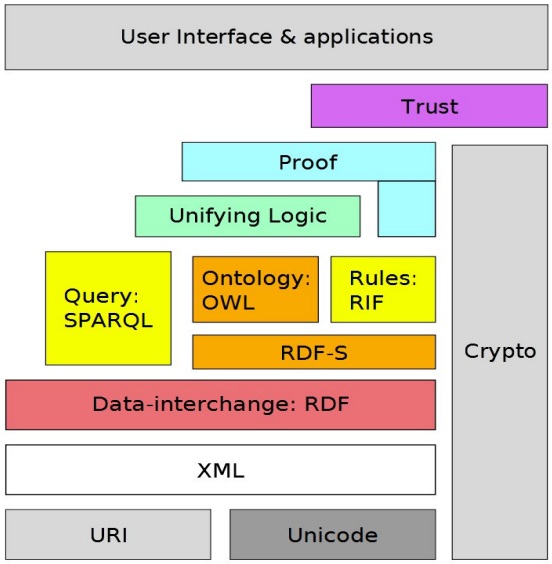


Abbildung 3.1: Semantic Web Stack des W3C (WIKIPEDIA 2010 c)

Abbildung 3.1 beschreibt den Aufbau des Semantic Web nach der Definition des World Wide Web Consortium (W3C). Im Folgenden wird auf die grundlegenden Standards XML, RDF(S) und OWL eingegangen. RDF(S) und OWL sind Ontologiesprachen, d. h. eine Ontologie ist im Sinne des Se-

mantic Web immer ein in RDF(S) oder OWL erstelltes Dokument. XML wird eigentlich eher dem klassischen Web als dem Semantic Web zugeordnet (vgl. HITZLER 2008, S. 11 f.).

3.2.1 XML

Die Extensible Markup Language (XML) ist eine Auszeichnungssprache um strukturierte Dokumente zu beschreiben. Ein XML Dokument besteht aus einfachem Text und Textmarken, den sogenannten Tags, die von einer Anwendung interpretiert werden (vgl. BREITMANN 2007, S. 58). „Daten in XML-Dokumenten können elektronisch verbreitet und verarbeitet werden, somit ist XML eine Art universelles Daten-Austauschformat“ (HITZLER 2008, S. 18).

XML wurde vom W3C zum ersten mal im Februar 1998 als Empfehlung veröffentlicht und liegt inzwischen in der Version 1.1 vor. In Bezug auf das Semantic Web spielt XML eine zentrale Rolle als Basistechnologie für RDF, RDFS und OWL, den anwendungsnahen Technologien der höheren Schichten (vgl. GRÜTTER 2008, S. 140 f.).

```
-<Bibliothek>
 -<Buch>
    <Titel> Semantic Web </Titel>
    <Autor Geschlecht="weiblich"> Karin Breitmann </Autor>
    <Bild Quelle="foto.png"/>
  </Buch>
</Bibliothek>
```

Abbildung 3.2: Beispiel eines XML-Dokumentes

Ein XML-Dokument beginnt mit einer XML-Deklaration, die das Dokument als XML-Dokument kennzeichnet und die die Versionsnummer des verwendeten XML-Standards enthält. Dabei kann auch ein Kodierungsformat für

die Zeichenkodierung angegeben werden (vgl. HITZLER 2008, S. 19). In obigem Beispiel könnte das die Zeile *<?xml version="1.0" encoding="UTF-8"?>* sein.

Weiterhin kommen XML-Elemente vor, die aus einem Start-Tag, einem End-Tag und dem Inhalt des Elementes bestehen. Eine Ausnahme bilden hier leere Elemente, die nur aus einem Tag bestehen. Der Inhalt eines Elementes können andere Elemente sein (vgl. DACONTA 2003, S 34). Ein Beispiel für XML-Elemente sind die Elemente *Titel* und *Autor* in Abbildung 3.2. Das Element *Bild* ist ein leeres Element.

Ein XML-Dokument hat genau ein Wurzel-Element, dessen Start-Tag in der ersten Zeile nach der XML-Deklaration stehen muss. Alle weiteren Elemente sind innerhalb des Wurzelelementes verschachtelt (vgl. HITZLER 2008, S. 20). Das Wurzelelement in Abb. 3.2 ist das Element *Bibliothek*.

Der Start-Tag eines Elementes oder ein leeres Element kann ein oder mehrere Attribute beinhalten, die in der Form *Attribut-Name="Attribut-Wert"* dargestellt werden (vgl. BREITMANN 2007, S. 58). Im Beispiel hat das Element *Autor* das Attribut *Geschlecht* und das leere Element *Bild* das Attribut *Quelle*.

Ein XML-Dokument kann auch eine *XML-Schema*-Deklaration enthalten. Mit XML-Schema werden Struktur, Inhalt und Semantik von XML-Dokumenten definiert und beschrieben (vgl. GRÜTTER 2008, S. 141).

3.2.2 Namensräume mit XML

Mit XML-Schema werden auch sogenannte *Namensräume* unterstützt, mit denen global einzigartige Namen für Elemente und Attribute eines XML-Dokumentes erstellt werden. Das hat den Vorteil, dass in verschiedenen XML-Dokumenten die gleichen Namen benutzt werden können und dass diese vermischt werden können, ohne dass Ambiguitäten auftreten (vgl. DACONTA 2003, S. 42).

„Der Name des Namensraums hat die Form einer URI und wird als Wert des Attributes *xmlns* deklariert“ (HITZLER 2008, S. 28). Eine URI (Uniform

Resource Identifier) ist ein einheitlicher Bezeichner für abstrakte oder physikalische Ressourcen. Im WWW werden diese Bezeichner als URLs benutzt, um Web-Seiten oder andere Dateien zu bezeichnen (vgl. HITZLER 2008, S. 26). Für das Beispiel in Abbildung 2.3 könnte man einen Namensraum mit dem Namen *buecherhallen* folgendermaßen deklarieren:

<Bibliothek xmlns:buecherhallen="http://www.buecherhallen.de">.

Wenn Namensräume angewendet werden, muss der Name eines XML-Elementes aus zwei Teilen bestehen: dem Präfix und einem lokalen Teil (vgl. DACONTA 2003, S. 43). Der Start-Tag des Elements *Buch* aus obigem Beispiel mit dem Namensraum *buecherhallen* versehen, würde also folgendermaßen aussehen:

<buecherhallen:Buch>.

3.2.3 RDF

Mit dem Resource Description Framework (RDF) und RDF Schema (RDFS) werden im Semantic Web die Metadaten- und die Schemaschicht realisiert. Beides sind verbindliche Empfehlungen des W3C, die 2001 entwickelt und im Mai 2006 überarbeitet wurden (vgl. GRÜTTER 2008, S. 141 f.).

Während man mit XML Daten strukturiert, werden mit RDF Informationen über die Daten selbst angereichert und mit anderen Informationen in Beziehung gesetzt und verknüpft. Mit diesen Metadaten soll im Semantic Web ein Geflecht semantischer Beziehungen zwischen den Ressourcen des Web aufgebaut werden, auf das Anwendungen aufsetzen können (vgl. BIRKENBIHL 2006, S. 80 f.).

Um Ressourcen zu beschreiben, werden sie als RDF-Statement in der Form *Subjekt – Prädikat – Objekt* definiert, wobei auch ein Objekt eine Ressource und Subjekt für weitere Statements sein kann (vgl. BIRKENBIHL 2006, S. 80). Statt von einem RDF-Statement spricht man aufgrund seiner drei Teile auch häufig von Tripeln. Zur genauen Bezeichnung der Ressourcen werden die bereits oben angesprochenen URI-Verweise ver-

wendet (vgl. PASSIN 2004, S. 22).

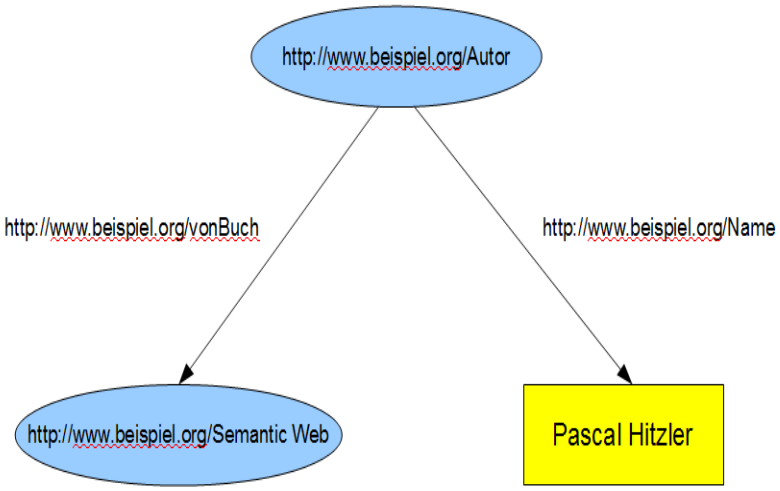


Abbildung 3.3: Beispiel für RDF-Graph

In Abbildung 3.3 wird ein einfacher RDF-Graph mit zwei RDF-Statements dargestellt. Ressourcen, also Subjekte und Objekte, werden in Ovalen dargestellt, Prädikate als Pfeile. Rechtecke stehen für Objekte mit literalen Werten und bezeichnen keine Ressource.

Die Darstellung als Graph ist für den Entwurf von RDF-Aussagen durch den Menschen gut geeignet. Für Computer und Netzwerke braucht man aber eine Serialisierung von RDF. Das W3C hat sich für eine Serialisierung in XML entschieden (vgl. BIRKENBIHL 2006, S. 81). In Abbildung 3.4 ist obiger Graph in *RDF/XML* dargestellt.

```
<rdf:RDF>
 -<rdf:Description rdf:about="http://www.beispiel.org/Autor">
   -<bsp:vonBuch>
      <rdf:Description rdf:about="http://www.beispiel.org/Semantic Web"></rdf:Description>
    </bsp:vonBuch>
    <bsp:Name> Pascal Hitzler </bsp:Name>
  </rdf:Description>
</rdf:RDF>
```

Abbildung 3.4: Beispiel für ein RDF/XML Dokument

Die Elemente des Typs *<rdf:Description>* beschreiben Ressourcen, das Attribut *rdf:about* bezeichnet die URIs der Ressourcen. Die Prädikate sind in diesem Beispiel die Elemente *<bsp:von Buch>* und *<bsp:Name>*, für die der Namensraum *bsp* beliebig gewählt wurde.

Mit RDF lassen sich auch mehrwertige Beziehungen darstellen. Dies erreicht man entweder durch leere Knoten, durch Container (offene Listen) oder durch Collections (geschlossene Listen) (vgl. HITZLER 2008, S. 57 ff.).

3.2.4 RDF Schema

„Während RDF im Wesentlichen dazu dient, grundlegende Aussagen über die Beziehungen zwischen Einzelobjekten (Individuen) zu treffen, bietet RDFS die Möglichkeit, terminologisches Wissen im Form von Klassen- und Propertyhierarchien und deren Zusammenhängen zu spezifizieren“ (HITZLER 2008, S. 86).

Mit RDF-Schema lassen sich in RDF Klassen von Objekten und deren Subklassen definieren sowie Eigenschaften Werte- und Gültigkeitsbereiche zuordnen. Klassen werden mit *rdfs:Resource* und *rdfs:Class* definiert, Prädikate mit *rdfs:subClassOf*, *rdfs:domain*, *rdfs:range* und *rdf:type*. Auch Eigenschaften sind Ressourcen und gehören zur Klasse *rdf:Properties*. Dadurch lassen sich mit RDFS Taxonomien und einfache Ontologien erstellen (vgl.

BIRKENBIHL 2006, S. 82).

Folgendes Beispiel eines RDFS-Dokuments drückt aus, dass die Klasse *Primaten* eine Unterklasse der Klasse *Saeugetiere* ist:

```
<rdf:RDF>
 −<rdfs:Class rdf:about="http://www.beispiel.org/Primates">
    <rdfs:label xml:lang="de">Primaten</rdfs:label>
    <rdfs:subClassOf rdfs:resource="http://www.beispiel.org/Mammalia"/>
  </rdfs:Class>
 −<rdfs:Class rdf:about="http://www.beispiel.org/Mammalia">
    <rdfs:label xml:lang="de">Saeugetiere</rdfs:label>
  </rdfs:Class>
</rdf:RDF>
```

Abbildung 3.5: Beispiel für ein RDFS-Dokument (vgl. HITZLER 2008, S. 82)

3.2.5 OWL

„Die Web Ontology Language (OWL) ist eine Sprache zur Definition von strukturierten, Web-basierten Ontologien“ (GRÜTTER 2008, S. 151). Die Grundlage für OWL ist DAML+OIL. OWL ist seit 2004 eine verbindliche Empfehlung des W3C (vgl. GRÜTTER 2008, S. 151). Das Ziel von OWL ist es, „Terminologien für einen bestimmten Zusammenhang zu erstellen, Eigenschaften besser einschränken zu können, die logischen Charakteristiken von Eigenschaften und die Äquivalenz von Begriffen zu definieren“ (BIRKENBIHL 2006, S. 83). Dabei erweitert OWL das Vokabular von RDF-Schema um neue Klassen und Prädikate (vgl. BIRKENBIHL 2006, S. 83).

„Die Grundbausteine von OWL bestehen aus Klassen und Propertys, wie sie schon von RDF(S) bekannt sind, sowie aus Individuen, die als RDF-Instanzen von Klassen deklariert werden“ (HITZLER 2008, S. 129). Klassen, Properties (oder auch Rollen) und Individuen können mit OWL definiert werden, ebenso Klassenbeziehungen und Beziehungen zwischen Individuen. Außerdem lassen sich auf Klassen logische Konstruktoren anwenden.

Rollen können eingeschränkt werden, auf verschiedene Weise zueinander bezogen werden und mit Eigenschaften ausgestattet werden (vgl. HITZLER 2008, S. 129 ff.).

OWL ist unterteilt in drei verschiedene Teilsprachen mit jeweils verschiedener Ausdrucksstärke: *OWL Lite*, *OWL DL* und *OWL Full*, wobei OWL Lite eine Teilsprache von OWL DL und diese wiederum eine Teilsprache von OWL Full ist (vgl. HITZLER 2008, S. 126).

OWL Lite ist vor allen Dingen für Klassifizierungshierarchien und einfache Bedingungen gedacht und liefert einen schnellen Migrationspfad für Thesauri und Taxonomien. Es soll mit Werkzeugen einfacher zu unterstützen sein als OWL DL und OWL Full und weist dabei eine geringere formale Komplexität auf als diese (vgl. GRÜTTER 2008, S. 151).

OWL DL liefert die größtmögliche Ausdruckskraft bei voller Verarbeitbarkeit und Entscheidbarkeit der Berechnungen. Entscheidbarkeit bedeutet hier, dass alle Berechnungen in einer endlichen Anzahl von Schritten abgeschlossen werden können. OWL DL enthält alle Sprachkonstrukte, die aber nur mit gewissen Einschränkungen verwendet werden können (vgl. GRÜTTER 2008, S. 151).

In OWL Full dürfen alle OWL- und RDF(S)-Sprachelemente uneingeschränkt benutzt werden, das Resultat muss nur gültiges RDF sein (vgl. HITZLER 2008, S. 151). OWL Full bietet die größtmögliche Ausdruckskraft und die syntaktische Freiheit von RDF, aber keine Gewährleistung der rechnergestützten Verarbeitbarkeit. Es ist nicht entscheidbar, d. h. dass keine Software Schlussfolgerungen für den vollen Umfang von OWL Full unterstützen kann (vgl. GRÜTTER 2008 , 151 f.).

4 Datenbanken

„Eine Datenbank ist eine Kollektion von Daten, die in einer Datenbasis gespeichert sind. Eine Datenbank verfügt außerdem noch über ein Datenbankmanagementsystem, das sämtliche Benutzeranfragen an die Datenbasis bearbeitet. Welche Daten zu einer Datenbank gehören, ist benutzerdefiniert und richtet sich nach dem (inhaltlich möglichst geschlossenem) Ausschnitt aus der realen Welt, der in der betreffenden Datenbank beschrieben werden soll“ (SCHUBERT 2004, S. 50).

4.1 Grundlagen

Im Folgenden wird auf Datenbanksysteme, Datenbankmanagementsysteme und Datenbankmodelle eingegangen.

4.1.1 Datenbanksysteme

Ein Datenbanksystem (oder auch: eine Datenbank) besteht aus vier Schichten:

- *Hardware*: dies sind alle physikalischen Geräte, mit denen das System aufgebaut ist (z. B. Datenbankserver, Clientrechner und Peripheriegeräte). Auf dieser Schicht basiert das ganze System.
- *Daten*: die Fakten, die im Datenbanksystem gespeichert werden. Diese müssen geschützt werden; nur das Datenbankmanagementsystem hat darauf Zugriff.
- *Software*: die wichtigste Software des Systems ist das Datenbankmanagementsystem, das die Verwaltung der Daten übernimmt. Dazu kommen Datenbankanwendungsprogramme für Anwender, Dienstprogramme für Datenbankadministratoren, -designer und Programmierer und die Betriebssysteme des Servers und der Clientrechner.

- *Personen*: Anwender, Datenbankadministratoren, Datenbankdesigner und Programmierer (vgl. GEISLER 2009, S. 46 ff.).

4.1.2 Datenbankmanagementsysteme

„Ein Datenbankmanagementsystem (DBMS) ist ein Softwaresystem (i.e. Sammlung von Programmen), das dem Benutzer das Erstellen und die Pflege einer Datenbank ermöglicht. Das DBMS ist folglich ein generelles Softwaresystem, das die Prozesse der Definition, Konstruktion und Manipulation von Datenbanken für verschiedene Anwendungen vereinfacht“ (ELMASRI 2005, S. 19).

Eine computergestützte Datenbank kann also mit Hilfe eines Datenbankmanagementsystems erstellt und gepflegt werden. Mit der Definition einer Datenbank ist die Spezifikation der Datentypen, die Struktur und die Einschränkungen der Daten gemeint. Die Konstruktion meint den Prozess des Speicherns der Daten auf ein Medium, während die Manipulation Anfragen an die Datenbank, Änderungen der Daten und das Erstellen von Berichten aus den Daten sein können (vgl. ELMASRI 2005, S. 10).

