Universität des Saarlandes - Fachrichtung 5.6 Informationswissenschaft:

Elektronischen Publizieren am Beispiel von Friedrich Nietzsche

Konzept für ein datenbankbasiertes Registersystem

Hauptseminararbeit von:

Till Kinstler

till@phil.uni-sb.de

April 2001

Inhaltsverzeichnis

Kapitel 1

Konzept

1.1 Einleitung

In dieser Arbeit wird ein Konzept zur Umsetzung eines elektronischen Registers für das "Nietzsche Online"-Angebot im WWW[1], das im Rahmen eines Forschungsseminars der Fachrichtung Informationswissenschaft im Wintersemester 2000/2001 entwickelt wurde, vorgestellt. Dieses Register soll – vergleichbar mit einem gedruckten Register in einem Buch – dem Benutzer eine Möglichkeit des Zugangs zu den in dem WWW-Angebot enthaltenen Inhalten bieten. Der Benutzer soll anhand der Einträge des Registers gezielt Inhalte von "Nietzsche Online" erschließen können. Die konkrete Ausgestaltung der Benutzerschnittstelle ist jedoch nicht Gegenstand dieser Arbeit. Die Möglichkeiten zur Gestaltung dieser Schnittstelle ergeben sich aus den (Abfrage-)Funktionen des Register-Systems. Es erscheint jedoch sinnvoll, dem Benutzer diese Funktionen nicht alle an einer einzigen Schnittstelle anzubieten, sondern sie über mehrere Zugangswege zu verteilen (zum Beispiel eine "einfache Suche" und eine "komplexe Suche" oder eine "Suche nur in den häufig gestellten Fragen" usw., die jeweils nur einen Teil der Abfragefunktionen benutzen). Deswegen sollte die kon-

[1] Das Angebot ist derzeit (April 2001) unter der Adresse `http://nietzsche.ps.uni-sb.de/` im WWW verfügbar.

krete Gestaltung der Benutzerschnittstelle des Registers als Teil der Gesamtgestaltung eines Online-Angebots behandelt werden, hier wird nur auf die technischen Möglichkeiten des Systems eingegangen.

Grundsätzlich ist das beschriebene System nicht auf Inhalte mit Bezug zu Friedrich Nietzsche beschränkt. Es lässt sich ebenso auf andere “Inhalts-Sammlungen” anwenden, die in ähnlich strukturierter Form wie “Nietzsche Online” vorliegen. Allerdings werden die Anforderungen und die Implementierung hier anhand des “Nietzsche Online”-Angebots beschrieben.

1.2 Voraussetzungen

Das Register soll den Benutzer über Begriffe und Begriffsketten (im Folgenden “*Bezeichnungen*” genannt) zu den Inhalten des Online-Angebots (“*Ressourcen*”) führen.

1.2.1 Bezeichnungen

Bezeichnungen können Zeichenketten sein, die in den Ressourcen selbst enthalten sind, wie ein Stichwort aus einem Text, der Autor eines Briefes oder ein Ort oder sie können Ressourcen beschreiben (zum Beispiel ein Schlagwort zu einem Bild oder eine thematische Zuordnung eines Zitats). Es können aber auch Einträge aus Wörterbüchern sein, die einen Bezug zu den Ressourcen oder den Bezeichnungen aufweisen, oder zum Beispiel Synonyme, Oberbegriffe oder Flexionsformen zu anderen Bezeichnungen. Bezeichnungen können aus einem oder mehreren Wörtern bestehen, grundsätzlich können es beliebige Zeichenketten sein. Sinnvoll sind natürlich nur solche Zeichenketten, die in Beziehungen zu anderen Bezeichnungen oder Ressourcen stehen.

1.2.2 Ressourcen

Ressourcen können alle Inhalte sein, die innerhalb oder außerhalb des Angebots im Internet eindeutig adressierbar sind, das heißt sie müssen über einen Uniform Ressource Locator (URL) ansprechbar sein. Mit URL ist hier eine spezielle Form von Uniform Ressource Identifiern (URI) nach (Barners-Lee u. a. 1998, Kapitel 1.2) gemeint[2]. Grundsätzlich ist auch die Unterstützung anderer Adressierungssysteme zum Verweis auf Ressourcen vorstellbar, allerdings handelt es sich bei dem hier vorgestellten System um ein auf dem World Wide Web basierendes System, das deswegen das Standardadressierungssystem des WWWs, nämlich URLs, benutzt[3].

Ressourcen müssen nicht in textueller Form oder einem bestimmten Dateiformat vorliegen; es kann auch auf binäre Daten wie Bilder, Filme oder Tondateien verwiesen werden. Zur automatischen Aufnahme in das Register sind allerdings strukturierte und offene Datenformate (insbesondere XML) am besten geeignet.

1.2.3 Beziehungen

Bezeichnungen stehen in Beziehung zu anderen Bezeichnungen und/oder Ressourcen. Eine Bezeichnung kann zum Beispiel ein *Synonym zu* einer anderen Bezeichnung oder ein *Schlagwort zu* einer Ressource sein. Eine Beziehung zwischen zwei Bestandteilen des Registers (Bezeichnungen oder Ressourcen) ist also immer von einem bestimmten Typ (zum Beispiel *Synonym zu* oder *Schlagwort zu*).

