

Albert-Ludwigs-Universität Freiburg
Institut für Informatik

Lehrstuhl für Datenbanken und Informationssysteme

Prof. Georg Lausen



Diplomarbeit
Entwicklung des Webportals
www.edeju-atlas.de

David Krause

19. März 2004

Ich erkläre hiermit, diese Arbeit selbständig verfasst und keine anderen als die hier angegebenen Quellen und Hilfsmittel verwendet zu haben.

Freiburg, den 19. März 2004

David Krause

Inhaltsverzeichnis

1	Einleitung	1
2	Anforderungsanalyse	3
2.1	Struktur	3
2.1.1	Bereiche	3
2.1.2	Lernmittel	5
2.1.3	Benutzer	5
2.1.4	Altersgruppen	5
2.2	Aufgaben	5
2.3	Probleme	8
2.4	Vergleichbare Systeme	8
3	Definitionsphase	11
3.1	Entity-Relationship Modell	11
3.1.1	Kompetenzträger	11
3.1.2	Bereiche, Lernmittel und Altersgruppen	13
3.1.3	Profil und Projekte	14
3.2	Rechte- und Benutzerverwaltung	15
3.3	Geschäftsprozesse	16
3.3.1	Verwaltung	16
3.3.2	Rechte	18
3.3.3	Mitglieder-Bereich	18
3.3.4	Öffentlicher Bereich	18
4	Entwurfsphase	21
4.1	Grundsatzentscheidungen	21
4.1.1	Architektur	21
4.1.2	Datenbank	23
4.2	Infrastruktur	23
4.3	Datenbankschema	24
4.4	Rechte- und Benutzerverwaltung	25
4.5	Bereiche	26
4.5.1	Modelle zur Abspeicherung von Baumstrukturen in Datenbanken	27
4.5.2	Vorstellung des Nested Sets Modell	28
4.5.3	Tabellenstruktur der Bereiche bei EDEJU	36

4.6	Datenverwaltung	37
4.6.1	Verwaltung	37
4.6.2	Benutzer	38
4.7	Funktionen im öffentlichen Bereich	39
4.7.1	Allgemeine Funktionen	40
4.7.2	Vorschläge	41
4.7.3	Vergleiche	43
4.7.4	Suche	43
4.8	Design von EDEJU	44
5	Implementierung	47
5.1	Struktureller Aufbau	47
5.1.1	Datenbank	47
5.1.2	Ordnerstruktur	48
5.1.3	Dateiaufbau	49
5.1.4	Implementierungsschritte	52
5.2	Implementierungsbeispiele	53
5.2.1	Bilder Upload	53
5.2.2	Vorschläge	54
5.2.3	Vergleiche	55
6	Erweiterungen	59
6.1	Datenbank Upgrade	59
6.2	Abfragekombinationen	60
6.3	Darstellungsformen	60
6.4	Fremdsprachen	61
6.5	Level	62
6.6	Hilfesystem bzw. Erste Schritte	62
6.7	Sonstige Erweiterungen	62
7	Zusammenfassung	65
A	Datenbank - Schema	67
B	Geschäftsprozesse	71
C	Ordnerstruktur & Dateien	73
D	Rechte	77
	Literaturverzeichnis	79

Kapitel 1

Einleitung

Die Neugier auf neue Dinge und Erfahrungen treibt den Menschen dazu an sich weiterzuentwickeln. In der Schule kann und wird diese Neugier meist nicht ausreichend beachtet und gefördert. Häufig ist es nicht möglich, genauer auf die Bedürfnisse einzelner Kinder einzugehen, sei es aus Zeitmangel oder aufgrund der Starrheit des Schulsystems. Viele Fragen bleiben daher unbeantwortet und potentielle Fähigkeiten werden nicht ausgebaut und verkümmern im schlimmsten Fall. Die Pisastudie hat gezeigt, dass das Schulsystem in Deutschland hinter dem System anderer Länder steht. Daher ist es notwendig Kinder auch ausserhalb der Schule zu fördern und die potentiellen Fähigkeiten zu erkennen und zu fördern. Problematisch ist, dass Eltern in vielen Fällen nicht in der Lage sind, die Neugier ihrer Kinder in bestimmten Bereichen zu wecken und weiterzuentwickeln. Einfache Beispiele sind Elektronik, Computer und Technik, wenn ein Kind z.B. "etwas elektronisches" selber bauen will, sind die meisten Eltern überfordert, weil sie selbst keinerlei Wissen bzw. Erfahrungen in diesem Bereich haben. Diese Aufgaben sollten dann Dritte übernehmen, die den Kindern helfen sich in dem relevanten Bereich weiterzuentwickeln. Dem Menschen sollte jederzeit die Möglichkeit geboten werden, je nach Interesse Einblick in die reale Lebenswelt zu bekommen.