Ein Datenbankmanagementsystem sollte folgende Funktionen bereitstellen: Redundanzkontrolle, Zugriffsbeschränkungen, persistente Speicherung von Programmobjekten und Datenstrukturen, Ableitungen und Aktionen durch Verwendung von Regeln, Mehrbenutzerverwaltung der Daten, Darstellung komplexer Beziehungen zwischen Daten, Integritätseinschränkungen sowie die Datensicherung und Wiederherstellung (vgl. ELMASRI 2005, S. 36).

4.1.3 Datenbankmodelle

„Unter einem Datenbankmodell versteht man eine abstrahierte Darstellung der Daten und der zwischen diesen Daten bestehenden Beziehungen“ (GEISLER 2009, S. 54).

Es wird zwischen konzeptionellen und implementativen Datenbankmodellen unterschieden. Konzeptionelle Modelle beschäftigen sich damit, was in

die Datenbank aufgenommen werden soll und mit der logischen Struktur der Daten. Es geht um das Verstehen und Erfassen der Daten und der zwischen ihnen bestehenden Beziehungen. Ein konzeptionelles Modell ist das ER-Datenmodell. Dagegen geht es bei implementativen Modellen darum, wie die Daten in der Datenbank gespeichert werden und wie die zu erzeugenden Strukturen sowie deren Beziehungen zueinander aussehen. Beispiele dafür sind das Netzwerkmodell und das relationale Datenbankmodell (vgl. GEISLER 2009, S. 54).

Auch ein relationales Datenbanksystem hat ein Datenbankmanagementsystem, das sogenannte RDBMS. Der Unterschied zum DBMS besteht in der größeren Funktionalität eines RDBMS. So muss sich ein Anwender einer relationalen Datenbank aufgrund der Kapselung der physikalischen Datenspeicherung in einem RDBMS nur noch mit der logischen Struktur der Daten beschäftigen (vgl. GEISLER 2009, S. 62).

4.2 Das relationale Datenbankmodell

„Relationale Datenbanken dominieren zur Zeit den Markt. Sie stellen bemerkenswert einfache Mittel zur Darstellung und Manipulation von Daten zur Verfügung“ (ROLLAND 2003, S. 51). Nach einer Definition von E. F. Codd ist ein System relational, wenn „alle Werte in Relationen dargestellt sind und jeder Benutzerzugriff auf die Daten über deren Werte erfolgt […] und es relationale Operatoren für die Selektion, die Projektion und den Verbund unterstützt“ (MATTHIESSEN 2008, S. 31).

4.2.1 Aufbau relationaler Datenbanken

In einer relationalen Datenbank werden alle Daten in einer einfachen zweidimensionalen Tabelle gespeichert, die man *Relation* nennt. Relationen haben Überschriften und Einträge. Die Überschrift einer Relation besteht aus dem Namen der Relation und den Namen der Spalten der Relation. Diese Spalten nennt man *Attribute*. Attribute haben Wertebereiche, die sie annehmen können, und es gibt Operationen, die auf diese Werte anwendbar

sind. Diese Einschränkung der Daten nennt man die *Domäne* des Attributs. Die Einträge einer Relation ergeben sich aus deren Zeilen. Sie werden auch *Tupel* genannt und sind geordnete Listen von Werten (vgl. ROLLAND 2003, S. 51 ff.).

Berater

Berater_ID	Name	Vorname	Stundensatz
1	Meier	Helena	50,00 €
2	Fuchs	Ingo	45,00 €

Zeile / Tupel

Spalte / Attribut

Abbildung 4.1: Beispiel einer Relation (vgl. GEISLER 2009, S. 97)

Abbildung 4.1 zeigt die Relation *Berater*, die aus vier Attributen (*Berater_ID*, *Name*, *Vorname*, *Stundensatz*) und zwei Tupeln besteht. Das Attribut *Name* hat im ersten Tupel beispielsweise den Wert *Meier*. Einen Wert kann man auch ein *Datum* nennen (vgl. GEISLER 2009, S. 97).

4.2.2 Schlüssel

In relationalen Datenbanken gibt es drei Arten von Schlüsseln: Kandidatenschlüssel, Primärschlüssel und Fremdschlüssel. Mit einem Kandidatenschlüssel wird jedes Tupel einer Relation eindeutig identifiziert. Ein Kandidatenschlüssel ist dabei ein Attribut oder eine Menge von Attributen. Im zweiten Fall spricht man von einem zusammengesetzten Kandidatenschlüssel. Wenn es in einer Relation mehr als einen Kandidatenschlüssel gibt, muss ein Kandidatenschlüssel als Primärschlüssel ausgewählt werden. In einer Relation kann es beliebig viele Kandidatenschlüssel geben, aber nur einen Primärschlüssel. Gibt es nur einen Kandidatenschlüssel, ist dieser automatisch der Primärschlüssel (vgl. ROLLAND 2003, S. 57).

In obiger Abbildung ist *Berater_ID* sowohl ein Kandidatenschlüssel als auch der Primärschlüssel. Würde man z. B. auch die Steuernummern der Berater in der Tabelle festhalten wollen und wären diese eindeutig, so hätte man zwei Kandidatenschlüssel und müsste einen davon als Primärschlüssel auswählen.

In einer Relation darf es keine zwei Tupel geben, die den selben Primärschlüssel haben. Diese Bedingung nennt man Entitätsintegrität (vgl. MATTHIESSEN 2008, S. 41). Das bedeutet, dass die Tupel (oder auch Entitäten) einer Relation unterscheidbar sein müssen. Insbesondere die Werte des Primärschlüsselattributes müssen sich unterscheiden und dürfen nicht den Wert *NULL* annehmen (vgl. ROLLAND 2003, S. 58), sie dürfen also nicht keinen Wert haben.

„Ein Fremdschlüssel ist ein Attribut (oder eine Menge von Attributen), das in mehr als einer Tabelle vorkommt und in einer dieser Tabellen den Primärschlüssel bildet“ (ROLLAND 2003, S. 57). Mit Fremdschlüsseln werden Daten in verschiedenen Tabellen verknüpft, d. h. es werden Beziehungen zwischen Tabellen aufgebaut. Damit diese Beziehungen gültig sind, muss die Regel der referentiellen Integrität eingehalten werden. Die Regel der referentiellen Integrität besagt, dass jeder Wert in einem Fremdschlüsselattribut auch in der Relation vorkommen muss, in der dieses Attribut als Primärschlüssel auftritt. Jeder Wert eines Fremdschlüssels ist also eine Referenz auf einen Wert eines Primärschlüssels, der wirklich existiert. Existiert dieser nicht, muss der Wert des Fremdschlüssels auf *NULL* gesetzt werden (vgl. ROLLAND 2003, S. 57 ff.).

Ein Problem mit der referentiellen Integrität kann auftreten, wenn Tupel aus einer Relation gelöscht oder geändert werden, auf die sich Fremdschlüssel einer abhängigen Relation beziehen. Dieses Problem kann auf drei Arten gelöst werden:

- referenzierte Schlüssel dürfen nicht geändert oder gelöscht werden
- Löschungen und Änderungen von Primärschlüsseln werden an die abhängigen Fremdschlüssel weitergegeben
- abhängige Fremdschlüssel werden beim Löschen der Primärschlüssel auf *NULL* gesetzt (vgl. MATTHIESSEN 2008, S. 43).

Wollte man in obigem Beispiel einer Relation auch den Geburtsort der Berater in die Datenbank aufnehmen, könnte man eine neue Tabelle mit dem

Namen *Geburtsort* und einem Primärschlüsselattribut sowie dem Attribut *Ort_Name* erstellen und diese mithilfe eines Fremdschlüssels mit der Tabelle *Berater* verknüpfen.

4.2.3 Beziehungen

Mit der Hilfe von Schlüsseln können Tabellen einer Datenbank in Beziehung zueinander gesetzt werden. Zwischen diesen Tabellen gibt es 1:1-, 1:N- und N:M-Beziehungen, die in einem ER-Diagramm dargestellt werden können (vgl. GEISLER 2009, S. 113).

Bei einer 1:1-Beziehung ist ein Datensatz (Tupel) einer Tabelle mit genau einem Datensatz einer anderen verknüpft. Diese Art der Beziehung kommt in einer Datenbank nicht oft vor und man sollte bei einem Vorkommen dieser Beziehung in den meisten Fällen überlegen, ob man nicht beide Tabellen zusammenführt. Die 1:N-Beziehung ist dagegen die am häufigsten in Datenbanken verwendete Beziehungsart. Dabei steht ein Datensatz einer Tabelle mit beliebig vielen Datensätzen einer anderen Tabelle in Beziehung. Schließlich lassen sich mit einer N:M-Beziehung Datensätze beliebig miteinander verknüpfen. Eine N:M-Beziehung lässt sich in zwei 1:N-Beziehungen zerlegen, die über ein Zwischentabelle miteinander verknüpft werden (vgl. GEISLER 2009, S. 113 ff.).

In obigem Beispiel würde die genannte Tabelle *Geburtsort* zu einer 1:N-Beziehung führen, da genau ein Ort der Geburtsort von mehreren Beratern sein könnte. Um eine N:M-Beziehung herzustellen, könnte man eine Tabelle mit Kundendaten erstellen und voraussetzen, dass ein Berater mehrere Kunden haben kann, aber ein Kunde auch mehrere Berater. Diese Beziehungen würde man über eine Zwischentabelle realisieren.

4.2.4 Relationale Operatoren

Relationale Operatoren kann man sich als Funktionen vorstellen, die als Eingabewerte zwei Tabellen enthalten und eine Tabelle als Ausgabe zurückliefern, die aus beiden Eingangstabellen hervorgeht. So lassen sich,

zusätzlich zur Verknüpfung mit Primär- und Fremdschlüsseln, Tabellen in Beziehung setzen. Man bezeichnet eine Datenbank als relational vollständig, wenn sie folgende acht Operatoren unterstützt: *DIFFERENCE*, *DIVIDE*, *INTERSECT*, *JOIN*, *PRODUCT*, *PROJECT*, *SELECT* und *UNION*. Um relational genannt werden zu können, muss eine Datenbank minimal die Operatoren *SELECT*, *PROJECT* und *JOIN* unterstützen. Mit relationalen Operatoren werden aus vorhanden Tabellen neue, temporäre Tabellen erzeugt, die wieder gelöscht werden, sobald sie nicht mehr benötigt werden (vgl. GEISLER 2009, S. 106).

4.3 Das Entity-Relationship-Modell

Das Entity-Relationship-Modell ist eine der am weitesten verbreiteten Notationen zur konzeptuellen Beschreibung von Datenbanken und basiert auf einem Artikel von Peter Chen aus dem Jahre 1976. Aufgrund der grafischen Struktur eignet sich dieses Modell besonders gut zum konzeptionellen Entwurf von Datenbanken. Mit einem ER-Modell kann man die Struktur der zu erfassenden Daten erarbeiten und diese Struktur in eine relationale Datenbank überführen, um so von einem konzeptionellen Modell zu einer funktionsfähigen Datenbank zu kommen (vgl. GEISLER 2009, S. 67 f.).

Ein ER-Modell besteht im wesentlichen aus Entitäten und Beziehungen. Eine Entität ist dabei „eine eigenständige Einheit, die im Rahmen des betrachteten Modells eindeutig identifiziert werden kann“ (MATTHIESSEN 2008, S. 96). Sie wird „durch eine Menge von Eigenschaften (Attributen) beschrieben“ (MATTHIESSEN 2008, S. 97). Wie Entitäten zueinander in Beziehung stehen wird durch verschiedene Beziehungstypen dargestellt, die auch als Konnektivität bezeichnet werden. Mögliche Typen von Beziehungen wurden in Kapitel 4.2.3 genannt. Beziehungen werden durch ein Verb charakterisiert (vgl. GEISLER 2009, S. 68 f.).

Bei der grafischen Darstellung eines ER-Modells spricht man von einem Entity-Relationship-Diagramm. In der Darstellung nach Chen werden Entitäten als Rechtecke, Attribute als Ovale und Beziehungen als Rauten dargestellt. Der Typ der Beziehung wird neben die jeweiligen Entitäten ge-

schrieben, Primärschlüsselattribute werden unterstrichen (vgl. GEISLER 2009, S. 67 f.; vgl. GEISLER 2009, S. 140 ff.).

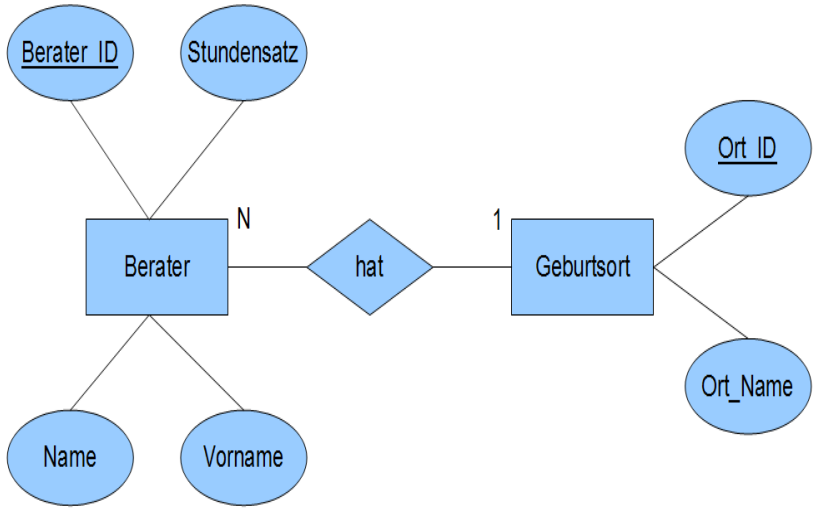


Abbildung 4.2: Beispiel eines ER-Diagramms nach Chen

Abbildung 4.2 stellt die in Kapitel 4.2.3 angesprochene 1:N-Beziehung der Entitäten *Berater* und *Geburtsort* samt ihrer Attribute in einem Chen-Diagramm dar.

4.4 Normalisierung

Durch ein semantisches Modellierungsverfahren wie der ER-Modellierung lassen sich ER-Diagramme in relationale Datenbanken transformieren. Damit die Datenstrukturen dieser Datenbanken auch effizient sind, muss man eine Normalisierung vornehmen. Um effizient zu sein, muss eine Datenbank folgende Eigenschaften besitzen:

- das Fehlen von Redundanz, d. h. es dürfen nicht an mehr als einer

Stelle gleiche Daten vorkommen

- der Einsatz von *NULL*-Werten sollte minimiert werden
- es darf zu keinem Datenverlust kommen.

Durch das Vorhandensein dieser Eigenschaften sollen Update-Anomalien, Einfügeanomalien und Löschanomalien verhindert werden. Update-Anomalien treten auf, wenn man beim Ändern von Daten gleiche Daten an mehreren Stellen einer Relation ändern müsste, da sonst die Datenbank inkonsistent werden würde. Eine Einfügeanomalie liegt vor, wenn man Daten nicht in eine Datenbank einfügen kann, ohne gleichzeitig andere einfügen zu müssen. Auch hier können Dateninkonsistenzen entstehen. Löschanomalien könne auftreten, wenn man durch das Löschen von bestimmten Daten auch andere Daten aus einer Relation entfernen würde, die eigentlich nicht gelöscht werden sollten (vgl. ROLLAND 2003, S. 83; vgl. GEISLER 2009, S. 38).

Der Prozess der Normalisierung wurde von Codd vorgeschlagen und umfasste ursprünglich drei Normalformen, die erste, die zweite und die dritte Normalform. Später schlugen Boyce und Codd eine stärkere Definition der 3NF, die Boyce-Codd-Normalform (BCNF), vor. Diese Normalformen basieren auf den funktionalen Abhängigkeiten zwischen den Attributen einer Relation (vgl. ELMASRI 2005, S. 317). Es gibt auch höhere Normalformen, die in der Praxis jedoch selten überprüft werden. Eine effiziente relationale Datenbank erhält man, wenn alle Relationen mindestens in der dritten Normalform, besser noch in BCNF sind (vgl. ROLLAND 2003, S. 95 f.).

4.4.1 Die erste Normalform

„Eine Relationsklasse befindet sich in der ersten Normalform, wenn zu jedem Attribut dieser Relationsklasse ein Wertebereich gehört, der nur atomare Werte enthält. Das bedeutet: solch ein Wertebereich darf also keine Werte enthalten, die aus mehreren Einzelwerten bestehen“ (SCHUBERT 2004, S. 286).

Folgende Abbildung zeigt eine Relation *Kunde,* in der Kundendaten, wie Namen und Telefonnummern eingetragen sind. Um diese Relation in die erste Normalform zu bringen, muss für jede eingetragene Telefonnummer eines Kunden ein weiteres Tupel erstellt werden. Außerdem wird das Attribut *Name* in *Vorname* und *Name* aufgeteilt.

Kunde

Kunde_ID	Name	Telefonnummern
1	Peter Meier	82234584; 3452267
2	Hans Müller	4385585; 42637777
3	Erna Schultze	23384888

Kunde (1NF)

Kunde_ID	Name	Vorname	Telefonnummer
1	Meier	Peter	82234584
1	Meier	Peter	3452267
2	Müller	Hans	4385585
2	Müller	Hans	42637777
3	Schultze	Erna	23384888

Abbildung 4.3: Tabelle in der ersten Normalform

In der Tabelle *Kunde* ist *Kunde_ID* der Primärschlüssel. Nach der Überführung in die erste Normalform ist dies aber nicht mehr so, da die Werte *1* und *2* mehrmals vorkommen. Hier könnte man nun einen zusammengesetzten Primärschlüssel aus den Feldern *Kunde_ID* und *Telefonnummer* verwenden, da Datensätze über diese Kombination immer eindeutig identifiziert werden können.