Diese typisierten Beziehungen soll dem Benutzer die Navigation durch das Register und damit die Inhalte, auf die es verweist, ermöglichen, indem der Benutzer sich zwischen Bezeichnungen und Ressourcen entlang der Beziehungen bewegen und durch die Typisierung

[2]Das bedeutet nicht, dass zu einer Ressource eine URL gespeichert werden muss, es kann auch ein anderer eindeutiger Identifier für diese Ressource abgelegt werden, aus dem sich eine URL für diese Ressource eindeutig berechnen lässt. Eine Diskussion dieser Frage erfolgt in Kapitel 2.2.

[3]zur Adressierung im WWW siehe (Fielding u. a. 1999, Kapitel 3.2 und 5.2)

der Beziehungen die Auswahl an Bezeichnungen und Ressourcen gezielt steuern kann. Der Benutzer soll an das Register Anfragen wie “Gib mir alle Briefe zum Thema Nietzsche und Gott” stellen können. Das System soll dann alle Ressourcen vom Typ *Brief* ausgeben, zu denen die Bezeichnung *Gott* die Beziehung *ist Thema von* hat. Sind die Ergebnisse nicht ausreichend, hilft vielleicht ein Oberbegriff zu *Gott* wie *Religion*. Das wäre eine weitere Anfrage an das System: “Gib mir den *Oberbegriff zu Gott* und alle *Briefe*, die diesen *zum Thema* haben”.

Die Beziehungstypen, die das System unterstützt, sollen frei definierbar und grundsätzlich in der Anzahl unbegrenzt sein. Auch das nachträgliche Hinzufügen von Typen soll möglich sein. Die Beziehungen zwischen Bezeichnungen und Ressourcen sollen jeden definierten Beziehungstyp annehmen können. Die Anzahl der Beziehungen, die von einer Bezeichnung ausgehen oder auf eine Bezeichnung oder Ressource verweisen, soll grundsätzlich unbegrenzt sein[4]. Es soll auch möglich sein, dass zwischen zwei Bezeichnungen oder einer Bezeichnung und einer Ressource mehrere Beziehungen unterschiedlichen Typs bestehen.

1.2.4 Datentypen

Die Inhalte von “Nietzsche Online” liegen überwiegend in Form von wohlgeformeten (*well formed*) und gültigen (*valid*) XML-Dateien vor[5]. Auch Inhalte, die in binären Formaten abgelegt sind (Bilder, Töne, Filme), werden durch zugehörige XML-Daten beschrieben. In den XML-Dateien gibt es ausgezeichnete Einträge, die Schlagwörter, Titel, Themen, Namen usw. enthalten, die als Bezeichnungen in das Register übernommen werden können. Dabei kann automatisch die Beziehung zwischen diesen Bezeichnungen und den Ressourcen, aus denen sie gewonnen werden, hergestellt werden.

[4] Beschränkungen bei der Anzahl der Beziehungstypen und der Anzahl der Beziehungen ergeben sich in der Praxis aus der konkreten Implementierung des Register-Systems und den Beschränkungen der verwendeten Systemumgebung. So kann eine MySQL-Datenbank auf Linux 2.2-Systemen auf x86-Architekturen ohne aufwendige Anpassungen maximal 2 GB groß werden.

[5] Die Bedeutung der Eigenschaften *well formed* und *valid* bei XML ist in (W3C 2000, Frage D.2) erklärt.

Ressourcen können aber auch in Form von anderen Datentypen vorliegen, allerdings müssen sie dann manuell in das System integriert werden.

1.3 Systemarchitektur

Das Registersystem muss die Registerdaten speichern und bereitstellen und soll flexibel auf unterschiedliche Benutzeranfragen antworten können. Der Benutzer greift über das World Wide Web auf das Register zu. Da die Benutzeranfragen an das Register sehr vielfältig sein können, muss die Benutzerschnittstelle dynamisch gestaltet werden. Eine Lösung, bei der das Register in Form von statischen HTML- oder auch XML-Dateien mit starren Verweisen innerhalb derselben und auf Ressourcen gestaltet wird, wäre mit großem Pflegeaufwand verbunden und würde den Benutzer auf die vom Ersteller des Registers “vorgedachten” Navigationswege durch das System und die Inhalte einschränken. Zur Speicherung der unterschiedlichen “Objekte” (Bezeichnungen und Ressourcen), die das Register enthält, und der Beziehungen zwischen diesen wird deswegen eine relationale Datenbank verwendet. In dieser relationalen Datenbank können die “Objekte” mit ihren Eigenschaften beschrieben werden. Auch die Beziehungen zwischen den “Objekten” können abgelegt und beschrieben werden. Mit SQL, der Standardsprache für relationale Datenbanken, können die Objekte nach ihrer Eigenschaften und ihrer Beziehungen und deren Eigenschaften schnell und flexibel selektiert werden.

Der Benutzer merkt nicht, dass er eine Datenbankanwendung verwendet. Er greift über eine HTML-Oberfläche, die in einem Web-Browser dargestellt wird, auf das Register zu. Diese HTML-Oberfläche wird bei jedem Zugriff des Benutzers von Programmen auf einem HTTP-Server generiert, die auch die Formulierung der Anfragen an die Datenbank in SQL übernehmen. Diese Anfragen werden über Parameter, die der Benutzer über die HTML-Oberfläche auswählt (durch explizite Auswahl oder implizit durch Vorgaben, die sich aus dem “Programmablauf” ergeben), gesteuert.