An diesem Punkt setzt diese Arbeit an. Es wird ein universelles Webportal entwickelt, welches den Selbstentwicklungsprozess von Kindern und Erwachsenen unterstützen soll. Das Webportal soll alle Bereiche des Lebens enthalten und strukturiert gliedern, ausgehend von 4 Bereichen Natur, Organisation, Technik und Kultur. So könnte Kultur sich z.B. gliedern in Musik, Sprache und Literatur, Musik in klassische, Pop und Techno Musik. Die Idee dabei ist, dass der Benutzer z.B. Interesse an Musik hat und aufgrund der Gliederung von Musik, Musikstile findet, die er noch nicht kannte und sein Interesse und Wissen dadurch erweitern kann. Das Portal soll Vorschläge machen, den Benutzer mit anderen Personen in Kontakt bringen, eine Plattform für den Austausch von Erfahrungen sein und dadurch den Entwicklungsprozess unterstützen. Dabei soll das Portal nicht nur für Kinder, sondern für alle Altersgruppen geeignet sein.

Die Idee für diese Arbeit stammt von Herrn Helmeth. Er ist Gründer des Vereins EDEJU¹.

Diese Arbeit hat das Ziel ein Grundgerüst für das EDEJU-System zu definieren, zu ent-

¹"Internationales Institut zur Entwicklungsförderung der Jugend e.V." siehe <http://www.edeju.org/>

wickeln und zu implementieren. In Kapitel 2 wird in einer Anforderungsanalyse ermittelt und beschrieben, was das System können und leisten soll. Es werden dabei die Struktur, die Aufgaben und die Probleme genauer betrachtet. In Kapitel 3 wird das System genauer spezifiziert, ein Entity-Relationship Modell erstellt und die benötigten Geschäftsprozesse konkretisiert, zusätzlich wird eine Rechte- und Benutzerverwaltung entworfen. In Kapitel 4 wird die Datenbank entworfen und verschiedene Grundsatzentscheidungen, über die Infrastruktur, die Strukturierung von Bereichen und das Design getroffen. In Kapitel 5 wird auf ausgewählte Teile und Probleme der Implementierung eingegangen. In Kapitel 6 wird ein Ausblick auf mögliche Erweiterungen des Systems gemacht, die nicht im Rahmen dieser Diplomarbeit realisiert wurden. Desweiteren werden Vorschläge erarbeitet, wie diese Erweiterungen realisiert werden können. Kapitel 7 schließt diese Arbeit mit einer Zusammenfassung der wichtigsten Punkte ab. Das implementierte System ist unter <http://www.edeju-atlas.de/de/> zu finden.

Kapitel 2

Anforderungsanalyse

Die möglichst volle Entfaltung des gesamten individuellen Anlagenspektrums des Menschen ist das Hauptziel des EDEJU - Atlas, im Folgenden kurz EDEJU. Der Einzelne soll sich selbst und mit Hilfe von EDEJU und anderen Menschen, die sich an EDEJU beteiligen autodidaktisch weiterentwickeln. Anhand von thematisierten Bereichen, die strukturiert dargestellt werden, soll die Neugier des Benutzers auf Neues geweckt und die Vertiefung von schon bekannten Themen gefördert werden. Jeder Mensch kommt mit verschiedenen Begabungen bzw. Potentialen auf die Welt. EDEJU soll helfen diese Begabungen und Potentiale zu entdecken und weiterzuentwickeln. EDEJU soll im Idealfall von der Kindheit an ein ständiger Begleiter in der Entwicklung der Menschen bis zum Lebensende sein.