„Die Bestandteile des Primärschlüssels werden auch als Schlüsselattribute oder Primärattribute bezeichnet, wohingegen Attribute, die nicht Teil des Primärschlüssels sind, als Nicht-Schlüsselattribute oder Nicht-Primärattribute bezeichnet werden“ (GEISLER 2009, S. 183).

4.4.2 Die zweite Normalform

„Eine Tabelle befindet sich in der zweiten Normalform, wenn

- sie sich in der ersten Normalform befindet und

- es keine teilweisen Abhängigkeiten gibt, das heißt, kein Nicht-Schlüssel-Attribut ist nur von einem Teil des Primärschlüssels der Tabelle abhängig, in dem es enthalten ist.

Besitzt eine Tabelle nur ein einziges Primärschlüsselattribut, so befindet sich diese Tabelle automatisch in der zweiten Normalform, wenn sie sich in der ersten Normalform befindet.

In einem Satz zusammengefasst kann man auch sagen:

Jedes Attribut ist entweder vollständig von einem Schlüssel abhängig oder selbst ein Schlüssel“ (GEISLER 2009, S. 187).

Von einer vollständigen Abhängigkeit spricht man, wenn das abhängige Feld von allen Schlüsselfeldern der Tabelle abhängig ist. Ist dieses Feld dagegen nur von Teilen des Schlüssels abhängig, spricht man von der teilweisen Abhängigkeit (vgl. GEISLER 2009, S. 184).

In der Tabelle *Kunde (1NF)* in Abbildung 4.3 sind die Nichtschlüsselattribute *Name* und *Vorname* abhängig vom Schlüsselattribut *Kunde_ID*, das Teil des zusammengesetzten Primärschlüssels *Kunde_ID*, *Telefonnummer* ist. Es bestehen in dieser Relation also teilweise Abhängigkeiten. Um die Tabelle *Kunde (1NF)* in die zweite Normalform zu überführen, muss man sie in die Tabellen *Kunde* und *Kunde_Telefon* aufteilen.

Kunde

Kunde_ID	Name	Vorname
1	Meier	Peter
2	Müller	Hans
3	Schultze	Erna

Kunde_Telefon

Kunde_ID	Telefonnummer
1	82234584
1	3452267
2	4385585
2	42637777
3	23384888

Abbildung 4.4.: Tabellen in der zweiten Normalform

Die Relation *Kunde* hat nun den Primärschlüssel *Kunde_ID*, von dem alle Nichtschlüsselattribute vollständig abhängig sind. Die Relation *Kunde_Telefon* enthält den bisherigen zusammengesetzten Primärschlüssel *Kunde_ID, Telefonnummer* und keine Nichtschlüsselattribute mehr, da diese nicht vollständig von *Kunde_ID, Telefonnummer* abhängig sind. Beide Relationen sind nun redundanzfrei, da es für jeden Kunden und für jede Kombination aus Kunde und Telefonnummer nur einen Tupel in der jeweiligen Relation gibt.

4.4.3 Die dritte Normalform

„Eine Relation ist genau dann in der dritten Normalform, wenn sie (i) in 2NF und (ii) frei von transitiven Abhängigkeiten ist“ (ROLLAND 2003, S. 91).

Von einer transitiven Abhängigkeit spricht man, wenn eine Abhängigkeit zwischen zwei Nichtschlüsselattributen vorliegt. Entdeckt man eine transitive Abhängigkeit, müssen diese Attribute aus der entsprechenden Relation entfernt werden und aus diesen entfernten Attributen sowie dem Attribut, von dem sie abhängen, eine neue Relation erzeugt werden. Letzteres Attribut wird in der neuen Relation der Primärschlüssel und verbleibt in der Originalrelation als Fremdschlüssel (vgl. ROLLAND 2003, S. 91 f.).

Berater

Berater_ID	Name	Vorname	Aufgabe	Stundenlohn
1	Mustermann	Max	Finanzberater	15,00 €
2	Musterfrau	Hertha	IT-Berater	16,00 €

Berater (3NF)

Berater_ID	Name	Vorname	Aufgabe_ID
1	Mustermann	Max	1
2	Musterfrau	Hertha	2

Aufgabe (3NF)

Aufgabe _ID	Beschreibung	Stundenlohn
1	Finanzberater	15,00 €
2	IT-Berater	16,00 €

Abbildung 4.5: Tabellen in der dritten Normalform (vgl. GEISLER 2009, S. 188 ff.)

In obiger Abbildung lässt sich in der Relation *Berater* eine transitive Abhängigkeit zwischen den Attributen *Aufgabe* und *Stundenlohn* feststellen, ausgehend davon, dass der Stundenlohn abhängig ist von der Art der Beschäftigung. Bei der Überführung in die dritte Normalform wurden diese Attribute in die neue Relation *Aufgabe (3NF)* verschoben, die als Primärschlüssel das Schlüsselattribut *Aufgabe_ID* erhalten hat. Über einen Fremdschlüssel werden diese Daten in der Tabelle *Berater (3NF)* referenziert.

„Die dritte Normalform ist die gebräuchlichste Normalform für Datenbankanwendungen, da in dieser Normalform keine Datenanomalien mehr auftreten können“ (GEISLER 2009, S. 190).

4.4.4 Die Boyce-Codd-Normalform

„Eine Tabelle befindet sich in der Boyce-Codd-Normalform, wenn

- sie sich in der dritten Normalform befindet und
- jede Determinante ein Schlüsselkandidat ist.

Besitzt die Tabelle nur ein einziges Primärschlüsselattribut und befindet sie sich in der 3NF, so ist die Tabelle automatisch in der BCNF“ (GEISLER 2009, S. 193). Die Boyce-Codd-Normalform betrifft also nur Relationen mit zusammengesetzten Kandidatenschlüsseln (vgl. ROLLAND 2003, S. 95).

Eine Determinante ist ein Attribut, dessen Wert die Werte anderer Attribute bestimmt (vgl. GEISLER 2009, S. 193). Folgende Abbildung zeigt die Tabellen *Vorlesungsverzeichnis* und *Vorlesungen*, die sich beide in der dritten Normalform befinden. Die Tabelle *Vorlesungsverzeichnis* hat einen aus den Attributen *Zeit* und *Dozent* zusammengesetzten Primärschlüssel, der jedes Tupel eindeutig macht. Doch da auch jedem Dozenten ein Raum zugeordnet wird, bestimmt hier ebenso das Nichtschlüsselattribut *Raum* den Wert des Schlüsselattributs *Dozent*.

Vorlesungsverzeichnis

Zeit	Dozent	Raum	Vorlesungsnr.
19:00	Müller-Schubert	R12.1	1
19:00	Geisler	R14.3	2
15:00	Geisler	R14.3	1
17:00	Müller-Schubert	R12.1	1

Vorlesungen

Vorlesungsnr.	Vorlesung
1	Wirtschaftsenglisch
2	Marketing

Abbildung 4.6: Tabellen in 3NF, aber nicht in BCNF (vgl. ROLLAND 2003, S. 94)

Um das Problem in der Relation *Vorlesungsverzeichnis* zu lösen, muss zuerst der Primärschlüssel der Relation geändert werden (vgl. GEISLER 2009, S. 194). Der neue Primärschlüssel besteht nun aus den Schlüsselattributen *Zeit* und *Raum*. Das ehemalige Schlüsselattribut *Dozent* ist jetzt ein Nichtschlüsselattribut.

Vorlesungsverzeichnis

Zeit	Raum	Dozent	Vorlesungsnr.
19:00	R12.1	Müller-Schubert	1
19:00	R14.3	Geisler	2
15:00	R14.3	Geisler	1

Abbildung 4.7: Tabelle *Vorlesungsverzeichnis* mit geändertem Primärschlüssel

Des Weiteren besteht eine vollständige Abhängigkeit der Nichtschlüsselattribute *Dozent* und *Vorlesungsnr.* vom neuen zusammengesetzten Primärschlüssel. Da es aber auch eine teilweise Abhängigkeit zwischen *Dozent* und dem Schlüsselattribut *Raum* gibt, muss die Tabellenstruktur in die zweite Normalform versetzt werden (vgl. GEISLER 2009, S. 194). Dazu lagert man die teilweise Abhängigkeiten in die neue Tabelle *Raumzuordnung* aus. In den neu erstellten Tabellen bestehen jetzt nur noch vollständige Abhängigkeiten. In Tabelle *Vorlesungsverzeichnis* ist das Nichtschlüsselattribut *Vorlesungsnr.* von dem zusammengesetzten Primärschlüssel *Zeit, Raum* vollständig abhängig. In der Tabelle *Raumzuordnung* das Nichtschlüsselattribut *Dozent* von dem Primärschlüssel *Raum*.

Vorlesungsverzeichnis

Zeit	Raum	Vorlesungsnr.
19:00	R12.1	1
19:00	R14.3	2
15:00	R14.3	1
17:00	R12.1	1

Raumzuordnung

Raum	Dozent
R12.1	Müller-Schubert
R14.3	Geisler

Abbildung 4.8: Tabellen in Boyce-Codd-Normalform überführt

4.5 SQL (Structured Query Language)

SQL (Structured Query Language) ist die Standardabfragesprache für relationale Datenbanken. SQL ist aus der Vorläufersprache SEQUEL entstanden und wurde erstmals 1987 als Standard veröffentlicht. Die neueste Version von SQL stammt aus dem Jahr 2003 (vgl. MATTHIESSEN 2008, S. 161 f.).

SQL-Datenbanken sind im strengen Sinne nicht relational, da die Daten in Tabellen dargestellt werden, in denen identische Tupel grundsätzlich mehrfach vorkommen können. Dieses Problem soll durch das Konzept des Primärschlüssels verhindert werden (vgl. MATTHIESSEN 2008, S. 161). Außerdem ist SQL als Sprache deklarativ und nicht prozedural, d. h. es werden keine Prozeduren für die Abfrage von Daten angegeben, sondern eine Beschreibung der Daten, an denen man interessiert ist. Diese Beschreibung hat die Rolle einer Anweisung und wird vom System in eine Folge von Operationen aus der relationalen Algebra übersetzt (vgl. ROLLAND 2003, S. 100).

„Des Weiteren stellt SQL die Schnittstelle zwischen der relationalen Datenbank und dem Anwendungsprogramm dar. Die Sprache ist in erster Linie nicht für Endanwender gedacht, sondern für Systementwickler“ (MATTHIESSEN 2008, S. 161).

4.5.1 Datentypen in SQL

Es gibt folgende Datentypen in SQL:

- Exakt numerische Datentypen, wie INTEGER oder DECIMAL
- Angenähert numerische Datentypen, wie REAL oder FLOAT
- Zeichenketten, wie CHARACTER oder VARCHAR
- Datentypen, die das Datum und die Uhrzeit betreffen, wie DATE, TIME oder TIMESTAMP
- Logische Datentypen, nämlich BOOLEAN mit den Werten TRUE, FALSE und UNKNOWN (vgl. MATTHIESSEN 2008, S. 168 ff.).

4.5.2 SQL-Komponenten

SQL ist in drei Untersprachen unterteilt: DDL (Data Definition Language), DML (Data Manipulation Language) und DCL (Data Control Language) (vgl. GEISLER 2009, S. 209).

DDL stellt Befehle zur Verfügung, mit denen sich Datenbanken erstellen, ändern oder löschen lassen. Diese Befehle lauten CREATE, ALTER und DROP (vgl. GEISLER 2009, S. 212).

DML stellt vier Anweisungen zur Datenmanipulation zur Verfügung:

- SELECT für die Abfrage von Daten
- INSERT für das Einfügen von Daten
- UPDATE für das Verändern von Daten
- DELETE zum Löschen von Daten (vgl. ROLLAND 2003, S. 100).

DML-Befehle können sehr lang, verschachtelt und unübersichtlich werden. Die Befehle bestehen aus verschiedenen Parametern und Klauseln. Klauseln können Ausdrücke, Unterabfragen, logische Verknüpfungen, Prädikate und Aggregatfunktionen enthalten (vgl. GEISLER 2009, S. 213).

Mit DCL hat man in SQL Befehle zur Verfügung, mit denen sich Daten in der Datenbank schützen lassen. Die am häufigsten benutzten DCL-Befehle lauten GRANT und DENY. Mit GRANT kann man Benutzern oder Gruppen Rechte auf ein Objekt in der Datenbank zuweisen, z. B. Rechte auf den Befehl SELECT in einer bestimmten Tabelle. Mit DENY wird Benutzern der Zugriff auf auf eine Tabelle verwehrt (vgl. GEISLER 2009, S. 218).

4.5.3 Datenabfragen mit SQL

„Die wichtigste und am häufigsten ausgeführte Aufgabe einer Datenbankanwendung ist es, Daten aus der Datenbank auszuwählen und diese z. B. dem Benutzer anzuzeigen, in einem Bericht auszugeben oder weiterzuverarbeiten. Für all diese Aufgaben wird der Befehl SELECT verwendet, mit dem Datenbanken ausgewählt und an das aufrufende Programm zurückgeliefert werden“ (GEISLER 2009, S. 223).

„Eine SELECT-Anweisung setzt auf einer Menge von Relationen […] auf und beschreibt die Bedingungen, die an diese Relationen gestellt werden müssen, damit nur eine bestimmte Menge von Zeichen übrig bleibt. Man kann sich SELECT als eine Anweisung vorstellen, die aus den Tabellen und Sichten einer Datenbank eine weitere (temporäre) Tabelle aufbaut“ (ROLLAND 2003, S. 100 f.).

Die obligatorischen Komponenten einer SELECT-Anweisung sind:

- *SELECT* : Angabe der gewünschten Spalten
- *FROM* : Angabe der zu verknüpfenden Tabellen

Optional sind folgende Komponenten einer SELECT-Anweisung:

- *WHERE* : Selektionsbedingung für Tupel
- *GROUP BY* : Gruppenbildung bei gleichen Werten in den angegebenen Spalten
- *HAVING* : Selektionsbedingung für Gruppen

- *ORDER BY* : Sortierfolge der Tupel in der Ergebnistabelle (vgl. MATTHIESSEN 2008, S. 202).

Mit einer SELECT-Anweisung können auch mehrere Tabellen verknüpft werden (vgl. MATTHIESSEN 2008, S. 201). Bei einer solchen Abfrage spricht man auch von einem Join. Je nach der Art der Verknüpfung spricht man von einem Equi Join, Cross Join, Inner Join, Outer Join, Left Outer Join, Right Outer Join oder Full Outer Join (vgl. GEISLER 2009, S. 223).

Das folgende Beispiel ist eine Abfrage einer Datenbank für Filme:

```
SELECT Titel, Jahr, Anzahl, Nominierungen
FROM Film F, Auszeichnung A, Auszeichnung_has_Film AF
WHERE (F.ID=AF.Film_ID)
AND (A.ID=AF.Auszeichnung_ID)
AND (A.Name="Oscar")
AND (AF.Jahr >= 2000)
AND (AF.Anzahl > 0)
ORDER BY Titel ASC
```

Dieses Beispiel ist eine Inner-Join-Abfrage von drei Tabellen der Datenbank, nämlich der Tabellen *Film*, *Auszeichnung* und *Auszeichnung_has_Film*. Aus der Namensgebung der Tabellen erkennt man, dass *Film* und *Auszeichnung* in einer N:M-Beziehung zueinander stehen und *Auszeichnung_has_Film* die entsprechende Zwischentabelle ist. Durch die Verknüpfung von Schlüsselwerten wird die Join-Abfrage realisiert. Um sich Schreibarbeit zu sparen, wird hinter die Tabellennamen ein Alias geschrieben, der dann im SQL-Befehl weiterverwendet werden muss, z. B. *F* für die Tabelle *Film*.

Die Spalten *Titel*, *Jahr*, *Anzahl* und *Nominierungen* der drei Tabellen sollen

als Abfrageergebnis angezeigt werden, wobei das anzuzeigende Ergebnis durch die WHERE-Klausel eingeschränkt wird. Durch sie werden nur Filme angezeigt, die seit dem Jahr 2000 einen Oskar erhalten haben und nicht nur nominiert waren. Die Filme werden für die Anzeige alphabetisch aufsteigend nach dem Filmtitel sortiert.

5 Verwendetes RDBMS und Beispielontologien

5.1 MySQL und MySQL Workbench

Für die Bearbeitung des Themas wird MySQL Server in der Version 5.1.45 als RDBMS verwendet. MySQL Server ist eine freie Software unter der General Public License (GPL), die ursprünglich vom schwedischen Unternehmen MYSQL AB entwickelt wurde. MySQL AB wurde im Februar 2008 von Sun Microsystems übernommen, das wiederum im Januar 2010 von Oracle übernommen wurde. MySQL Server ist heute mit mehr als 6 Millionen Installationen das populärste Open-Source-Datenbankverwaltungssystem der Welt (vgl. WIKIPEDIA 2010 d).

Da MySQL auf der Datenbanksprache SQL basiert (vgl. MAURICE 2010, S. 312), kann mit den in Abschnitt 4.5. genannten Befehlen und Anweisungen gearbeitet werden.

Standard-Tabellentypen sind in MySQL die Storage-Engines InnoDB und MyISAM, wobei MyISAM unter Unix und InnoDB unter Windows als Standard-Engine installiert wird. InnoDB steht wie MySQL selbst unter der GPL und gewährleistet im Gegensatz zu MyISAM die referentielle Integrität über Fremdschlüssel (vgl. WIKIPEDIA 2010 e). Weitere Vorteile des InnoDB-Tabellentyps sind unter anderem die Unterstützung von Transaktionen, die automatische Fehlerbeseitigung und das Fehlen von Beschränkungen der Tabellengröße (vgl. KANNENGIESSER 2007, S. 646 f.). Aus diesen Gründen werden in der zu erstellenden Datenbank nur Tabellen des Typs InnoDB angelegt.

Für den Entwurf des Datenbankmodells wird die MySQL Workbench Community Edition in der Version 5.2.25 verwendet, die ebenfalls unter der General Public License steht. Mit diesem Tool lassen sich Entwurfs-, Entwicklungs- und Administrationsaufgaben im Bereich Datenbanken und MySQL durchführen (vgl. MYSQL WORKBENCH 2010).