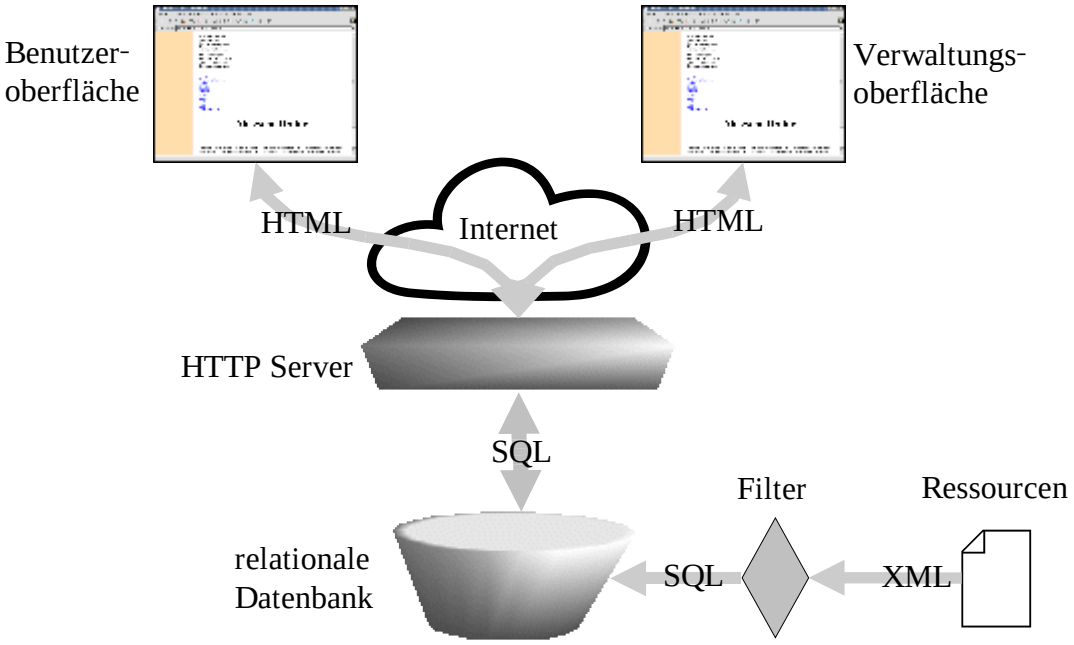

Abbildung 1.1: grobe Architektur des Registersystems

Durch die Speicherung der Registerdaten in einer relationalen Datenbank ist jederzeit eine Veränderung (Hinzufügen, Löschen, Verändern von Einträgen) des Datenbestands möglich. Eine Änderung der Benutzerschnittstelle ist dazu nicht notwendig. Die Erfassung der Registerdaten aus strukturierten Dateien, die die notwendigen Angaben zur Beschreibung von Objekten in der Datenbank enthalten, kann mit angepassten Importfiltern automatisch erfolgen. Zusätzlich ist eine Verwaltungsschnittstelle notwendig, die die manuelle Eingabe von nicht automatisch erfassbaren Daten erlaubt, insbesondere die Herstellung von Beziehungen zwischen Objekten des Registers. Diese Verwaltungsschnittstelle wird ebenfalls auf HTML basierend zur Benutzung mit einem Webbrowser ausgeführt. Damit nur berechtigte Benutzer die Verwaltungsfunktionen nutzen können, muss der Zugriff auf diese durch geeignete Maßnahmen beschränkt werden, zum Beispiel durch Authentifizierung mit Benut-

zernamen und Passwörtern.

In Abbildung 1.1 ist die grobe Architektur des Registersystems dargestellt. Grundlage bildet eine relationale Datenbank, in der alle Registerdaten abgelegt sind. Zur Benutzung des Systems stehen eine normale Benutzeroberfläche und eine Verwaltungsoberfläche in HTML zur Verfügung. Der Benutzer greift über einen HTTP-Server auf das System zu, der die HTML-Seiten für die Darstellung der Oberflächen generiert. Dazu werden auf dem HTTP-Server Programme ausgeführt, die die Benutzeranfragen in SQL umsetzen und an die Datenbank übergeben und die Ergebnisse aus der Datenbank als HTML über das HTTP-Protokoll durch das Internet an den Benutzer zurück geben. Für Ressourcen, die in strukturierten Formaten wie XML vorliegen, gibt es zusätzlich Importfilter, die Elemente aus den Ressourcen auslesen und per SQL in die Datenbank eintragen.

Kapitel 2

Implementierung

Dieses Kapitel beschreibt einen Entwurf für eine mögliche Implementierung des Registersystems. Das System ist bisher noch nicht vollständig wie hier beschrieben implementiert.

Zunächst wird die Softwareumgebung, in der das System eingebettet wird, beschrieben. Danach folgt die Beschreibung der Datenbankstruktur, in der die Registerdaten abgelegt werden. Der letzte Teil dieses Kapitels befasst sich schließlich mit den Möglichkeiten der Gestaltung der (Benutzer-)Schnittstellen zu der Datenbank.

2.1 verwendete Software

Zur Implementierung des Registersystems muss zunächst festgelegt werden, in welcher Softwareumgebung das System realisiert werden soll. Dazu gehören das Datenbankmanagementsystems (DBMS), die Programmiersprache für die Schnittstellen zur Datenbank und der HTTP-Servers, über den der Zugriff auf das Registersystem erfolgt.