Da EDEJU kein statisches Nachschlagewerk sein soll, müssen Kinder und Erwachsene EDEJU nutzen und ständig weiterentwickeln. Ein Ziel, das natürlich nie erreicht werden kann, ist die "Abbildung des Weltwissens auf eine Datenbank"¹.

In diesem Kapitel wird EDEJU genauer beschrieben, zunächst werden die gebrauchten Begriffe, die Struktur und die Aufgaben von EDEJU genauer erläutert. Diese wurden in Gesprächen mit Herrn Helmeth erstellt und zusammengetragen. Jetzt schon ersichtliche Probleme und vergleichbare Systeme werden am Ende dieses Kapitels genauer beschrieben.

2.1 Struktur

2.1.1 Bereiche

Die Struktur von EDEJU stützt sich auf *Lebensbereiche*, im weiteren kurz Bereiche genannt. Ein Bereich ist im Prinzip immer ein Thema, welches sich auf etwas bezieht. Z.B. kann das Auto, Astronomie, Informatik oder Frisuren ein Bereich sein. Alles, was der Mensch macht, betrachtet, erforscht oder denkt, kann ein Bereich sein.

Diese Bereiche sollen strukturiert werden und ausgehend von wenigen Bereichen breit

¹Zitat Herr Helmeth

gefächert, ausgebaut und erweitert werden. Herrn Helmeth schlägt für die Ausgangsbereiche *Natur*, *Technik*, *Organisation* und *Kultur* vor.

Diese Anfangsbereiche sind ausgehend von der Idee folgendermaßen gewählt: "am Anfang von allem Leben steht immer die *Natur* und um besser und komfortabler leben zu können, hat der Mensch wie auch Tiere und Pflanzen *Techniken* entwickelt, die *Natur* zu manipulieren. Um harmonisch miteinander zu leben und sich stetig weiterentwickeln zu können, ist die *Organisation* der Lebensbereiche notwendig. Schließlich entwickelt sich dabei eine *Kultur*, die das Leben sinnvoll und lebenswert erscheinen lässt."²

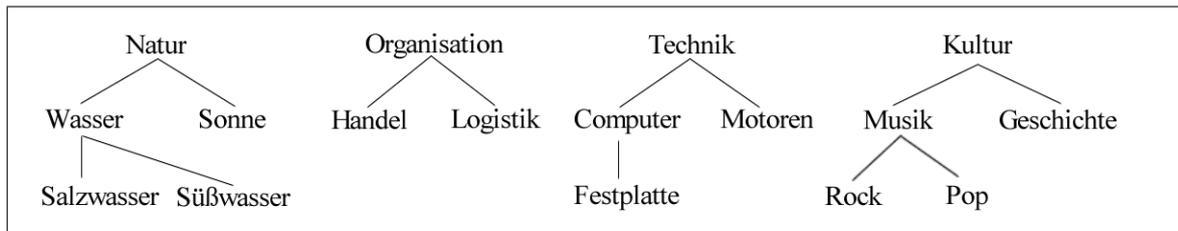


Abbildung 2.1: Beispiel Struktur der Bereiche

In der Abbildung 2.1 ist ein kleines Beispiel einer solchen Struktur skizziert. Die Bereiche sind gegliedert in die 4 vorgeschlagenen Bereiche *Natur*, *Organisation*, *Technik* und *Kultur*. Diese 4 Bereiche werden baumartig weiter aufgegliedert in Unterbereiche. Beispielsweise wird *Natur* aufgegliedert in *Wasser* und *Sonne*. *Wasser* wird weiter aufgegliedert in *Salzwasser* und *Süßwasser*.

Es ist sofort ersichtlich, dass es nicht möglich ist, alle vorhandenen Bereiche zu erfassen. Daher ist es notwendig, dass diese Strukturierung laufend immer wieder erweitert und neu angeordnet werden kann.