Für die Bearbeitung des Themas sind folgende Funktionen der MySQL

Workbench relevant: es lassen sich komplexe ER-Modelle erstellen und in eine Datenbank überführen. Diese Datenbank lässt sich mit Daten füllen und über den SQL-Editor abfragen.

5.2 Verwendete Ontologien

Die Funktionalität des zu erstellenden Datenbankmodells wird anhand von zwei Ontologien getestet. Zum einen wird der strukturelle Teil der Ontologie verwendet, die Andre Eisenmenger in seiner Bachelorarbeit zum Thema Ontologien und Chatbots erstellt hat (vgl. EISENMENGER 2008, S. 29 ff.). Sie trägt den Namen *lookedup*. Da diese Ontologie aber nicht alle Bestandteile von OWL abdeckt, wird noch eine zweite, etwas umfangreichere Ontologie in einer Datenbank gespeichert und abgefragt.

Bei dieser zweiten Ontologie handelt es sich um eine Arbeit, die im Kurs *Wissensorganisation* des Masterstudiengangs *Informationswissenschaft und -management* der Hochschule für Angewandte Wissenschaften in Hamburg im Wintersemester 09/10 entstanden ist. Die Ontologie wurde von Sina Ingber und Patricia Wollschläger erstellt. Sie trägt den Titel *Erlebnis Hamburg* und soll jungen Nutzern eine strukturierte, durchsuchbare Auswahl an Aktivitäten und sportlichen Tätigkeiten in der Hansestadt Hamburg bieten (vgl. WOLLSCHLÄGER 2010, S. 1 f.). Mit der Ontologie sollen z. B. folgende Fragen beantwortet werden können:

- „Welche Möglichkeiten habe ich im Sommer draußen Sport zu machen?
- Was kann ich tagsüber in Hamburg kostengünstig (max. 10 Euro) unternehmen?
- Welche kulturellen Aktivitäten bieten sich in der Wintersaison an einem Sonntag an?
- Was kann ich nachmittags unternehmen, wenn ich nur eine Stunde Zeit habe?
- Wo kann ich abends am Wochenende tanzen gehen, ohne Eintritt

zu bezahlen?“ (WOLLSCHLÄGER 2010, S. 4).

Zur Zeit beinhaltet die Ontologie 18 Instanzen im Bereich *Freizeitangebote* und 10 Instanzen im Bereich *sportliche Aktivitäten* (vgl. WOLLSCHLÄGER 2010, S. 2). Da zur Klärung der Funktionalität des Datenbankmodells nicht so viele Instanzen nötig sind, werden aus dem Bereich der Freizeitangebote 10 Instanzen und aus dem Sportbereich 4 Instanzen in die Datenbank eingetragen.

6 Erstellung des Datenbankmodells

Im folgenden Abschnitt wird das erstellte Datenbankmodell erläutert. Es werden die Tabellen und deren Beziehungen untereinander für die Klassen und Klassenbeziehungen einer Ontologie, für Rollen und deren Beziehungen, Einschränkungen und Eigenschaften, sowie für Individuen und deren Beziehungen erläutert. Außerdem wird auf das Problem eingegangen, logische Konstruktoren für Klassen in das Modell zu integrieren.

Für die Erstellung des Datenbankmodells sind folgende Überlegungen wichtig und werden deshalb an dieser Stelle erwähnt:

Ressourcen werden in RDF(S) und somit auch in OWL mit eindeutigen Bezeichnern, den sogenannten URIs versehen (vgl. HITZLER 2008, S. 37 f.). Auch eine Ontologie wird mit einer URI und einem Standardnamensraum ausgestattet, wodurch alle Elemente innerhalb dieses Namensraumes einen eindeutigen Bezeichner erhalten (vgl. HITZLER 2008, S. 128). Das heißt, dass die Namen von Klassen, Rollen und Individuen innerhalb der Ontologie eindeutig sind und kein zweites Mal vergeben werden können.

Auch die Primärschlüssel einer Relation in einer Datenbank müssen aufgrund der Entitätsintegrität unterscheidbar sein und dürfen in einer Relation nicht zweimal den selben Wert besitzen. Aufgrund dieser Tatsache werden in dem Datenbankmodell im überwiegenden Teil der Relationen die Namen der Ontologie-Elemente als Primärschlüssel verwendet. Das hat den Vorteil, dass die Datenbank leichter abgefragt werden kann. Tabellen können größtenteils direkt abgefragt werden, wodurch Join-Abfragen, mit denen mehrere Tabellen miteinander verknüpft werden, weitgehend unnötig sind.

Bei diesem Verfahren kann es aber vorkommen, dass nur jeweils eine Ontologie in einer Datenbank abgelegt werden kann. Verschiedene Ontologien verwenden unterschiedliche Namensräume, wodurch es dazu kommt, dass Elemente von verschiedenen Ontologien auch den gleichen Bezeichner für unterschiedliche Konzepte haben können. Da aber aufgrund der Entitätsintegrität keine zwei Primärschlüssel einer Relation den gleichen Wert besitzen dürfen, ist es nicht möglich, mehrere Ontologien mit gleichen Bezeich-

nern für verschiedene Begriffe in einer Datenbank abzubilden, die nach dem hier behandelten Datenbankmodell erstellt wurde. Das bedeutet auch, dass es nicht möglich ist, solche Ontologien innerhalb einer Datenbank zusammenzufügen. Dafür bräuchte es ein Datenbankmodell, in dem sowohl die einzelnen Ontologien, als auch die einzelnen Elemente der Ontologien mit einer Nummer als Primärschlüssel ausgestattet wären, die sie unterscheidbar machen würde.

Außerdem wurde beim Erstellen des Datenbankmodells versucht, die Funktionalität von OWL DL umzusetzen, da OWL Lite in der Praxis nur von eingeschränkter Bedeutung ist und OWL Full von aktuellen Inferenzmaschinen nur bedingt und meist überhaupt nicht unterstützt wird (vgl. HITZLER 2008, S. 152 f.). Bei der Umsetzung wurden die Einschränkungen von OWL DL beachtet.

6.1 Klassen und einfache Klassenbeziehungen

Für die Klassen einer Ontologie und deren Eigenschaften wurden die Tabellen *klassen*, *k_beziehungen* und *oberklassen* erstellt. Attribute der Tabelle *klassen* sind *K_Name*, *K_Oberklasse*, *K_Abgeschlossen*, *K_Kommentar* und *K_URI*.

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
K_Name	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
K_Oberklasse	VARCHAR(75)	☐	☐	☐	☐	☐	☐	☐	
K_Abgeschlossen	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
K_Kommentar	TEXT	☐	☐	☐	☐	☐	☐	☐	
K_URI	VARCHAR(150)	☐	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.1: Tabelle *klassen*

Das Attribut *K_Name* steht für den Namen einer Klasse und ist der Primärschlüssel der Tabelle. In die Spalte *K_Oberklasse* wird eine eventuell vor-

handene Oberklasse einer Klasse eingetragen. Dabei ist *K_Oberklasse* ein Fremdschlüssel, der die Tabelle *oberklassen* referenziert. Die Tabelle *oberklassen* hat nur ein Attribut, nämlich den Primärschlüssel *OK_Name*. Da kein Primärschlüsselattribut einer Tabelle den Wert NULL annehmen darf, ist für jeden Primärschlüssel des Datenbankmodells das Feld *Not Null (NN)* in der jeweiligen Tabelle markiert.

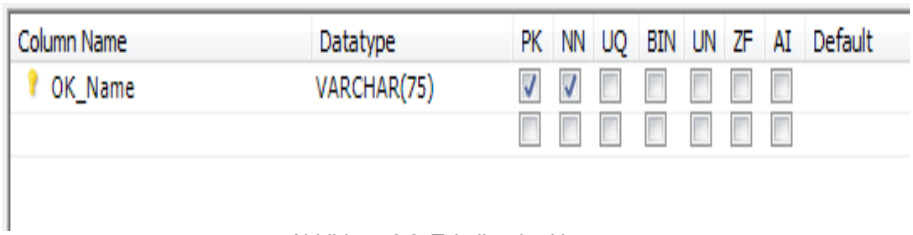


Abbildung 6.2: Tabelle *oberklassen*

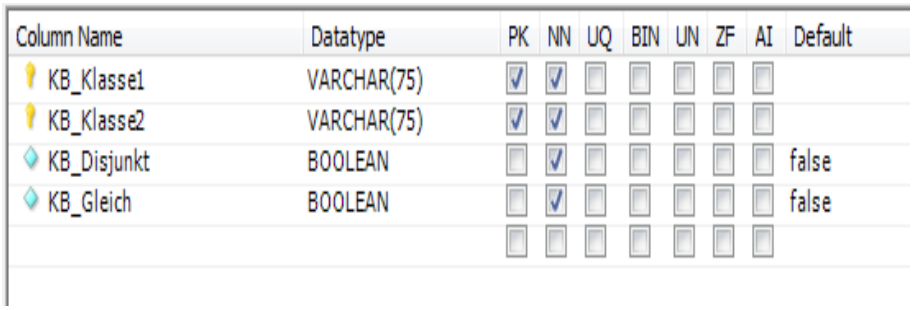
Die Tabellen *oberklassen* und *klassen* stehen in einer 1:N-Beziehung, da eine Klasse gleichzeitig Oberklasse von mehreren anderen Klassen sein kann. Das Auslagern des Namens der Oberklassen in eine eigene Tabelle dient dem Vermeiden von Lösch- und Änderungsanomalien, die in diesem Fall auftreten können, wenn man die in der Datenbank enthaltene Ontologie in ihrer Hierarchiestruktur ändert, bzw. die Namen von Klassen, die gleichzeitig Oberklassen sind, ändert oder löscht. Das spielt vor allen Dingen bei größeren Ontologien eine Rolle. Bei kleineren Ontologien wie *Erlebnis Hamburg* könnte man eventuell auch auf eine Auslagerung verzichten, man müsste aber bei einer Änderung von Klassennamen mehrere Einträge in der Datenbank ändern.

Die Attribute *K_Kommentar* und *K_URI* stehen für Kommentare, die man zu bestimmten Klassen festhalten will, und für die einheitlichen Bezeichner der Klassen. Für *K_Name*, *K_Oberklasse* und *K_URI* wurde als Datentyp eine Zeichenkette mit der Länge von 75 oder 150 Zeichen definiert. *K_Kommentar* hat den Datentyp *TEXT*, mit dem auch längere Zeichenketten gespeichert werden können (vgl. KANNENGIESSER 2007, S. 629).

Das Attribut *K_Abgeschlossen* wurde in die Tabelle aufgenommen, damit man festlegen kann, ob es sich bei einer Klasse um eine abgeschlossene

Klasse mit einer bestimmten Anzahl von Individuen handelt. Diese Funktion wird in OWL mit dem Sprachelement *owl:oneOf* verwirklicht (vgl. HITZLER 2008, S. 136). *K_Abgeschlossen* hat dabei den Datentyp *BOOLEAN*, der standardmäßig auf *FALSE* bzw. *0* gesetzt wird. Möchte man eine abgeschlossene Klasse definieren, muss man für dieses Attribut den Wert *TRUE* bzw. *1* eintragen.

Das Festlegen des Standardwertes auf FALSE für Attribute mit dem Datentypen *BOOLEAN*, sowie das Markieren des Feldes Not Null (NN) für diese Attribute wird auf diese Weise im gesamten Datenbankmodell praktiziert.



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
KB_Klasse1	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
KB_Klasse2	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
KB_Disjunkt	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
KB_Gleich	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.3: Tabelle *k_beziehungen*

In der Tabelle *k_beziehungen* können jeweils zwei Klassen als disjunkt oder als gleich deklariert werden. Den Primärschlüssel bildet dabei die Kombination aus den Namen der Klassen, die man auf diese Weise in Beziehung setzen will. Gleichzeitig ist dieser Primärschlüssel (*KB_Klasse1* und *KB_Klasse2*) auch Fremdschlüssel, der die Werte des Primärschlüssels aus der Tabelle *klassen* (*K_Name*) referenzieren.

Die Attribute *KB_Disjunkt* und *KB_Gleich* sind vom Datentyp *BOOLEAN* und müssen jeweils auf *TRUE* gesetzt werden, wenn man zwei Klassen als disjunkt oder als gleich deklarieren will. Da diese Klassen aber nicht gleichzeitig disjunkt und gleich sein können, dürfen in einem Tupel nicht beide Attribute den Wert *TRUE* annehmen. Da das jedoch technisch gesehen in dieser Tabelle machbar ist, muss diese Möglichkeit mit Hilfe von PHP bei einem späteren Füllen oder Ändern der Tabelle ausgeschlossen werden. Bei einem Bearbeiten der Tabelle per Hand muss ebenfalls auf diese Ein-

schränkung geachtet werden.

Mit den Tabellen *klassen*, *oberklassen* und *k_beziehungen* werden folgende Sprachelemente aus OWL funktionell abgedeckt: *owl:Class* für die Deklaration von Klassen; *rdfs:subClassOf*, *owl:disjointWith*, *owl:equivalentClass* für die Definition von Klassenbeziehungen; *owl:oneOf* für abgeschlossene Klassen und *rdfs:comment* für Kommentare.

6.2 Rollen, Rollenbeziehungen und Rolleneigenschaften

In OWL gibt es zwei Arten von Rollen: abstrakte Rollen (owl:ObjectProperty) und konkrete Rollen (owl:DatatypeProperty). Abstrakte Rollen verbinden Individuen mit Individuen, während konkrete Rollen Individuen mit Datenwerten verbinden, also mit Elementen von Datentypen (vgl. HITZLER 2008, S. 130).

In OWL DL gibt es Einschränkungen, was die Verwendung von konkreten Rollen betrifft. So darf einer konkreten Rolle keine inverse Rolle zugewiesen werden. Auch die Rolleneigenschaften der Transitivität, der Symmetrie und der inversen Funktionalität dürfen nicht mit konkreten Rollen verwendet werden (vgl. HITZLER 2008, S. 153).

Aus diesen Gründen wurde im Datenbankmodell jeweils eine Tabelle für abstrakte und eine für konkrete Rollen erstellt. Dadurch sollen unnötige *NULL*-Werte vermieden werden, zu denen es kommen würde, wenn man beide Arten von Rollen in einer Tabelle zusammenfassen würde. Diese Trennung von abstrakten und konkreten Rollen entspricht auch der Einschränkung der Typentrennung und -deklaration von OWL DL (vgl. HITZLER 2008, S. 153).

In die Tabelle *abstrakte_rollen* lassen sich der Rollenname einer abstrakten Rolle als Primärschlüssel, der eindeutige Bezeichner und ein Kommentar eintragen. Die zugehörigen Datentypen wurden hier wie in der Tabelle *klassen* definiert. Zusätzlich lassen sich eventuell vorhandene Oberrollen sowie äquivalente und inverse Rollen definieren. Zwei Rollen, die invers zueinan-

der sind, stellen dieselbe Relation mit vertauschen Argumenten dar (vgl. HITZLER 2008, S. 147).

Die Attribute *AR_Oberrolle* und *AR_Gleich* sind Fremdschlüssel, die auf den Primärschlüssel der Tabelle *og_rollen* verweisen. Das heißt, dass, wie in der Tabelle *klassen*, die Oberrollen (und hier auch die äquivalenten Rollen) in eine andere Tabelle ausgelagert werden, zu der eine 1:N-Beziehung besteht. Eine Rolle aus der Tabelle *og_rollen* kann eine Oberrolle oder äquivalente Rolle von einer oder mehreren Rollen aus der Tabelle *abstrakte_rollen* sein. Die Auslagerung dient wieder der Vermeidung von Lösch- und Änderungsanomalien. Diese könnten auftreten, wenn man dieselben Oberrollen oder äquivalente Rollen mehrfach in die Tabelle *abstrakte_rollen* eintragen würde. Man müsste bei einer Änderung dieser Informationen Daten an mehreren Stellen ändern.

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
AR_Name	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
AR_Oberrolle	VARCHAR(75)	☐	☐	☐	☐	☐	☐	☐	
AR_Gleich	VARCHAR(75)	☐	☐	☐	☐	☐	☐	☐	
AR_Inverse	VARCHAR(75)	☐	☐	☐	☐	☐	☐	☐	
AR_Kommentar	TEXT	☐	☐	☐	☐	☐	☐	☐	
AR_URI	VARCHAR(150)	☐	☐	☐	☐	☐	☐	☐	
AR_Funktional	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
AR_InversFunktional	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
AR_Symmetrisch	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
AR_Transitiv	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.4: Tabelle *abstrakte_rollen*

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
OGR_Name	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.5: Tabelle *og_rollen*

In der Tabelle *abstrakte_rollen* ist es der Übersichtlichkeit halber vorgesehen, nur eine Rolle als äquivalente Rolle einer anderen definieren zu kön-

nen. Würde man, wie in OWL möglich, mehrere Rollen als äquivalent zu einer anderen bestimmen wollen, müsste man die Namen der äquivalenten Rollen in eine Tabelle auslagern und diese in eine N:M-Beziehung zu der Tabelle *abstrakte_rollen* setzen.

Zuletzt lässt sich in der Tabelle anhand von logischen Datentypen bestimmen, ob eine abstrakte Rolle funktional, invers funktional, transitiv oder symmetrisch ist. Diese Rolleneigenschaften haben folgende Bedeutung: Steht ein Individuum A zu einem Individuum B in einer symmetrischen Rollenbeziehung, dann steht auch B zu A in Beziehung. Transitivität bedeutet, dass, wenn A zu B in Beziehung steht und B zu C, auch A zu C in Beziehung steht. Funktionalität einer Rolle bedeutet, dass B und C identisch sind, wenn A zu B in Beziehung steht und A auch zu C. Dabei sind transitive Rollen sinnvollerweise nie funktional. Die inverse Funktionalität einer Rolle R ist gleichbedeutend dazu, dass die Rolle, die zu R invers ist, funktional ist (vgl. HITZLER 2008, S. 149 ff.).