2.1.1 Datenbankmanagementsystem

Als relationales Datenbankmanagementsystem wird MySQL verwendet. Ein wichtiger Grund für diese Entscheidung ist die lizenzkostenfreie Verfügbarkeit von MySQL als Open Sour-

ce Software für den Einsatz in dieser Anwendung (siehe (MLI 2001)). Darüberhinaus ist MySQL ein kompaktes, schnelles, ressourcenschonendes und robustes relationales Datenbanksystem[1], das sich wegen diesen Eigenschaften gerade beim Einsatz für Anwendungen im WWW großer Beliebtheit erfreut. MySQL fehlen jedoch einige Eigentschaften anderer relationaler Datenbanksysteme wie Oracle oder DB/2. Ein wesentlicher Nachteil ist das Fehlen von Fremdschlüsselbeziehungen zwischen Tabellen. Das erfordert besondere Vorsicht beim Löschen und Ändern von Daten, damit die Integrität des Datenbestands gewährleistet bleibt. MySQL unterstützt den Sprachumfang von entry level ANSI SQL92 (ANSI X3.135) mit einigen Erweiterungen.

2.1.2 Programmiersprache

Die Programmiersprache, in der die Schnittstellen zur Datenbank implementiert werden, muss eine Programmierschnittstelle (API) zur verwendeten Datenbank bieten und sollte gemäß den in Kapitel 1 diskutierten Anforderungen ein einfaches Erstellen einer Benutzeroberfläche in HTML erlauben. Außerdem muss sie mit dem verwendeten HTTP-Server zusammenarbeiten.

Als Programmiersprache wurde deswegen PHP in der aktuellen Version 4 gewählt. PHP 4 bietet eine sehr mächtige Programmierschnittstelle zu MySQL (aber auch zu anderen Datenbanksystemen). Außerdem kann der Programmcode direkt in HTML eingebettet werden, wodurch sich sehr leicht HTML-Oberflächen mit dynamischen Elementen erstellen lassen. PHP 4 ist eine Sprache, die zur Laufzeit interpretiert wird. Auf dem HTTP-Server muss also eine Laufzeitumgebung für PHP 4 vorhanden sein. Für einige HTTP-Server steht eine angepasste und optimierte Laufzeitumgebung zur Verfügung, zum Beispiel für Apache HTTP Server und Microsoft Internet Information Server. Es existiert aber auch eine Laufzeitumge-

[1] Ein Geschwindigkeitsvergleich mit anderen gängigen Datenbanksystemen wie Microsoft Access, PostgreSQL, Adabas D und anderen liefert (MPE 2001).

bung, die mit jedem HTTP-Server, der über das Common Gateway Interface (CGI)[2] verfügt, eingesetzt werden kann. PHP 4 und die Laufzeitumgebungen sind Open Source Software und stehen ebenfalls lizenzkostenfrei zur Verfügung (siehe (PLI 2000)).

Die Importfilter, mit denen Ressourcen, die im XML-Format vorliegen, in die Datenbank eingelesen werden können, verlangen eine Programmiersprache, die eine Schnittstelle zur verwendeten Datenbanksoftware und Funktionen zum Umgang mit XML-Datenformaten bietet. Auch dafür ist PHP 4 geeignet. Eine gute Einführung in die Verarbeitung von XML mit PHP 4 gibt (Bergmann 2001).

2.1.3 HTTP-Server

Als HTTP-Server wird die Software Apache HTTP Server verwendet. Apache ist ein sehr weit verbreiteter WWW-Server[3]. Durch den modularen Aufbau lässt sich Apache sehr leicht mit zusätzlichen Funktionen ausstatten, so auch mit einer Laufzeitumgebung für PHP 4. Auch zur Verarbeitung von XML und XSL-Stylesheets, die bei “Nietzsche Online” ebenfalls eingesetzt werden, ist Zusatzsoftware für Apache verfügbar. Hinweise auf die Verarbeitung von XML und XSL mit Apache geben (Mintert und Menge 2000) und (Behme 2001). Apache ist Open Source Software und steht für die Verwendung für das Registersystem lizenzkostenfrei zur Verfügung (s. (ALI 2000)).

2.2 Datenbankstruktur

Alle für das Register relevanten Daten werden in einer relationalen Datenbank abgelegt. Das Registersystem kennt zwei unterschiedliche elementare “Datenobjekte”: Bezeichnun-

[2] CGI beschreibt eine Standardschnittstelle zwischen HTTP-Servern und externen Programmen, die alle gängigen Webserver mitbringen.

[3] Bei den monatlichen Messungen von `http://www.netcraft.com/`, die die verwendete HTTP-Serversoftware von über 25 Millionen Websites erfasst, hat Apache einen Anteil von deutlich über 50%, siehe `http://www.netcraft.com/survey/`.