Die Tabelle, die für die konkreten Rollen erstellt wurde, ist der für abstrakte Rollen ähnlich. Beide Tabellen unterscheiden sich in zwei Punkten: konkreten Rollen lassen sich in OWL DL einige Rolleneigenschaften nicht zuweisen. Die Attribute für inverse Rollen, Symmetrie, Transitivität und inverse Funktionalität sind in der Tabelle *konkrete_rollen* nicht vorhanden.

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
KR_Name	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
KR_Oberrolle	VARCHAR(75)	☐	☐	☐	☐	☐	☐	☐	
KR_Gleich	VARCHAR(75)	☐	☐	☐	☐	☐	☐	☐	
KR_Kommentar	TEXT	☐	☐	☐	☐	☐	☐	☐	
KR_URI	VARCHAR(150)	☐	☐	☐	☐	☐	☐	☐	
KR_Funktional	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
KR_Range	INT	☐	☑	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.6: Tabelle *konkrete_rollen*

Die Attribute *KR_Oberrolle* und *KR_Gleich* bezeichnen aber auch in dieser Tabelle eventuell vorhandene Oberrollen oder äquivalente Rollen. Auch hier sind es Fremdschlüssel, die eine 1:N-Beziehung mit der Tabelle *og_rollen* herstellen. Das bedeutet, dass die Tabelle *og_rollen* sowohl die Namen der Oberrollen und äquivalenten Rollen von abstrakten als auch von konkreten Rollen beinhaltet.

Der andere Punkt, in dem sich die Tabellen *abstrakte_rollen* und *konkrete_rollen* unterscheiden, ist das Attribut *KR_Range*. Dieses Attribut bezeichnet den Wertebereich einer konkreten Rolle, der immer von einem bestimmten Datentyp ist. Dabei ist *KR_Range* ein Fremdschlüssel des Datentyps *INT*, der auf den Primärschlüssel der Tabelle *datentypen* verweist. Zwischen beiden Tabelle besteht eine 1:N-Beziehung, da für mehrere Rollen der gleiche Datentyp als Wertebereich definiert sein kann.

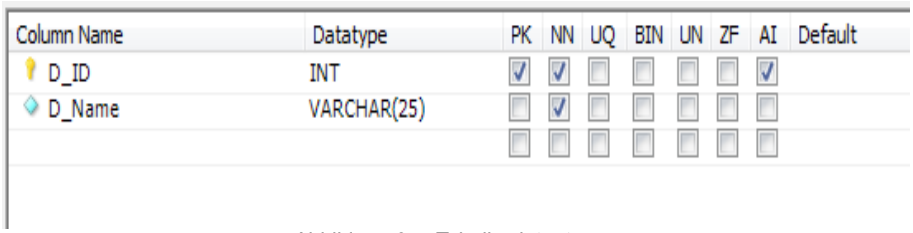
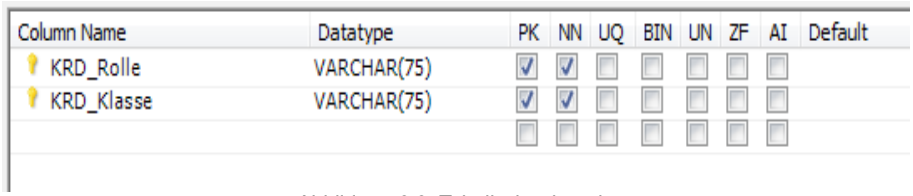


Abbildung 6.7: Tabelle *datentypen*

Die Tabelle *datentypen* besteht aus dem Primärschlüssel *D_ID*, der in diesem Fall eine Zahl des Datentyps *INT* ist, und aus dem Attribut *D_Name*, das den Namen des Datentypen bezeichnet.

Der Wertebereich, der in OWL durch *rdfs:range* definiert wird, bestimmt die Objekte, die gemeinsam mit einem Prädikat im Tripel vorkommen. Demgegenüber werden mit *rdfs:domain* den Subjekten im Tripel Typen zugewiesen, wodurch der Definitionsbereich einer Rolle bestimmt wird. Dabei kann eine Rolle eine oder mehrere Klassen als Definitionsbereich erhalten, aus denen sie Individuen als Subjekte verwenden darf. Da abstrakte Rollen Individuen mit Individuen verbinden, besteht auch deren Wertebereich aus einer oder mehreren Klassen (vgl. HITZLER 2008, S. 76 f.).

Der Definitionsbereich von Rollen kann als Kombination der jeweiligen Rollen mit verschiedenen Klassen dargestellt werden. In dem Datenbankmodell wird diese Kombination als N:M-Beziehung der Tabellen *klassen* und *abstrakte_rollen* bzw. *konkrete_rollen* definiert. Der Definitionsbereich von konkreten Rollen wird dabei in der Zwischentabelle *kr_domain* eingetragen. Diese enthält die Kombinationen der Namen von Rollen und von Klassen, die den Definitionsbereich der konkreten Rollen darstellen.

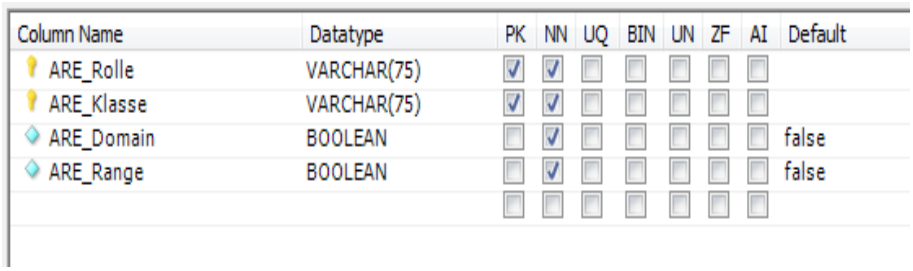


Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
KRD_Rolle	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
KRD_Klasse	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.8: Tabelle *kr_domain*

Diese Kombination aus *KRD*_Rolle und *KRD_Klasse* bildet den Primärschlüssel der Tabelle. Gleichzeitig sind beide Attribute Fremdschlüssel, die die Primärschlüssel der Tabellen *klassen* und *konkrete_rollen* referenzieren.

Da bei abstrakten Rollen auch der Wertebereich aus einer oder mehreren Klassen besteht und dieser auch in einer N:M-Beziehung festgehalten werden kann, wird sowohl der Definitionsbereich als auch der Wertebereich von abstrakten Rollen in einer Tabelle eingetragen. Die Tabelle *ar_eigenschaften* ist eine Zwischentabelle, deren Fremdschlüssel die Primärschlüssel der Tabellen *klassen* und *abstrakte_rollen* referenzieren.



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
ARE_Rolle	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
ARE_Klasse	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
ARE_Domain	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
ARE_Range	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.9: Tabelle *ar_eigenschaften*

In dieser Tabelle lassen sich verschiedene Kombinationen von Rollen- und Klassennamen festhalten. Ob diese Kombinationen einen Wertebereich oder einen Definitionsbereich darstellen, muss mit den beiden Attributen *ARE_Domain* und *ARE_Range* bestimmt werden, indem eines dieser Attribute den Wert *TRUE*, bzw. *1* zugewiesen bekommt. Auch hier ist darauf zu achten, dass beim Eintragen in die Datenbank nicht beide Attribute den selben Wert erhalten.

In den in diesem Kapitel genannten Tabellen werden unter anderem folgende Sprachelemente abgedeckt: *owl:ObjectProperty* und *owl:DatatypeProperty* zur Deklaration von Rollen; r*dfs:subPropertyOf*, *owl:equivalentProperty* und *owl:inverseOf* für Rollenbeziehungen; *rdfs:domain* und *rdfs:range* für die Definition von Rolleneigenschaften. Nicht genannt wurden in dieser Aufzählung die Sprachelemente, die die Rolleneigenschaften der Symmetrie, Transitivität, Funktionalität und der inversen Funktionalität deklarieren.

6.3 Rolleneinschränkungen

In OWL wird zwischen zwei Arten von Rolleneinschränkungen unterschieden: Einschränkungen, die die Kardinalität einer Rolle betreffen, und Einschränkungen des Wertebereichs einer Rolle. Rolleneinschränkungen werden innerhalb des Sprachelements *owl:Restriction* deklariert, das als Unterklasse von *owl:Class* definiert ist. Der Unterschied zwischen Einschränkungen des Wertebereiches einer Rolle und *rdfs:range* besteht darin, dass *rdfs:range* für jede Situation gilt, in der die Rolle verwendet wird, wohingegen die Werteinschränkung nur für eine Kombination aus einer Rolle mit einer bestimmten, deklarierten Klasse Bestand hat (vgl. OWL-REFERENZ 2004).

Folgende Sprachelemente werden in OWL für Rolleneinschränkungen verwendet: *owl:allValuesFrom*, *owl:someValuesFrom* und *owl:hasValue* für Einschränkungen des Wertebereiches; *owl:maxCardinality*, *owl:minCardinality* und *owl:cardinality* für Einschränkungen der Kardinalität einer Rolle, wobei Kardinalität die Anzahl der Werte bezeichnet. Eine genaue Beschreibung der Funktionalität dieser Elemente in OWL findet sich auf den Inter-

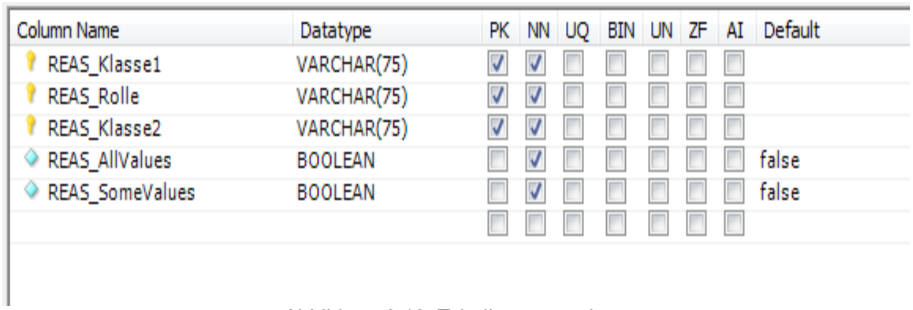
netseiten des W3C (vgl. OWL-REFERENZ 2004).

Rolleneinschränkungen können sowohl auf abstrakte als auch auf konkrete Rollen angewendet werden. Einschränkungen von Wertebereichen beziehen sich also auf Klassen und Individuen bzw. auf Wertebereiche oder Werte von Datentypen (vgl. OWL-REFERENZ 2004).

In dem erstellten Datenbankmodell wird nur auf Rolleneinschränkungen von abstrakten Rollen eingegangen. Das liegt zum einen daran, dass das Modell übersichtlich gehalten werden sollte, zum anderen gibt es in den beiden verwendeten Ontologien keine Beispiele für Einschränkungen von konkreten Rollen. Für diesen Zweck können aber nachträglich Tabellen in das Modell eingefügt werden, die vom Prinzip her wie die im Folgenden beschriebenen aufgebaut sind. Im Datenbankmodell sind für Rolleneinschränkungen die drei Tabellen *re_asvalues*, *re_hvalues* und *re_kardinalitaet* erstellt worden.

In die Tabelle *re_asvalues* werden die Einschränkungen eingetragen, die in OWL mit den Sprachelementen *owl:allValuesFrom* und *owl:someValuesFrom* deklariert werden. Die drei Bestandteile von solchen Einschränkungen sind eine Rolle, eine Klasse, für die diese Rolle beschränkt wird, und eine weiteren Klasse, die den eingeschränkten Wertebereich der Rolle darstellt. Geht man davon aus, dass eine Rolle für eine Klasse auch mehrfach beschränkt werden kann, so gibt es unterschiedliche Kombinationen dieser drei Bestandteile. Diese Kombinationen sind jedoch immer eindeutig, da eine bestimmte Einschränkung in OWL nicht mehrfach deklariert wird.

Daher werden die drei genannten Bestandteile als Attribute *REAS_Klasse1*, *REAS_Rolle* und *REAS_Klasse2* der Tabelle *re_asvalues* erstellt und als Primärschlüssel definiert. Gleichzeitig sind sie Fremdschlüssel, die auf die Primärschlüssel der Tabellen *klassen* und *abstrakte_rollen* verweisen.

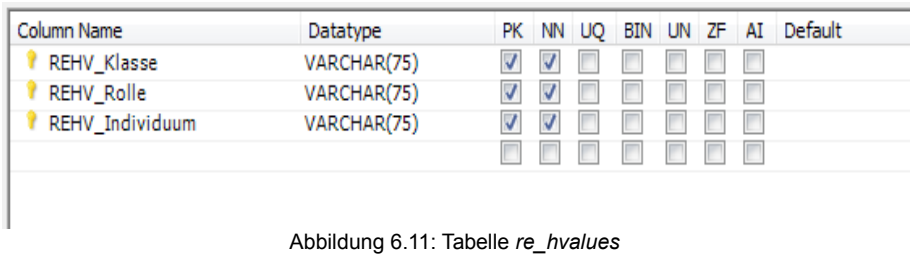


Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
REAS_Klasse1	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
REAS_Rolle	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
REAS_Klasse2	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
REAS_AllValues	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
REAS_SomeValues	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.10: Tabelle *re_asvalues*

Neben den drei Elementen des Primärschlüssels muss der Typ der Einschränkung definiert werden. Dazu werden die Attribute *REAS_AllValues* und *REAS_SomeValues* erstellt, für die der Wert *TRUE* oder *1* in die Tabelle eingetragen werden muss. Auch hier ist darauf zu achten, dass nicht beide Attribute den selben Wert erhalten.

In die obige Tabelle wird der Wertebereich einer Rolleneinschränkung als Name einer Klasse eingetragen. Dies entspricht einer Einschränkung von OWL Lite, die besagt, das Wertebereiche in Bezug auf *owl:allValuesFrom* und *owl:someValuesFrom* nur Klassennamen sein können. Auf den Referenzseiten des W3C wird ein Wertebereich aber auch als *Klassenbeschreibung* definiert (vgl. OWL-REFERENZ 2004). Das heißt genaugenommen, der Wertebereich könnte ebenso eine Aufzählung von Individuen sein, die zusammen eine Klasse bilden. Im Datenbankmodell lassen sich auch solche Aufzählungen als Wertebereiche eintragen, nämlich in der Tabelle *re_hvalues.*



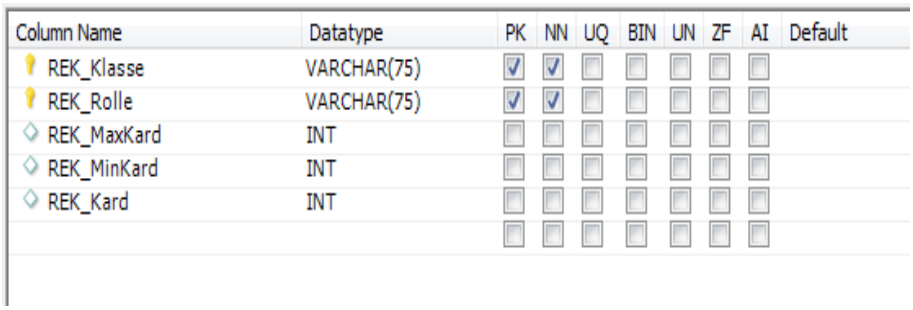
Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
REHV_Klasse	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
REHV_Rolle	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
REHV_Individuum	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.11: Tabelle *re_hvalues*

Im Gegensatz zu den oben genannten Rolleneinschränkungen hat die Einschränkung, die durch *owl:hasValue* ausgedrückt wird, folgende drei Be-

standteile: eine Rolle, eine Klasse, für die die Rolle eingeschränkt wird, und ein Individuum als Wertebereich. Diese Einschränkung kann nicht in die Tabelle *re_asvalues* aufgenommen werden, da aus ihren Bestandteilen ein anderer Primärschlüssel gebildet wird, einer mit einem Individuum als drittem Primärschlüsselattribut.

Es wird also eine Tabelle *re_hvalues* mit den Primärschlüsselattributen *REHV_Klasse*, *REHV_Rolle* und *REHV_Individuum* erstellt. Mit diesem zusammengesetzten Primärschlüssel werden die drei Bestandteile der Rolleneinschränkung mit *owl:hasValue* beschrieben. Wie schon erwähnt lassen sich durch diesen Primärschlüssel auch Aufzählungen von Individuen als Wertebereich einer Rolle definieren. Diese Funktion ist äquivalent zum OWL-Sprachelement *owl:someValuesFrom* in Verbindung mit *owl:oneOf* (vgl. HITZLER 2008, S. 143).



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
REK_Klasse	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
REK_Rolle	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
REK_MaxKard	INT	☐	☐	☐	☐	☐	☐	☐	
REK_MinKard	INT	☐	☐	☐	☐	☐	☐	☐	
REK_Kard	INT	☐	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.12: Tabelle *re_kardinalitaet*

Die dritte Tabelle für Rolleneinschränkungen ist die Tabelle *re_kardinalitaet*. Sie beinhaltet die Attribute *REK_MaxKard*, *REK_MinKard* und *REK_Kard*, die den OWL-Sprachelementen *owl:maxCardinality*, *owl:minCardinality* und *owl:cardinality* entsprechen. Sie sind von einem Primärschlüssel abhängig, der aus den beiden Attributen *REK_Klasse* und *REK_Rolle* zusammengesetzt ist.

Die Nichtschlüsselattribute dieser Tabelle können nicht in eine der beiden anderen Tabellen für Rolleneinschränkungen integriert werden, da sie in diesen nur von einem Teil der Primärschlüsselattribute abhängig wären. Somit wären die Tabellen nicht in der zweiten Normalform der Normalisierung.

Ferner könnte man die Tabelle nicht mit einer ID als Primärschlüssel und allen anderen Attributen als Nichtschlüsselattribut erstellen, da so eine transitive Abhängigkeit der Nichtschlüsselattribute *REK_MaxKard*, *REK_MinKard* und *REK_Kard* von den Nichtschlüsselattributen *REK_Klasse* und *REK_Rolle* entstehen würde. Somit wäre die Tabelle nicht in der dritten Normalform.

Die Schlüsselattribute *REK_Klassen* und *REK_Rollen* sind wiederum auch Fremdschlüssel, die die Primärschlüssel der Tabellen *klassen* und *abstrakte_rollen* referenzieren und diese Tabellen in eine N:M-Beziehung bringen. Für die drei Nichtschlüsselattribute lassen sich Zahlenwerte entsprechend der Kardinalitäten der Rolleneinschränkungen eintragen.

Beim Eintragen der Werte muss aber auf eine Einschränkung von OWL DL geachtet werden, die abstrakte Rollen betrifft: „Zahlenrestriktionen mittels owl:cardinality, owl:minCardinality oder owl:maxCardinality dürfen nicht mit transitiven Rollen, deren Inversen oder deren Oberrollen verwendet werden“ (HITZLER 2008, S. 153). Daher muss versucht werden, diese Einschränkung mittels Programmierung in PHP zu berücksichtigen.

6.4 Individuen und Beziehungen zwischen Individuen

Die Individuen einer Ontologie werden in die Tabelle *individuen* eingetragen. Diese Tabelle hat die Attribute *I_Name*, *I_Kommentar* und *I_URI*, in die der Name eines Individuums, dessen eindeutiger Bezeichner und ein möglicher Kommentar eingetragen werden. Als Datentypen sind, wie auch für die anderen Elemente von Ontologien, *VARCHAR* mit einer Länge von 75 bis 150 Zeichen und *TEXT* für den Kommentar definiert.

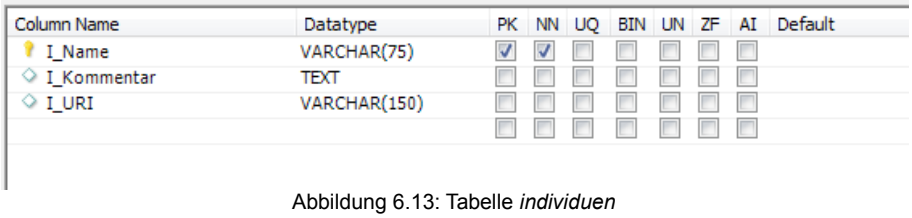
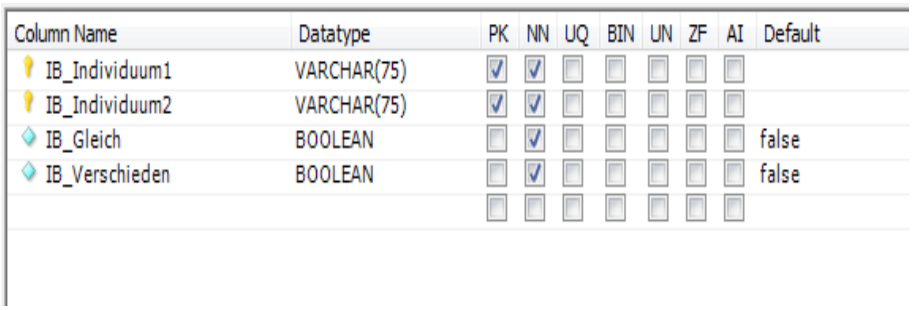


Abbildung 6.13: Tabelle *individuen*

Um in OWL Beziehungen zwischen Individuen zu definieren, gibt es die Sprachelemente *owl:sameAs* und *owl:differentFrom*. Mit *owl:sameAs* lässt sich darstellen, dass ein Individuum verschiedene Bezeichner hat, während man mit *owl:differentFrom* zwei Individuen als verschieden deklarieren kann (vgl. HITZLER 2008, S. 133 ff.).

In dem erstellten Datenbankmodell werden diese Beziehungen zwischen Individuen in die Tabelle *i_beziehungen* eingetragen. Diese ist ähnlich aufgebaut, wie die Tabelle *k_beziehungen*, die einfache Klassenbeziehungen enthält. Der zusammengesetzte Primärschlüssel besteht aus den beiden Schlüsselattributen *IB_Individuum1* und *IB_Individuum2*, die als Fremdschlüssel die Primärschlüssel der Tabelle *individuen* referenzieren. Für jede Kombination aus zwei Individuen wird in die Tabelle eingetragen, ob es sich um gleiche oder verschiedene Individuen handelt. Dazu muss für eines der beiden Attribute *IB_Gleich* oder *IB_Verschieden*, die den Datentyp *BOOLEAN* haben, der Wert *TRUE* oder *1* eingetragen werden, wobei darauf geachtet werden muss, dass nicht beide Attribute gleichzeitig diesen Wert erhalten können.

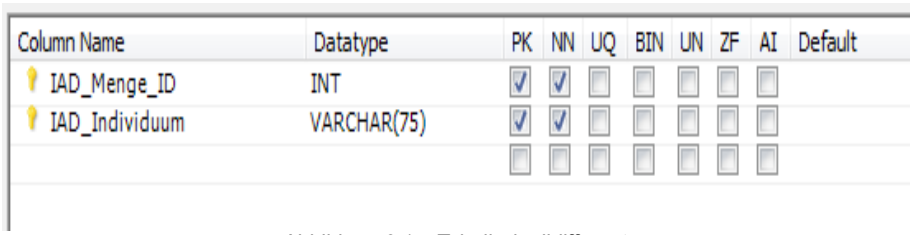


Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
IB_Individuum1	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
IB_Individuum2	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
IB_Gleich	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
IB_Verschieden	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	false
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.14: Tabelle *i_beziehungen*

In OWL gibt es außerdem ein Sprachkonstrukt, mit dem man die Verschiedenheit einer Menge von Individuen beschreiben kann. Hierfür verwendet man die Sprachelemente *owl:AllDifferent* und *owl:distinctMembers* (vgl. HITZLER 2008, S. 135). Diese Beziehung wird im erstellten Datenbankmodell in die Tabelle *i_alldifferent* eingetragen. Diese Tabelle enthält nur einen zusammengesetzten Primärschlüssel, der aus den Schlüsselattributen *IAD_Menge_ID* und *IAD_Individuum* besteht. In *IAD_Menge_ID* lässt sich

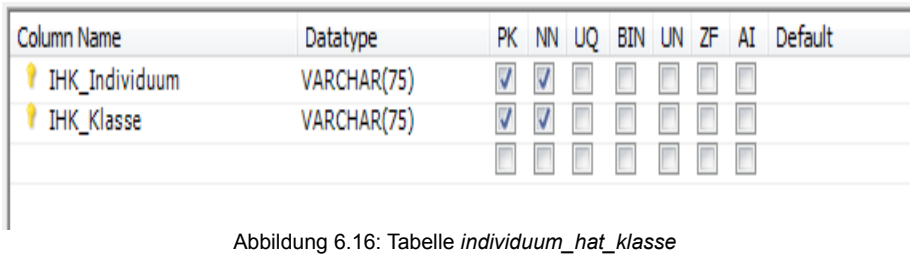
die Nummer der Menge eintragen, die verschiedene Individuen enthält. In *IAD_Individuum* werden die Namen der Individuen der entsprechenden Menge eingetragen. Dieses Schlüsselattribut ist gleichzeitig ein Fremdschlüssel, der den Primärschlüssel der Tabelle *individuen* referenziert. So lassen sich in die Tabelle *i_alldifferent* mehrere Mengen unterschiedlicher Individuen eintragen, die als verschieden deklariert werden sollen.



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
IAD_Menge_ID	INT	☑	☑	☐	☐	☐	☐	☐	
IAD_Individuum	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.15: Tabelle *i_alldifferent*

Die Klassenzugehörigkeit von Individuen wird in der Tabelle *individuum_hat_klasse* deklariert. Diese Zwischentabelle bringt die Tabellen *individuen* und *klassen* in eine N:M-Beziehung.



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
IHK_Individuum	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
IHK_Klasse	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.16: Tabelle *individuum_hat_klasse*

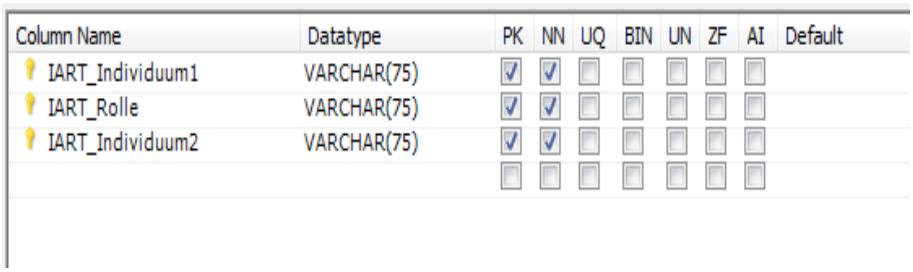
Durch den zusammengesetzten Primärschlüssel können verschiedene Kombinationen von Klassen und Individuen eingetragen werden. Dabei kann ein Individuum zu mehreren Klassen gehören sowie eine Klasse mehrere Individuen besitzen.

6.5 Rollenbeziehungen von Individuen

Die Verknüpfungen von zwei Individuen durch eine Rolle werden in den

beiden Tabellen *iar_tripel* und *ikr_tripel* gespeichert. Da bei diesen Verknüpfungen das Objekt abstrakter Rollen ein Individuum und konkreter Rollen ein Datenwert ist, müssen zwei verschiedene Tabellen erstellt werden.

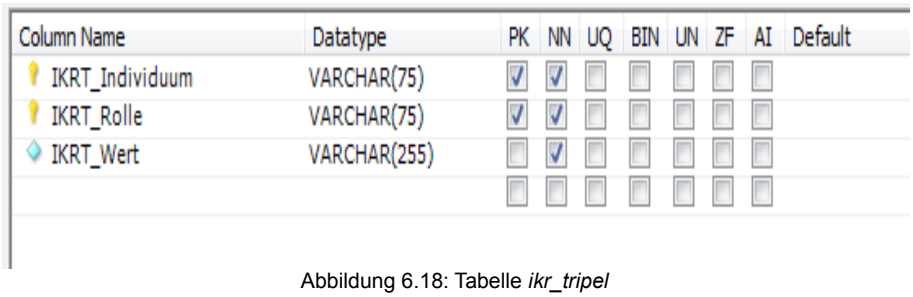
Die Tabelle *iar_tripel* ist für abstrakte Rollen gedacht und besteht aus einem zusammengesetzten Primärschlüssel, der aus den drei Attributen *IART_Individuum1*, *IART_Rolle* und *IART_Individuum2* besteht. Diese Schlüsselattribute sind gleichzeitig Fremdschlüssel, die die Primärschlüssel der Tabellen *individuen* und *rollen* referenzieren. Durch den aus drei Attributen zusammengesetzten Primärschlüssel lassen sich alle möglichen Verknüpfungen von Individuen durch eine abstrakte Rolle eindeutig beschreiben.



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
IART_Individuum1	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
IART_Rolle	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
IART_Individuum2	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.17: Tabelle *iar_tripel*

Die Beziehungen zwischen Individuen und konkreten Rollen beschreibt die Tabelle *ikr_tripel*.



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
IKRT_Individuum	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
IKRT_Rolle	VARCHAR(75)	☑	☑	☐	☐	☐	☐	☐	
IKRT_Wert	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

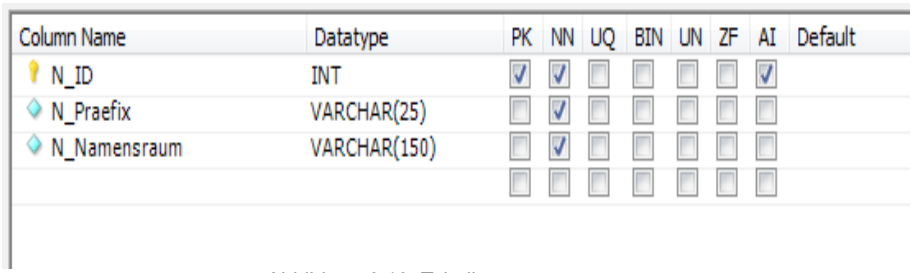
Abbildung 6.18: Tabelle *ikr_tripel*

Hier besteht der Primärschlüssel aus den zwei Schlüsselattributen *IKRT_Individuum* und *IKRT_Rolle*, die wiederum als Fremdschlüssel die

Primärschlüssel der Tabellen *individuen* und *rollen* referenzieren. Das dritte Attribut ist *IKRT_Wert*, das den Datentyp *VARCHAR(255)* hat, also eine Zeichenkette mit der maximalen Länge von 255 Zeichen ist. Dieses Attribut steht für den Datenwert, der durch eine konkrete Rolle mit einem Individuum verbunden wird. Durch die maximale Länge von 255 Zeichen ist zu beachten, dass dieser Datenwert nicht längere Texte enthalten kann. Sollte man längere Texte als Datenwert speichern wollen, müsste man für *IKRT_Wert* den Datentyp *TEXT* definieren. Außerdem ist in dieser Tabelle zu beachten, dass sich durch die Eindeutigkeit der Primärschlüssel nur ein Datenwert pro Kombination aus Individuum und konkreter Rolle zuweisen lässt. Dies ist anders als in *iar_tripel*, wo man jede Kombination aus Individuum und abstrakter Rolle mit mehreren Objekten verknüpfen kann.

6.6 Namensräume und Metadaten

Namensräume werden im Kopf eines OWL-Dokumentes spezifiziert (vgl. HITZLER 2008, S. 128). Die in einer Ontologie verwendeten Namensräume können in dem erstellten Datenbankmodell in der Tabelle *namensraeume* gespeichert werden.



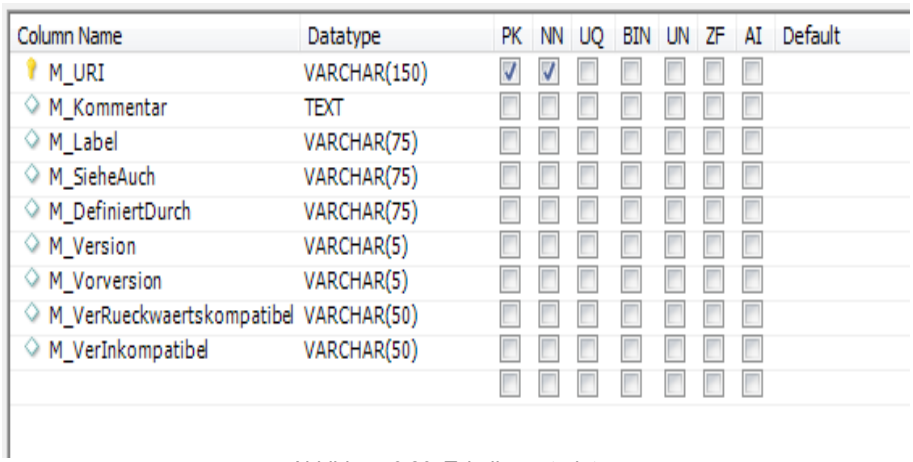
Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
N_ID	INT	☑	☑	☐	☐	☐	☐	☑	
N_Praefix	VARCHAR(25)	☐	☑	☐	☐	☐	☐	☐	
N_Namensraum	VARCHAR(150)	☐	☑	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.19: Tabelle *namensraeume*

Diese besteht aus dem Primärschlüssel *N_ID* und den Attributen *N_Praefix* und *N_Namensraum*. Es können die Präfixe und die Namen der Namensräume - als URI - eingetragen werden.

Ebenfalls im Kopf eines OWL-Dokumentes findet man allgemeine Informationen, die die gesamte Ontologie betreffen, z. B. Informationen zur Versio-

nierung (vgl. HITZLER 2008, S. 128). Diese Informationen werden neben der URI der Ontologie in der Tabelle *metadaten* abgelegt.



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
M_URI	VARCHAR(150)	☑	☑	☐	☐	☐	☐	☐	
M_Kommentar	TEXT	☐	☐	☐	☐	☐	☐	☐	
M_Label	VARCHAR(75)	☐	☐	☐	☐	☐	☐	☐	
M_SieheAuch	VARCHAR(75)	☐	☐	☐	☐	☐	☐	☐	
M_DefiniertDurch	VARCHAR(75)	☐	☐	☐	☐	☐	☐	☐	
M_Version	VARCHAR(5)	☐	☐	☐	☐	☐	☐	☐	
M_Vorversion	VARCHAR(5)	☐	☐	☐	☐	☐	☐	☐	
M_VerRueckwaertskompatibel	VARCHAR(50)	☐	☐	☐	☐	☐	☐	☐	
M_VerInkompatibel	VARCHAR(50)	☐	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	

Abbildung 6.20: Tabelle *metadaten*

Die Nichtschlüsselattribute dieser Tabelle entsprechen folgenden Sprachelementen aus OWL: *rdfs:comment*, *rdfs:label*, *rdfs:seeAlso*, *rdfs:isDefinedBy*, *owl:versionInfo*, *owl:priorVersion*, *owl:backwardCompatibleWith*, *owl:incompatibleWith*. Die Sprachelemente *owl:DeprecatedClass, owl:DeprecatedProperty* und *owl:imports* wurden nicht berücksichtigt, da sie innerhalb des Datenbankmodells nicht nötig sind. Die beiden ersten Elemente beschreiben Teile der Ontologie, die zwar unterstützt werden, aber nicht mehr verwendet werden sollten. Mit *owl:imports* können andere Ontologien importiert werden (vgl. HITZLER 2008, S. 128). Ein Import von anderen Ontologien ist aber in diesem Datenbankmodell aufgrund der definierten Primärschlüssel nicht ohne weiteres möglich.

6.7 Logische Konstruktoren auf Klassen

Die übrigen Sprachelemente von OWL, auf die an dieser Stelle eingegangen wird, sind die logischen Konstruktoren auf Klassen. „Mit ihrer Hilfe lassen sich atomare Klassen […] zu komplexen Klassen verbinden“ (HITZLER 2008, S. 135). „Zur Verfügung stehen Konjunktion, Disjunktion und Negation, d. h. logisches *und*, *oder* und *nicht*, realisiert durch die OWL-Sprachele-

mente *owl:intersectionOf*, *owl:unionOf* und *owl:complementOf*“ (HITZLER 2008, S. 135). Diese Elemente können beliebig geschachtelt werden (vgl. HITZLER 2008, S. 139).