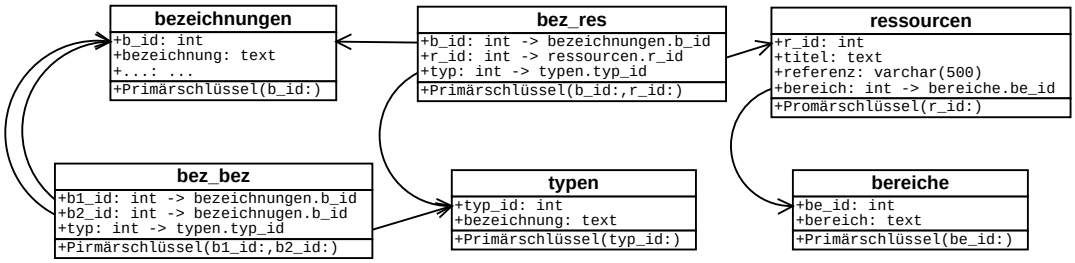

Abbildung 2.1: Datenbankstruktur des Registersystems

gen und Ressourcen. Diese werden jeweils in einer eigenen Tabelle gespeichert und mit ihren Eigenschaften beschrieben. Jede Bezeichnung kann viele Beziehungen zu anderen Bezeichnungen und Ressourcen haben. Zwischen Bezeichnungen und zwischen Bezeichnungen und Ressourcen bestehen also n:m-Beziehungen, die in der Datenbank abgebildet werden. Dazu existieren eigene Tabellen, in denen Verweise auf die beiden Enden einer Beziehung und der Typ der Beziehung zwischen diesen abgelegt werden. Außerdem sind einige Hilfstabellen sinnvoll, die einzelne Bestandteile der Datenbank näher beschreiben.

Die Datenbank besteht aus insgesamt 6 Tabellen, deren Aufbau im Folgenden erläutert und tabellarisch dargestellt wird. Abbildung 2.1 zeigt dazu ein Diagramm der Datenbankstruktur mit den Beziehungen zwischen den Tabellen.

In der Tabelle *bezeichnungen* (siehe Tabelle 2.1) werden die Bezeichnungen, die das Register enthält, abgelegt. Diese Bezeichnungen werden in das Feld bezeichnung eingetragen und bekommen beim Eintragen von der Datenbank automatisch einen eindeutigen, ganzzahligen Identifier zugeordnet. Innerhalb der Datenbank wird nur dieser Identifier zum Zugriff auf die Bezeichnungen verwendet. Hier wird davon ausgegangen, dass eine Bezeichnung eindeutig ist und nicht nach weiteren Merkmalen unterschieden werden muss. Eine solche Unterscheidung ist jedoch möglich, indem weitere Felder zur Aufnahme von Eigenschaften einer Bezeichnung verwendet werden. Dadurch könnten zum Beispiel bei mehrdeutigen Bezeichnungen die unterschiedlichen Bedeutungen differenziert werden. Andererseits erhält eine Bezeichnung erst in einer Beziehung zu einer anderen Bezeichnung oder einer Res-

source eine Bedeutung, deswegen erscheint es sinnvoller solche Unterscheidungen durch unterschiedliche Typisierungen der Beziehungen zu anderen Bezeichnungen oder Ressourcen vorzunehmen.

Feld	**Typ**	**Bedeutung**
b_id	int, auto_increment	eindeutiger, ganzzahliger Identifier für eine Bezeichnung; wird fortlaufend vergeben und dient der eindeutigen Identifizierung einer Bezeichnung innerhalb der Datenbank
bezeichnung	text	alphanumerische Darstellung der Bezeichnung
...	...	Felder, die weitere wesentliche Eigenschaften von Bezeichnungen enthalten, mit denen Bezeichnungen unterschieden werden können sollen
Primärschlüssel: b_id		

Tabelle 2.1: Tabelle *bezeichnungen*

Die Tabelle *ressourcen* (2.2) enthält die Beschreibungen der Ressourcen, auf die verwiesen wird. Eine Ressource hat einen Titel und bekommt beim Einfügen in die Datenbank automatisch einen eindeutigen Identifier (Feld r_id) zugeteilt, der die Ressource in der Datenbank repräsentiert.

In dem Feld referenz muss ein eindeutiger Verweis auf die Ressource gespeichert werden, der die Adressierung und das Auffinden der Ressource ermöglicht. Im World Wide Web dienen Uniform Ressource Locator (URL) dem eindeutigen Adressieren von Ressour-

cen (vgl. (Fielding u. a. 1999, Kapitel 3.2 und 5.2)). Eine Möglichkeit, auf Ressourcen im WWW zu verweisen, wäre also die Angabe einer URL im Feld referenz. Eine URL enthält jedoch immer den tatsächlichen Speicherort einer Ressource im Netzwerk. Sie ist per Definition eindeutig, erfüllt aber nicht das Kriterium der Persistenz: Wechselt der Speicherort einer Ressource, so ändert sich auch ihr URL, sie kann insbesondere mit einem alten URL nicht mehr aufgefunden werden (vgl. (Barners-Lee u. a. 1998, Kapitel 1.1)). Es erscheint deswegen sinnvoller, im Feld referenz ein persistentes und eindeutiges Merkmal einer Ressource abzulegen, mit der die Ressource auch nach einem Wechsel des Speicherorts auffindbar bleibt. Es ist jedoch ein Dilemma des Internets, dass es zwar einige Vorschläge zur Einführung von persistenten Identifiern für Ressourcen im Netzwerk gibt, jedoch bisher keiner dieser Vorschläge breite Unterstützung durch Netzwerkprotokolle und Anwendungen, die auf diesen aufsetzen, gefunden hat. Die Adressierung von Ressourcen im WWW geschieht eben (noch) mit URLs. Selbst wenn also jede Ressource, die in dem Registersystem verzeichnet werden soll, ein eindeutiges, persistentes Merkmal besitzt, so muss es eine einfache Regel zur Übersetzung dieses Merkmals in einen URL geben, damit die Ressource über das WWW aufgerufen werden kann. Da nicht abschätzbar ist, ob diese beiden Voraussetzungen für alle Ressourcen, die das Registersystem verzeichnen soll, erfüllt sind, scheint die Speicherung von URLs im Feld referenz derzeit die sinnvollste – wenn auch nicht beste – Lösung zu sein.