Im folgenden Beispiel mit *owl:unionOf* und *rdfs:subClassOf* wird ausgesagt, dass jeder Professor mindestens einer der beiden Klassen *aktivLehrend* oder *imRuhestand* angehört (vgl. HITZLER 2008, S. 139).

```
− <rdf:RDF>
  − <owl:Class rdf:about="Professor">
    − <rdfs:subClassOf>
      − <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="aktivLehrend"/>
          <owl:Class rdf:about="imRuhestand"/>
        </owl:unionOf>
      </rdfs:subClassOf>
    </owl:Class>
  </rdf:RDF>
```

Abbildung 6.21: Beispiel mit logischem Operator *owl:unionOf* (HITZLER 2008, S. 139)

Die Verwendung von logischen Konstruktoren wurde nicht in das Datenbankmodell integriert. Das liegt zum einen an der hohen Komplexität, die durch die Schachtelung der Konstruktoren ermöglicht wird. Zum anderen lassen sich logische Konstruktoren an verschiedene OWL-Sprachelemente knüpfen, z. B. an *owl:Class*, *rdfs:subClassOf* und *owl:equivalentClass* (vgl. HITZLER 2008, S. 138 f.). Da diese Elemente im Datenbankmodell über mehrere Tabellen verteilt sind, wurde keine Möglichkeit gesehen, logische Konstruktoren im Modell zu berücksichtigen.

7 Überprüfung des Datenbankmodells

Zur Überprüfung der Funktionalität des Datenbankmodels wurden zwei Datenbanken mit den Ontologien von Eisenmenger (*lookedup*) sowie von Wollschläger und Ingber (*Erlebnis Hamburg*) erstellt. Die Funktionalität wird anhand von beispielhaften Abfragen in beiden Datenbanken geprüft. In der Ontologie *Erlebnis Hamburg* werden sich diese Abfragen sowohl auf die inhaltlichen, als auch auf die strukturellen Aspekte der Ontologien beziehen. Es werden hier fünf Abfragen in Bezug auf den Inhalt und drei strukturelle Abfragen durchgeführt. Der strukturelle Teil der Ontologie von Andre Eisenmenger wird dreimal abgefragt.

7.1 Abfragen in der Ontologie *lookedup*

Mit folgender Abfrage werden alle Individuen der Klasse *Lemma* mit ihren konkreten Rollenbeziehungen alphabetisch geordnet angezeigt. Die Verknüpfung der Tabellen *individuum_hat_klasse* und *ikr_tripel* wird durch einen *LEFT JOIN* Ausdruck erreicht:

SELECT IHK_Individuum AS Individuum, IHK_Klasse AS Klasse, IKRT_Rolle AS Rolle, IKRT_Wert AS Wert

FROM individuum_hat_klasse LEFT JOIN ikr_tripel ON IHK_Individuum = IKRT_Individuum

WHERE IHK_Klasse = 'Lemma'

ORDER BY IHK_Individuum ASC;

Individuum	Klasse	Rolle	Wert
Abonnement	Lemma	hatKurztext	Ein Abonnement ist der regelmaeßige Bezug einer Leistung
Allergie	Lemma	hatKurztext	Allergien sind eine ueberempfindliche Reaktion des Koerpers auf v...
Arbeitsvertrag	Lemma	hatKurztext	Ein Arbeitsvertrag ist ein Vertrag zwischen einem Arbeitgeber und ...
Extremsportart	Lemma	hatKurztext	Unter Extremsportarten versteht man sportliche Aktivitaeten, bei ...
Freemail	Lemma	hatKurztext	Freemail ist die Bezeichnung für kostenlose E-Mails im Internet.
Geldkarte	Lemma	hatKurztext	Eine Geldkarte ist eine Bankkarte, in die zusaetzlich ein Mikro-Chip i...
Plagiat	Lemma	hatKurztext	Der Diebstahl von geistigem Eigentum wird auch als Plagiat bezeich...
Schulden	Lemma	hatKurztext	Schulden hat man, wenn man eine Rechnung, geliehenes Geld ode...
Spurenelement	Lemma	hatKurztext	purenelemente sind chemische Stoffe in Miniaturformat, die für leb...
Zigaretten	Lemma	hatKurztext	Zigaretten sind Genussmittel, die aus den getrockneten und feinge...

Abbildung 7.1: Ergebnis der ersten Abfrage in *lookedup*

Durch die zweite Abfrage lassen sich alle Stichworte zu dem Lemma *Extremsportart* finden:

SELECT IHK_Individuum AS Individuum, IART_Rolle AS Rolle, IART_Individuum2 AS Wert

FROM individuum_hat_klasse LEFT JOIN iar_tripel ON IHK_Individuum = IART_Individuum1

WHERE IHK_Individuum = 'Extremsportart' AND IART_Rolle = 'hatStichwort';

Individuum	Rolle	Wert
Extremsportart	hatStichwort	Adrenalin
Extremsportart	hatStichwort	Bungee
Extremsportart	hatStichwort	Fallschirmspringen
Extremsportart	hatStichwort	Rafting

Abbildung 7.2: Ergebnis der zweiten Abfrage in *lookedup*

Das Ergebnis der dritten Abfrage ist eine Auflistung aller Lemmata der Ontologie mit ihren entsprechenden Kategorien:

SELECT IART_Individuum1 AS Individuum, IART_Rolle AS Rolle, IART_Individuum2 AS Wert

FROM individuum_hat_klasse LEFT JOIN iar_tripel ON IHK_Individuum = IART_Individuum1

WHERE IHK_Klasse = 'Lemma' AND IART_Rolle = 'gehoertZuKategorie'

ORDER BY IART_Individuum1;

	Individuum	Rolle	Wert
▶	Abonnement	gehoertZuKategorie	Computer_und_Internet
	Allergie	gehoertZuKategorie	Koerper_und_Gesundheit
	Arbeitsvertrag	gehoertZuKategorie	Geld_und_Job
	Extremsportart	gehoertZuKategorie	Sport_und_Freizeit
	Freemail	gehoertZuKategorie	Computer_und_Internet
	Geldkarte	gehoertZuKategorie	Geld_und_Job
	Plagiat	gehoertZuKategorie	TV_und_Musik
	Plagiat	gehoertZuKategorie	Verbraucherrecht
	Schulden	gehoertZuKategorie	Geld_und_Job
	Spurenelement	gehoertZuKategorie	Koerper_und_Gesundheit
	Zigaretten	gehoertZuKategorie	Koerper_und_Gesundheit

Abbildung 7.3: Ergebnis der dritten Abfrage in *lookedup*

7.2 Inhaltliche Abfragen in der Ontologie *Erlebnis Hamburg*

Als Beispiele für inhaltliche Abfragen in der Ontologie Erlebnis Hamburg werden die Fragen verwendet, die Wollschläger und Ingber in ihrem Konzept als beispielhafte Fragen erarbeitet haben und die anhand der Ontologie beantwortet werden sollen.

Frage 1: „Welche Möglichkeiten habe ich im Sommer draußen Sport zu machen?“

SELECT IHK_Individuum AS Individuum, IHK_Klasse AS Klasse FROM individuum_hat_klasse

WHERE IHK_Klasse IN ('Radfahren', 'Tretbootfahren', 'Segelbootfahren', 'Paddel-_und_Ruderbootfahren')

AND IHK_Individuum IN (SELECT IART_Individuum1 FROM iar_tripel WHERE IART_Rolle = 'hatJahreszeit' AND IART_Individuum2 ='Sommersaison');

Individuum	Klasse
Alfred_Seebeck	Paddel-_und_Ruderbootfahren
Segelschule_Pieper	Paddel-_und_Ruderbootfahren
Fischmarkt_Breite_Straße	Radfahren
Jungfernstieg_Ballindamm	Radfahren
Alfred_Seebeck	Segelbootfahren
Segelschule_Pieper	Segelbootfahren
Alfred_Seebeck	Tretbootfahren
Segelschule_Pieper	Tretbootfahren

Abbildung 7.4: Ergebnis der ersten inhaltlichen Abfrage in *Erlebnis Hamburg*

In dieser Abfrage müssen zwei Bedingungen erfüllt sein, damit man das richtige Ergebnis erhält. Zum ersten muss sich der Wert des Attributes *IHK_Klasse* innerhalb der Werte *Radfahren*, *Tretbootfahren*, *Segelbootfahren* und *Paddel-_und_Ruderbootfahren* befinden. Zum zweiten muss das gesuchte Individuum in der Tabelle *iar_tripel* die Rolle *hatJahreszeit* und das verknüpfte Individuum *Sommersaison* besitzen. Die zweite Selektionsbedingung wird durch eine Unterabfrage realisiert.

In der ersten Selektionsbedingung ist es nicht möglich, eine Angabe wie

WHERE IHK_Klasse IN (SELECT K_Name FROM klassen WHERE K_Oberklasse = 'Sport')

zu machen, da man mit dieser Unterabfrage nur die erste Ebene von Unterklassen der Klasse *Sport* abfragen würde. Dadurch würden nur Individuen der Klasse *Radfahren* gefunden werden, nicht aber die der Klassen *Radfahren*, *Tretbootfahren*, *Segelbootfahren* und *Paddel-_und_Ruderbootfah-*

ren, da diese Unterklassen der Klasse *Wassersport* sind.

Frage 2: „Was kann ich tagsüber in Hamburg kostengünstig (max. 10 Euro) unternehmen?“

SELECT IHK_Individuum AS Individuum, IHK_Klasse AS Klasse FROM individuum_hat_klasse

WHERE IHK_Klasse IN (SELECT K_Name FROM klassen WHERE K_Oberklasse IN ('Sport', 'Wassersport', 'Ausgehen', 'Kunst_und_Kulturerlebnis'))

AND IHK_Individuum IN (SELECT IART_Individuum1 FROM iar_tripel WHERE IART_Rolle = 'hatTageszeit' AND IART_Individuum2 IN ('Vormittag', 'Nachmittag'))

AND IHK_Individuum IN (SELECT IART_Individuum1 FROM iar_tripel WHERE IART_Rolle = 'hatPreis' AND IART_Individuum2 IN ('kostenlos', 'bis_zehn_Euro'));

Individuum	Klasse
Fischmarkt_Breite_Straße	Radfahren
Jungfernstieg_Ballindamm	Radfahren

Abbildung 7.5: Ergebnis der zweiten inhaltlichen Abfrage in *Erlebnis Hamburg*

Für das richtige Ergebnis müssen in dieser Abfrage drei Bedingungen erfüllt sein. Erstens müssen die gesuchten Individuen in einer Klasse sein, die eine Unterklasse der Klassen *Sport*, *Wassersport*, *Ausgehen* oder *Kunst_und_Kulturerlebnis* ist. Zweitens müssen diese Individuen in der Tabelle *iar_tripel* die Rolle *hatTageszeit* und das verknüpfte Individuum *Vormittag* oder *Nachmittag* besitzen. Drittens muss in der gleichen Tabelle die Rolle *hatPreis* und das verknüpfte Individuum *kostenlos* oder *bis_zehn_Euro* für die gesuchten Individuen eingetragen sein.

Diese drei Bedingungen werden durch Unterabfragen realisiert. Das Ergebnis ist die Schnittmenge dieser drei Unterabfragen und besteht in diesem Fall nur aus zwei Individuen. Die Antwort auf die oben gestellte Frage ist demnach, dass man tagsüber in Hamburg kostengünstig auf verschiede-

nen Strecken Fahrradfahren kann.

Frage 3: „Welche kulturellen Aktivitäten bieten sich in der Wintersaison an einem Sonntag an?“

SELECT IKRT_Individuum AS Individuum, IKRT_Rolle AS Rolle, IKRT_Wert AS Wert FROM ikr_tripel

WHERE IKRT_Individuum IN (SELECT IHK_Individuum FROM individuum_hat_klasse WHERE IHK_Klasse IN (SELECT K_Name FROM klassen WHERE K_Oberklasse = 'Kunst_und_Kulturerlebnis'))

AND IKRT_Individuum IN (SELECT IART_Individuum1 FROM iar_tripel WHERE IART_Rolle = 'hatJahreszeit' AND IART_Individuum2 = 'Wintersaison')

AND IKRT_Individuum IN (SELECT IART_Individuum1 FROM iar_tripel WHERE IART_Rolle = 'hatWochentag' AND IART_Individuum2 = 'Sonntag');

Individuum	Rolle	Wert
Cap_San_Diego	hatPLZ	20459
Cap_San_Diego	hatStraße	Überseebrücke
Cap_San_Diego	hatWebseite	http://www.capsandiego.de
Deutsches_Schauspielhaus	hatPLZ	20099
Deutsches_Schauspielhaus	hatStraße	Kirchenallee 39
Deutsches_Schauspielhaus	hatWebseite	http://www.schauspielhaus.de/
Fischmarkt	hatPLZ	22767
Fischmarkt	hatStraße	Fischmarkt
Fischmarkt	hatWebseite	http://www.hamburg-tourism.de/themen-touren/hambu
Gewürzmuseum	hatPLZ	20457
Gewürzmuseum	hatStraße	Am Sandtorkai 32
Gewürzmuseum	hatWebseite	http://www.spicys.de
Landungsbrücken	hatPLZ	20459
Landungsbrücken	hatStraße	St.-Pauli-Hafenstraße
Landungsbrücken	hatWebseite	http://www.hamburg-tourism.de/themen-touren/hambu
Ohnsorg_Theater	hatPLZ	20354
Ohnsorg_Theater	hatStraße	Große Bleichen 23 – 25
Ohnsorg_Theater	hatWebseite	http://www.ohnsorg.de

Abbildung 7.6: Ergebnis der dritten inhaltlichen Abfrage in *Erlebnis Hamburg*

Hierbei handelt es sich um eine Abfrage mit mehreren, teilweise geschachtelten Unterabfragen. In dieser Abfrage wird in der SELECT-Anweisung aber nicht die Tabelle *individuum_hat_klasse* abgefragt, sondern die Tabelle *ikr_tripel*. Dadurch lassen sich im Ergebnis nicht die den Individuen zugehörigen Klassen, sondern die Adresse und die Website der gesuchten Individuen anzeigen.

Frage 4: „Was kann ich nachmittags unternehmen, wenn ich nur eine Stunde Zeit habe?“

```
SELECT IHK_Individuum AS Individuum, IHK_Klasse AS Klasse, I_Kommentar AS Kommentar
FROM individuum_hat_klasse LEFT JOIN individuen ON IHK_Individuum = I_Name
WHERE IHK_Klasse IN (SELECT K_Name FROM klassen WHERE K_Oberklasse IN ('Sport', 'Wassersport', 'Ausgehen', 'Kunst_und_Kulturerlebnis'))
AND IHK_Individuum IN (SELECT IART_Individuum1 FROM iar_tripel WHERE IART_Rolle = 'hatTageszeit' AND IART_Individuum2 = 'Nachmittag')
AND IHK_Individuum IN (SELECT IART_Individuum1 FROM iar_tripel WHERE IART_Rolle = 'hatDauer' AND IART_Individuum2 = 'kürzer_eine_Stunde');
```

Individuum	Klasse	Kommentar
Fischmarkt_Breite_Straße	Radfahren	NULL
Jungfernstieg_Ballindamm	Radfahren	NULL

Abbildung 7.7: Ergebnis der vierten inhaltlichen Abfrage in *Erlebnis Hamburg*

Auch hier ist das Ergebnis die Schnittmenge von drei Selektionsbedingungen. Die Antwort auf die Frage lautet, dass man nachmittags auf verschiedenen Strecken Fahrradfahren kann, wenn man nur eine Stunde Zeit hat. Durch die Verknüpfung der Tabellen *individuum_hat_klasse* und *individuen* durch einen *LEFT JOIN* lassen sich im Ergebnis auch eventuell vorhande-

ne Kommentare zu den einzelnen Individuen anzeigen.

Frage 5: „Wo kann ich abends am Wochenende tanzen gehen, ohne Eintritt zu bezahlen?“

```
SELECT IHK_Individuum AS Individuum FROM individuum_hat_klasse
WHERE IHK_Klasse = 'Club_und_Discobesuch'
AND IHK_Individuum IN (SELECT IART_Individuum1 FROM iar_tripel WHERE
IART_Rolle = 'hatTageszeit' AND IART_Individuum2 = 'Abend')
AND IHK_Individuum IN (SELECT IART_Individuum1 FROM iar_tripel WHERE
IART_Rolle = 'hatPreis' AND IART_Individuum2 = 'kostenlos');
```

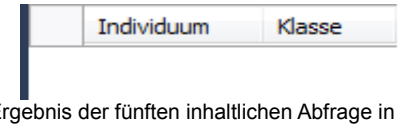


Abbildung 7.8: Ergebnis der fünften inhaltlichen Abfrage in *Erlebnis Hamburg*

Diese Abfrage besteht aus drei Selektionsbedingungen und zwei Unterabfragen. Sie liefert kein Ergebnis, d. h. in der Ontologie Erlebnis Hamburg gibt es keine Individuen, die auf die gestellte Frage zutreffen.

An dieser Stelle lässt sich zusammenfassend sagen, dass alle fünf Fragen anhand des Datenbankmodells beantwortbar sind. Die Beantwortbarkeit der Fragen wurde durch die Verwendung von Unterabfragen sowie dem Abfragen der Tabellen *individuum_hat_klasse*, *klassen* und der Tabelle *iar_tripel* oder *ikr_tripel* erreicht.