Im Feld bereich kann ein Verweis auf einen Bereich des Angebots abgelegt werden, zu dem die Ressource gehört. Bei “Nietzsche Online” könnten das zum Beispiel die Bereiche “häufig gestellte Fragen”, “Briefe” oder “Personen” sein. Damit ist eine bereichsabhängige Auswahl von Ressourcen möglich. Die Bereiche werden in der Tabelle *bereiche* (s. Tabelle 2.6) definiert und bekommen dort einen eindeutigen Identifier (Feld be_id in *bereiche*) zugeteilt, der in das Feld bereich von *ressourcen* eingetragen werden kann.

Feld	**Typ**	**Bedeutung**
r_id	int, auto_increment	eindeutiger, ganzzahliger Identifier für eine Ressource; wird automatisch fortlaufend vergeben und dient der eindeutigen Identifizierung von Ressourcen innerhalb der Datenbank
titel	text	alphanumerischer Titel für die Ressource
referenz	varchar	Verweis auf den “Fundort” der Ressource; soll eine eindeutige Adressierung der Ressource und damit das Auffinden ermöglichen
bereich	int	bei Ressourcen, die zu “Nietzsche Online” gehören: numerische Kennung auf den Bereich des Angebots, zu dem die Ressource gehört; die “Bereichsnummern” werden in der Tabelle *bereiche* definiert
Primärschlüssel: r_id; **Fremdschlüssel:** bereich->*bereiche*.be_id		

Tabelle 2.2: Tabelle *ressourcen*

Die Beziehungen zwischen Bezeichnungen werden in der Tabelle *bez_bez* (2.3) abgelegt. Dazu werden in den Feldern b1_id und b2_id Identifier von Bezeichnungen, wie sie in der Tabelle *bezeichnungen* definiert sind, eingetragen. In das Feld typ wird ein Identifier des Beziehungstyps von b1_id zu b2_id eingetragen, wie er in der Tabelle *typen* definiert ist.

Feld	Typ	Bedeutung
b1_id	int	numerischer Identifier aus *bezeichnungen* für die erste Bezeichnung, die eine Beziehung zu b2_id hat
b2_id	int	numerischer Identifier aus *bezeichnungen* für die zweite Bezeichnung, zu der b1_id eine Beziehung hat
typ	int	numerischer Identifier aus *typen*, der den Typ der Beziehung von b1_id zu b2_id beschreibt
Primärschlüssel: (b1_id, b2_id); **Fremdschlüssel:** b1_id->*bezeichnungen*.b_id, b2_id->*bezeichnungen*.b_id, typ->*typen*.typ_id		

Tabelle 2.3: Tabelle *bez_bez*

Beziehungen von Bezeichnungen zu Ressourcen sind in der Tabelle *bez_res* (2.4) gespeichert. Das Feld b_id nimmt einen numerischen Identifier einer Bezeichnung aus der Tabelle *bezeichnungen* auf, r_id speichert den numerischen Identifier einer Ressource aus der Tabelle *ressourcen*. Der Typ der Beziehung von b_id zu r_id wird durch einen Identifier eines Beziehungstyps im Feld typ festgelegt, die Definition der Identifier für Typen erfolgt in der Tabelle *typen*.

Feld	Typ	Bedeutung
b_id	int	numerischer Identifier aus *bezeihnungen* für eine Bezeichnung, die eine Beziehung zu einer Ressource hat
r_id	int	numerischer Identifier aus *ressourcen* für eine Ressource, zu der eine Beziehung besteht
typ	int	numerischer Identifier aus *typen*, der den Typ der Beziehung von b_id zu r_id beschreibt
Primärschlüssel: (b_id, r_id), **Fremdschlüssel:** b_id->*bezeichnungen*.b_id, r_id->*ressourcen*.r_id, typ->*typen*.typ_id		

Tabelle 2.4: Tabelle *bez_res*

Neben den vier zentralen Tabellen, die der Abbildung von Bezeichnungen, Ressourcen und Beziehungen zwischen diesen dienen, sind zwei weitere Hilfstabellen zur Definition von Identifiern sinnvoll. Die Tabelle *typen* (2.5) legt die eindeutigen, ganzzahligen Identifier fest, die Beziehungstypen innerhalb der Datenbank eindeutig repräsentieren. In das Feld bezeichnung wird die ausgeschriebene Bedeutung einer Beziehung eingetragen, zum Beispiel “Schlagwort zu”, “Oberbegriff zu” oder “Plural von”. Die Datenbank erzeugt dazu automatisch den numerischen Identifier. Es können beliebig viele Beziehungstypen eingetragen werden und es können jederzeit neue Typen hinzugefügt oder alte gelöscht werden[4].

[4]Beim Löschen sind dann allerdings auch alle Beziehungen dieses Typs im Registersystem verloren

Feld	Typ	Bedeutung
typ_id	int, auto_increment	eindeutiger, ganzzahliger Identifier für einen Bezihungstyp, wird automatisch fortlaufend vergeben und dient der eindeutigen Identifizierung von Beziehungstypen innerhalb der Datenbank
bezeichnung	text	alphanumerische Beschreibung der Beziehung
Primärschlüssel: typ_id		

Tabelle 2.5: Tabelle *typen*

Die Tabelle *bereiche* (2.6) definiert die Bereiche eines Angebots, denen Ressourcen zugeordnet werden können. Auch jeder Bereich wird innerhalb der Datenbank durch einen eindeutigen, ganzzahligen Identifier dargestellt. Das Feld bereich nimmt die textuelle Beschreibung des Bereichs auf, zum Beispiel “Briefe”, “Personen” oder “häufig gestellte Fragen”. Der zugehörige Identifier im Feld be_id wird dazu von der Datenbank erzeugt.

Feld	Typ	Bedeutung
be_id	int, auto_increment	eindeutiger, ganzzahliger Identifier für einen Bereich, wird automatisch fortlaufend vergeben und dient der eindeutigen Identifizierung eines Bereichs innerhalb der Datenbank
bereich	text	alphanumerische Beschreibung des Bereichs
Fortsetzung $\longrightarrow$		

Fortsetzung Tabelle 2.6
Primärschlüssel: be_id

Tabelle 2.6: Tabelle *bereiche*

Diese Datenbankstruktur erlaubt die Beschreibung von Bezeichnungen, Ressourcen und Beziehungen zwischen diesen. Abfragen können durch die Verknüpfungen der Tabellen über Fremdschlüssel eingeschränkt durch beliebigen Eigenschaften erfolgen. Es kann insbesondere gezielt nach Bezeichnungen und Ressourcen, die durch bestimmte Beziehungstypen mit anderen Bezeichnungen oder Ressourcen verbunden sind, gesucht werden.

2.3 Datenbankschnittstellen

Der Zugriff auf die Datenbank geschieht über Benutzerschnittstellen, die mit Bedienoberflächen in HTML ausgestattet sind. Die Benutzerschnittstellen lassen sich in Abfragefunktionen und Verwaltungsfunktionen einteilen. Abfragefunktionen sind die Schnittstellen für Endbenutzer, die mit Hilfe des Registersystems Ressourcen in einem Online-Angebot suchen. Verwaltungsfunktionen dienen der Verwaltung des Registersystems, zum Beispiel dem Hinzufügen neuer Ressourcen, Bezeichnungen oder Beziehungen.

2.3.1 Abfragefunktionen

Der Benutzer soll die Möglichkeit haben, durch Anfragen an das Registersystem Ressourcen eines Online-Angebots zu finden. Diese Anfragen sind für das System immer Datenbankabfragen, die in der Sprache SQL gestellt werden. Der Zugriff durch den Benutzer soll aber über das WWW mit einer komfortablen HTML-Oberfläche geschehen. Auf dem HTTP-Server sind also Funktionen notwendig, die die Benutzerwünsche in SQL umsetzen, an die

Datenbank weiterreichen und die Ergebnisse aus der Datenbank wieder in HTML formatiert an den Benutzer zurück liefern.

Diese Funktionen werden in einem Programm in der Programmiersprache PHP 4 implementiert, das die Eingaben des Benutzers entgegennimmt und daraus SQL-Abfragen für die Datenbank generiert. Die Anfragen des Benutzers können sehr vielfältig sein, prinzipiell soll jede Abfrage, die in SQL an die Datenbank gestellt werden kann, auch über eine HTML-Oberfläche erstellt werden. Zur Gestaltung der Oberfläche stehen dabei die üblichen Möglichkeiten von HTML zur Gestaltung von interaktiven Benutzeroberflächen zur Verfügung, also insbesondere Formularelemente wie Texteingabefelder, Auswahllisten, Checkboxen usw., aber auch Links, die Parameter an einen HTTP-Server übergeben können[5]. Die Abfragen werden dabei durch folgende Parameter bestimmt:

- Einschränkung nach Eigenschaften von Bezeichnungen
- Einschränkung nach Eigenschaften von Ressourcen
- Einschränkung nach Eigenschaften von Beziehungen

Damit die Gestaltung der Benutzeroberfläche möglichst flexibel und angepasst an die jeweiligen Erfordernisse des Online-Angebots erfolgen kann, sollte ein PHP-Programm auf dem HTTP-Server alle erforderlichen SQL-Abfragen generieren können. Dazu sollte das Programm einige Standardabfragen bereit halten, die durch die Parameter, die von der Benutzeroberfläche übergeben werden, verfeinert und erweitert werden können.

Die Ausgaben der Datenbank werden von dem PHP-Programm entgegengenommen und können mit allen in HTML zur Verfügung stehenden Gestaltungsmöglichkeiten formatiert ausgegeben werden, insbesondere in Listen- oder Tabellenform.

[5] Zur Parameterübergabe bei HTTP-Requests mittels der Methoden `GET` oder `POST` vgl. (Fielding u. a. 1999, Kapitel 9.3 und 9.5)

2.3.2 Verwaltung des Systems

Zur Verwaltung des Registersystems gehören folgende Aufgaben:

- Hinzufügen/Löschen/Ändern von Ressourcen
- Hinzufügen/Löschen/Ändern von Bezeichnungen
- Hinzufügen/Löschen/Ändern von Beziehungen
- Hinzufügen/Löschen/Ändern von Beziehungstypen
- Hinzufügen/Löschen/Ändern von Bereichen

Das sind alles Veränderungen an den Inhalten der Datenbank, zum Teil in einzelnen Tabellen, zum Teil in mehreren, so dass auch hier wieder HTML-Oberflächen für den Benutzer zur Verfügung stehen, die über ein Programm in PHP 4 mit der Datenbank kommunizieren.