7.3 Strukturelle Abfragen in der Ontologie *Erlebnis Hamburg*

Zuletzt werden drei Abfragen durchgeführt, die die Struktur der Ontologie *Erlebnis Hamburg* betreffen. Mit folgender Abfrage lassen sich alle abge-

schlossenen Klassen der Ontologie und deren Individuen finden. Dabei werden die Tabellen *individuum_hat_klasse* und *klassen* durch einen *LEFT JOIN* verknüpft.

```
SELECT K_Name AS Abgeschlossene_Klasse, IHK_Individuum AS Individuum
FROM klassen LEFT JOIN individuum_hat_klasse ON K_Name = IHK_Klasse
WHERE K_Abgeschlossen = 1;
```

Abgeschlossene_Klasse	Individuum
Dauer	ein_bis_zwei_Stunden
Dauer	kürzer_eine_Stunde
Dauer	länger_zwei_Stunden
Jahreszeit	Sommersaison
Jahreszeit	Wintersaison
Lage	Indoor
Lage	Outdoor
Preis	bis_zehn_Euro
Preis	bis_zwanzig_Euro
Preis	kostenlos

Abbildung 7.9: Auszug aus dem Ergebnis der ersten strukturellen Abfrage in *Erlebnis Hamburg*

Mit der zweiten strukturellen Abfrage lassen sich alle Klassen finden, die mit der Klasse *Jahreszeit* disjunkt sind. Dies sind die Klassen *Dauer*, *Tageszeit* und *Wochentag*.

```
SELECT KB_Klasse1 AS Klasse, KB_Klasse2 AS Disjunkt_mit FROM k_beziehungen
WHERE KB_Klasse1 = 'Jahreszeit'
AND KB_Disjunkt = 1;
```

Klasse	Disjunkt_mit
Jahreszeit	Dauer
Jahreszeit	Tageszeit
Jahreszeit	Wochentag

Abbildung 7.10: Ergebnis der zweiten strukturellen Abfrage in *Erlebnis Hamburg*

Das Ergebnis der dritten Abfrage sind die Rolleneinschränkungen der Klasse *Aktivität*, die in der Tabelle *re_asvalues* eingetragen sind. Dies sind die Einschränkungen, bei denen der Wertebereich einer Rolle aus allen Werten oder aus mindestens einem Wert einer definierten Klasse besteht.

SELECT REAS_Klasse1 AS Klasse, REAS_Rolle AS für_Rolle, REAS_Klasse2 AS Werte_aus_Klasse, REAS_AllValues AS alle_Werte, REAS_SomeValues AS mindestens_einen_Wert

FROM re_asvalues

WHERE REAS_Klasse1 = 'Aktivität';

Klasse	für_Rolle	Werte_aus_Klasse	alle_Werte	mindestens_einen_Wert
Aktivität	hatJahreszeit	Jahreszeit	0	1
Aktivität	hatPreis	Preis	0	1
Aktivität	hatWochentag	Wochentag	0	1

Abbildung 7.11: Ergebnis der dritten strukturellen Abfrage in *Erlebnis Hamburg*

Rolleneinschränkungen einer Klasse werden in einer Ontologie an die jeweiligen Unterklassen vererbt (vgl. WIKIPEDIA 2010 b). Das bedeutet z. B., dass die Klasse *Ausgehen*, die eine Unterklasse der Klasse *Aktivität* ist, alle Einschränkungen ihrer Oberklasse erbt. Für das Datenbankmodell heißt das, dass man mehrere Abfragen durchführen muss, um alle Rolleneinschränkungen, auch die geerbten, einer Klasse zu erhalten. Bei einem automatisierten Abfragen muss dies mit PHP realisiert werden.

Die Ontologie *Erlebnis Hamburg* lässt sich innerhalb des Datenbankmodells auch strukturell abfragen. Damit kann man sagen, dass es möglich ist, sowohl die Ontologie *lookedup* als auch *Erlebnis Hamburg* vollständig in dem Modell zu speichern und beide Ontologien auf Inhalt und Struktur abzufragen.

8 Schlussbemerkungen

Nach einigen theoretischen Grundlagen wurde in der vorliegenden Arbeit ein Datenbankmodell erstellt, mit dem sich Ontologien unterschiedlicher Komplexität in einer relationalen Datenbank abbilden lassen. Zur Überprüfung der Funktionalität dieses Modells wurden zwei Ontologien in jeweils einer relationalen Datenbank gespeichert. Beide Ontologien konnten erfolgreich auf inhaltliche und strukturelle Aspekte abgefragt werden. Damit lässt sich sagen, dass eine Funktionalität des erstellten Datenbankmodells gegeben ist. Dies ist jedoch mit einigen Einschränkungen verbunden, die im Folgenden genannt werden.

Als Primärschlüssel der einzelnen Tabellen wurden größtenteils die Bezeichner der Ontologie-Elemente bzw. Kombinationen aus diesen verwendet. Der Grund dafür lag in den einfacheren Abfragemöglichkeiten der Datenbank und darin, dass diese Elemente innerhalb einer Ontologie immer eindeutig sind und somit als Primärschlüssel verwendet werden konnten. Durch die Wahl dieser Primärschlüssel entsteht aber die Einschränkung, dass es nicht möglich ist, mehrere Ontologien, die gleiche Bezeichner für unterschiedliche Konzepte haben, innerhalb einer relationalen Datenbank abzubilden.

Bei der Erstellung des Datenbankmodells wurde versucht, die Funktionalität von OWL DL zu erreichen. Diese Funktionalität wurde mit der Einschränkung umgesetzt, dass es nicht gelungen ist, logische Konstruktoren auf Klassen in das Modell miteinzubeziehen. Da es die Verwendung von logischen Konstruktoren jedoch ermöglicht, komplexere Ontologien zu erstellen, als die in der Arbeit verwendeten, sollte das Modell auch anhand solcher Ontologien getestet werden und bei Bedarf funktionell erweitert werden.

Nicht berücksichtigt wurden dagegen Rolleneinschränkungen für konkrete Rollen, da diese in den verwendeten Ontologien nicht enthalten waren. Tabellen für solche Einschränkungen können aber nach dem Vorbild der vorhandenen Tabellen erstellt und in das Modell integriert werden.

Für Oberklassen und Oberrollen wurden in dem Modell eigene Tabellen erstellt, die jeweils nur den Namen des Elementes als Primärschlüssel enthalten. Damit sollten Lösch- und Änderungsanomalien verhindert werden. Hier wäre zu überlegen, ob man, der Übersichtlichkeit wegen, auf diese Tabellen verzichtet und die Oberklassen und -rollen direkt in die Relationen für die Klassen und Rollen einträgt. Bei einem automatisierten Bearbeiten der Datenbank wäre darauf zu achten, Lösch- und Änderungsanomalien eventuell durch Programmierung mit PHP zu verhindern.

Konkrete und abstrakte Rollen wurden entsprechend der Einschränkung von OWL DL durch die Erstellung von zwei verschiedenen Tabellen getrennt. Auch an dieser Stelle könnte man prüfen, ob es möglich ist, beide Arten von Rollen in einer Tabelle darzustellen. Dies würde ebenfalls der Übersichtlichkeit dienen, hätte aber zur Folge, dass es zu mehr *NULL*-Werten in der Tabelle kommen würde.

In dieser Arbeit wurde OWL in der Version berücksichtigt, wie sie vom W3C am 10. Februar 2004 empfohlen wurde und wie sie online auf den Seiten des W3C einzusehen ist (vgl. OWL-REFERENZ 2004). Nicht berücksichtigt wurden die Änderungen in der Funktionalität, die OWL 2 mit sich bringt. Eine Übersicht zu diesen Funktionen findet sich ebenfalls auf den Seiten des W3C (vgl. OWL2 2009).

Ebenfalls nicht berücksichtigt wurde die Anwendung von terminologischer Logik und die Möglichkeit von automatischen Schlussfolgerungen. Auch das Definieren von Axiomen innerhalb einer Ontologie wurde vernachlässigt.

Trotz dieser Einschränkungen konnte in der vorliegenden Arbeit gezeigt werden, dass es möglich ist, Ontologien in relationalen Datenbanken abzubilden und diese sowohl inhaltlich als auch strukturell abzufragen. Dabei konnte die Funktionalität der Teilsprache OWL DL zum großen Teil umgesetzt werden.

Im Rahmen der Chatbotprogrammierung muss jetzt geprüft werden, wie diese Datenbanken zu verwenden sind. Im Zuge dieser Überprüfung müssen die Möglichkeiten untersucht werden, die die Skriptsprache PHP in Be-

zug auf das Abfragen und Bearbeiten der Datenbanken und somit der enthaltenen Ontologien bietet. In Bezug auf Logiken und automatische Schlussfolgerungen sollten ebenfalls die Möglichkeiten von PHP überprüft werden.

Literaturverzeichnis

BIRKENBIHL 2006

BIRKENBIHL, Klaus: Standards für das Semantic Web. In: PELLEGRINI, Tassilo (Hrsg.) ; BLUMAUER, Andreas (Hrsg.): *Semantic Web : Wege zur vernetzten Wissensgesellschaft*. Berlin : Springer, 2006. - ISBN 3-540- 29324-8

BLUMAUER 2006

BLUMAUER, Andreas ; PELLEGRINI, Tassilo: Semantic Web und semantische Technologien : Zentrale Begriffe und Unterscheidungen. In: PELLEGRINI, Tassilo (Hrsg.) ; BLUMAUER, Andreas (Hrsg.): *Semantic Web : Wege zur vernetzten Wissensgesellschaft*. Berlin : Springer, 2006. - ISBN 3-540-29324-8

BREITMANN 2007

BREITMANN, Karin K. ; CASANOVA, Marco Antonio ; TRUTZKOWSKI, Walter: *Semantic Web : Concepts, Technologies and Applications*. London : Springer, 2007. - ISBN 978-1-84628-581-3

DACONTA 2003

DACONTA, Michael C. ; OBRST, Leo J. ; SMITH, Kevin T.: *The Semantic Web : A Guide to the Future of XML, Web Services, and Knowledge Management*. Indianapolis : Wiley, 2003. - ISBN 0-471-43257-1

DIN 1463/1 1987

NORM DIN 1463 Teil 1 11.87. *Erstellung und Weiterentwicklung vonThesauri : Einsprachige Thesauri*

EHRIG 2006

EHRIG, Marc ; STUDER, Rudi: Wissensvernetzung durch Ontologien. In: PELLEGRINI, Tassilo (Hrsg.) ; BLUMAUER, Andreas (Hrsg.): *Semantic Web : Wege zur vernetzten Wissensgesellschaft*. Berlin : Springer, 2006. - ISBN 3-540-29324-8

EISENMENGER 2008

EISENMENGER, Andre: *Semantic Web: Konzept einer OWL-Ontologie für einen Chatbot.* Hamburg, Hochschule für Angewandte Wissenschaften, Studiendepartement Information, Dipl.-Arb., 2008

ELMASRI 2005

ELMASRI, Ramez ; NAVATHE, Shamkant B.: *Grundlagen von Datenbanksystemen : Ausgabe Grundstudium*. 3. Aufl. München : Pearson Studium, 2005. - ISBN 3-8273-7153-8

GEISLER 2009

GEISLER, Frank: *Datenbanken : Grundlagen und Design*. 3. aktualisierte und erweiterte Aufl. Heidelberg : mitp, 2009. - ISBN 978-3-8266-5529-6

GRIMM 2007

GRIMM, Stefan ; HITZLER, Pascal ; ABECKER, Andreas: Knowledge Representation and Ontologies. In: STUDER, Rudi (Hrsg.) ; GRIMM, Stefan (Hrsg.) ; ABECKER, Andreas (Hrsg.): *Semantic Web Services : Concepts, Technologies and Applications*. Berlin : Springer, 2007. - ISBN 978-3-540-70893-3

GRUBER 1993

GRUBER, Tom: A Translation Approach to Portable Ontology Specifications. In: *Knowledge Acquisition* 5 (1993), Nr. 2, S. 199-220. - ISSN 1071-5819 (online abrufbar unter URL: http://tomgruber.org/writing/ontolingua-kaj-1993.pdf)

GRÜTTER 2008

GRÜTTER, Rolf: *Semantic Web zur Unterstützung von Wissensgemeinschaften*. München : Oldenbourg, 2008. - ISBN 978-3-486-58626-8

HESSE 2002

HESSE, Wolfgang: Ontologie(n). In: *Informatik Spektrum* 25 (2002), Nr. 6, S. 477 – 480. - ISSN 0170-6012

HITZLER 2008

HITZLER, Pascal ; KRÖTZSCH, Markus ; RUDOLPH, Sebastian ; SURE, York: *Semantic Web : Grundlagen*. Berlin : Springer, 2008. - ISBN 978-3-540-33993-9

KANNENGIESSER 2007

KANNENGIESSER, Caroline ; KANNENGIESSER, Matthias: *PHP5 / MySQL5. 2., überarb. Ausg.* Poing : Franzis, 2007

KIENREICH 2006

KIENREICH, Wolfgang ; STROHMAIER, Markus: Wissensmodellierung : Basis für die Anwendung semantischer Technologien. In: PELLEGRINI, Tassilo (Hrsg.) ; BLUMAUER, Andreas (Hrsg.): *Semantic Web : Wege zur vernetzten Wissensgesellschaft*. Berlin : Springer, 2006. - ISBN 3-540-29324-8

KOOS 2005

KOOS, Karen: *Wissen macht warm : Modellierung einer Ontologie zur Unterstützung einer nutzerfreundlichen Online-Wärmedämmberatung*. Hamburg, Hochschule für Angewandte Wissenschaften, Fachbereich Bibliothek und Information, Dipl.-Arb., 2005

MATTHIESSEN 2008

MATTHIESSEN, Günter ; UNTERSTEIN, Michael: R*elationale Datenbanken und Standard-SQL : Konzepte der Entwicklung und Anwendung*. München : Addison-Wesley, 2008. - ISBN 978-3-8273- 2656-0

MAURICE 2010

MAURICE, Florence: *PHP 5.3 & MySQL 5.1 : Der Einstieg in die Programmierung dynamischer Websites*. München : Addison-Wesley, 2010. - ISBN 978-3-8273-2723-9

MYSQL WORKBENCH 2010

SUN MICROSYSTEMS GMBH: *MySQL : MySQL Workbench 5.2.* URL: http://www.mysql.de/products/workbench/. Stand: 2010-08-03. Letzter Aufruf: 2010-08-03

OLS 2010

EUROPEAN BIOINFORMATICS INSTITUTE (Hrsg.): *Ontology Lookup Service*. URL: http://www.ebi.ac.uk/ontology-lookup/databaseExport.do. Stand: 2010-08-04. Letzter Aufruf: 2010-08-04

OWL-REFERENZ 2004

WORLD WIDE WEB CONSORTIUM (Hrsg.): *OWL Web Ontology Language Reference.* URL: http://www.w3.org/TR/owl-ref/. Stand: 2004-02-10. Letzter Aufruf: 2010-08-08

OWL2 2009

WORLD WIDE WEB CONSORTIUM (Hrsg.): *OWL 2 Web Ontology Language Document Overview.* URL: http://www.w3.org/TR/owl2-overview/. Stand: 2009-11-12. Letzter Aufruf: 2010-08-08

PASSIN 2004

PASSIN, Thomas, B.: *Explorer's Guide to the Semantic Web*. Greenwich : Manning, 2004. - ISBN 1-932394-20-6

PROTÉGÉ 2010

STANFORD UNIVERSITY SCHOOL OF MEDICINE (Hrsg.); STANFORD CENTER FOR BIOMEDICAL INFORMATICS RESEARCH (Hrsg.): *Protégé : JDBC Backend Design Rationale*. URL: http://protege.stanford.edu/doc/design/jdbc_backend.html. Stand: 2010-08-04. Letzter Aufruf: 2010-08-04

ROLLAND 2003

ROLLAND, F. D.: *Datenbanksysteme : Im Klartext*. München : Pearson Studium, 2003. - ISBN 3-8273-7066-3

SCHUBERT 2004

SCHUBERT, Matthias: *Datenbanken : Theorie, Entwurf und Programmierung relationaler Datenbanken*. 1. Aufl. Wiesbaden : Teubner, 2004. - ISBN 3-519-00505-0

STOCK 2008

STOCK, Wolfgang ; STOCK, Mechthild: *Wissensrepräsentation : Informationen auswerten und bereit stellen*. München : Oldenbourg, 2008. - ISBN 978-3-486-58439-4

STUCKENSCHMIDT 2009

STUCKENSCHMIDT, Heiner: *Ontologien : Konzepte, Technologien und Anwendungen*. Berlin : Springer, 2009. - ISBN 978-3-540-79333-5

WIKIPEDIA 2010 a

WIKIPEDIA: *Topic Maps*. URL: http://de.wikipedia.org/wiki/Topic_Maps. Stand: 2010-02-24. Letzter Aufruf: 2010-03-16

WIKIPEDIA 2010 b

WIKIPEDIA: *Ontologie (Informatik).* URL: http://de.wikipedia.org/wiki/Ontologie_(Informatik). Stand: 2010-03-07. Letzter Aufruf: 2010-08-20

WIKIPEDIA 2010 c

WIKIPEDIA: *Semantisches Web.* URL: http://de.wikipedia.org/wiki/Semantic_web. Stand: 2010-03-11. Letzter Aufruf: 2010-03-22

WIKIPEDIA 2010 d

WIKIPEDIA: *MySQL*. URL: http://de.wikipedia.org/wiki/MySQL. Stand: 2010-08-03. Letzter Aufruf: 2010-08-03

WIKIPEDIA 2010 e

WIKIPEDIA: *InnoDB.* URL: http://de.wikipedia.org/wiki/InnoDB. Stand: 2010-07-13. Letzter Aufruf: 2010-08-13

WOLLSCHLÄGER 2010

WOLLSCHLÄGER, Patricia ; INGBER, Sina: *Freizeit und Hansestadt Hamburg : Eine Ontologie zum Kennenlernen und Erleben von Hamburg.* Hamburg, Hochschule für Angewandte Wissenschaften, Departement Information, Seminararb., 2010

Anhang: CD - Inhalt

Die der Arbeit beigefügte CD enthält den Anhang, bestehend aus folgenden vier Dateien:

- Sicherungsdateien der beiden erstellten Datenbanken (erlebnis-hamburg.sql und lookedup.sql)
- SQL-Datei zur Erstellung einer leeren Datenbank nach dem in der Arbeit erstellten Datenbankmodell (create_script.sql)
- EER-Modell des Datenbankmodells zur Ansicht in der MySQL-Workbench (ontologien.mwb)

Eidesstattliche Versicherung

Ich versichere, die vorliegende Arbeit selbstständig ohne fremde Hilfe verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt zu haben. Die aus anderen Werken wörtlich entnommenen Stellen oder dem Sinn nach entlehnten Passagen sind durch Quellenangabe kenntlich gemacht.

Ort, Datum Unterschrift