Die Verwaltung der Bezeichnungen und Ressourcen und Beziehungen zwischen diesen kann zum Teil über automatische Importfilter für Ressourcen geschehen. Diese Importfilter können aus XML-Daten, die als Ressourcen aufgenommen werden, einzelne Elemente für die Aufnahme als Bezeichnung in das Register auswerten. Aus dem Elementtyp ergibt sich die Beziehung der Bezeichnung zu der Ressource. Aus einem Element `<schlagwort>` mit dem Inhalt `Religion` lässt sich eine Bezeichnung *Religion* mit der Beziehung *ist Schlagwort zu* zu der Ressource, aus der das Schlagwort stammt, herstellen. Die Importfilter müssen jedoch für jeden einzelnen XML-Datentyp angepasst werden.

Zusätzlich muss es aber die Möglichkeit der manuellen Pflege von Ressourcen, Bezeichnungen und Beziehungen durch einen Verwalter des Systems geben, da nicht alle Bezeichnungen aus XML-Daten stammen und insbesondere Beziehungen zwischen Bezeichnungen nicht automatisch hergestellt werden können. Dazu wird es eine HTML-Oberfläche geben, die es erlaubt eine neue Bezeichnung und Ressourcen einzugeben oder aus einer Liste der schon erfassten Bezeichnungen und Ressourcen eine auszuwählen. Zu dieser Bezeichnung

oder Ressource werden dann alle von ihr ausgehenden und auf sie verweisende Beziehungen angezeigt. Durch Auswahl einer Beziehung kann diese bearbeitet werden (neuer Typ zuweisen oder löschen). Zusätzlich kann durch Auswahl einer weiteren Bezeichnung oder einer Ressource eine neue Beziehung zu dieser erstellt werden.

Zum Pflegen der Beziehungstypen und Bereiche genügt eine einfache HTML-Schnittstelle, die das Ändern und Löschen von einzelnen Einträgen aus den vorhandenen Daten der Tabellen *typen* oder *bereiche* erlaubt. Zum Anlegen neuer Typen oder Bereiche reicht eine Möglichkeit zur Eingabe der Bezeichnung des neuen Typs oder Bereichs.

Literaturverzeichnis

[ALI 2000] The Apache Software Foundation: *The Apache Software License, Version 1.1*. 2000. – online unter `http://www.apache.org/LICENSE.txt` (23.4.2001)

[PLI 2000] The PHP group: *The PHP license FAQ*. 2000. – online unter `http://www.php.net/license/#FAQ` (23.4.2001)

[MPE 2001] MySQL AB, Sweden: *MySQL Benchmarks*. 2001. – online unter `http://www.mysql.com/information/benchmarks.html` (23.4.2001), es handelt sich hierbei um einen Banchmarktest des Herstellers der Software, allerdings ist der Prozess zur Ermittlung der Ergbnisse offengelegt und nachvollziehbar.

[MLI 2001] MySQL AB, Sweden: *MySQL Licensing Policy*. 2001. – online unter `http://www.mysql.com/support/arrangements/policy.html` (23.4.2001)

[Barners-Lee u. a. 1998] BARNERS-LEE, T. ; FIELDING, R. ; MASINTER, L.: RFC 2396: Uniform Resource Indentifiers (URI): Generic Syntax. (August 1998). – online unter `ftp://ftp.isi.edu/in-notes/rfc2396.txt` (23.4.2001)

[Behme 2001] BEHME, Hening: XSLT-Tutorial - Teile 1 bis 3. In: *iX Magazin für professionelle Informationstechnik* (2001), Januar bis März, Nr. 1/2001 bis 3/2001, S. 167 (1/2001), 142 (2/2001), 167 (3/2001). – online unter `http://www.heise.de/ix/artikel/2001/01/167/`

[Bergmann 2001] BERGMANN, Sebastian: XML-Daten mit PHP verarbeiten. In: *iX Magazin für professionelle Informationstechnik* (2001), April, Nr. 4/2001, S. 201. – online unter `http://www.heise.de/ix/artikel/2001/04/201/` (23.4.2001)

[Fielding u. a. 1999] FIELDING, R. ; GETTYS, J. ; MOGUL, J. ; FRYSTYK, H. ; MASINTER, L. ; LEACH, P. ; BARNERS-LEE, T.: RFC 2616: Hypertext Transfer Protocol – HTTP/1.1. (Juni 1999). – online unter `ftp://ftp.isi.edu/in-notes/rfc2616.txt` (23.4.2001)

[Mintert und Menge 2000] MINTERT, Stefan ; MENGE, Rainald: XML verpuppt - Aufbereitung mit Cocoon und Extensible Server Pages. In: *c't Magazin für Computertechnik* (2000), Nr. 10/2000, S. 222

[W3C 2000] W3C (The World Wide Web Consortium): *Frequently asked questions about the extensible markup language, version 1.6 (21 July 2000)*. Juni 2000. – online unter `http://www.ucc.ie/xml/` (23.4.2001)