Diese Strukturierung ist komplex: man hat in vielen Fällen mit Bereichen zu tun, die mehreren anderen Bereichen zugeordnet werden können. So könnte z.B. ein Baum sowohl der *Natur* zugeordnet werden wie auch der *Religion*, da z.B. die Kelten Bäume verehrt haben. Es ist daher notwendig für diese Mehrdeutigkeiten eine befriedigende Lösung zu finden.

Nachdem eine solche Struktur erstellt wurde, hat man eine Art Atlas, daher EDEJU - Atlas, in dem die Bereiche nachgeschlagen werden können. So könnte man z.B. beim Bereich *Bäume* nachlesen, was die Gattung Baum charakterisiert und wie diese wachsen. Der Bereich *Bäume* könnte dann weiter aufgegliedert sein in die verschiedenen Arten von Bäumen Nadelbäume und Laubbäume. Aufgrund dieser Informationen hätte man die Möglichkeit mehr über einen Bereich zu lernen und erfahren. Durch die Verknüpfung mit anderen Bereichen könnte der Horizont erweitert und der Zusammenhang zwischen verschiedenen Dingen besser verstanden werden.

Diese Struktur und Informationen müssen anschließend um sogenannte Lernmittel erweitert werden.

²nach Herrn Helmeth

2.1.2 Lernmittel

Der Zweck eines Lernmittels ist es, dem Benutzer die Möglichkeit zu geben mehr über einen bestimmten Bereich zu lernen und praktische Erfahrungen zu sammeln. Lernmittel werden Bereichen zugeordnet, zu denen sie einen Bezug haben. Ein Computer wird z.B. den Bereichen Informatik, Informationstechnik, Internet usw. zugeordnet, da das Lernmittel Computer mit diesen Bereichen zusammenhängt. Zu beachten ist dabei, dass es nicht nur das Lernmittel Computer geben kann, sondern auch einen Bereich Computer. Dort wird dann z.B. die Geschichte des Computers und der Aufbau eines Rechners beschrieben. Ein Lernmittel muss dabei nicht zwangsläufig ein Objekt sein, es kann auch eine Internetseite, ein Artikel in der Zeitung oder ein Volkshochschulkurs sein. Daher ist darauf zu achten, dass es möglich sein soll, Lernmittel in EDEJU einzutragen, die von der Art bzw. Form völlig unterschiedlich sind.

Die Strukturierung der Bereiche und der Zuordnung der Lernmittel zu den Bereichen bildet den Kern von EDEJU. Ausgehend von diesem Kern werden nun Erweiterungen hinzugefügt und Sinn und Zweck von EDEJU genauer beschrieben. Zunächst wird jedoch der Begriff eines Benutzers genauer definiert.

2.1.3 Benutzer

Ein Benutzer ist ein Mensch oder eine Institution, der mit EDEJU arbeitet bzw. EDEJU nutzt. Einem Benutzer von EDEJU muss es möglich sein ein Profil (siehe 2.2) zu erstellen, Informationen zu nutzen, die bei EDEJU vorhanden sind und in Kontakt mit anderen Benutzern zu treten. Neben diesen Einzelpersonen sind auch noch andere Arten von Benutzern denkbar z.B. Firmen oder Vereine. Diese könnten sich bei EDEJU darstellen und dabei helfen EDEJU zu erweitern, indem sie z.B. Lernmittel anbieten oder sich am Ausbau der Datenbasis beteiligen.

2.1.4 Altersgruppen

Zusätzlich zur Strukturierung der Bereiche sollen Lernmittel und Bereiche noch eine weitere Einteilung erhalten. Beide sollen in Altersgruppen unterteilt werden. Es ist sinnvoll, solch eine Einteilung vorzunehmen, da z.B. Lernmittel nicht für jedes Alter geeignet sind. So kann z.B. für ein Brettspiel die Altersangabe 9 - 99 Jahre lauten. Dabei ist zu beachten, dass dies natürlich immer nur ein Richtwert sein kann.

2.2 Aufgaben

Mithilfe der Bereiche und Lernmittel soll der Benutzer von EDEJU sein eigenes Profil erstellen. Das Profil besteht aus *Interesse*, *Können* und *Wissen*. Der Benutzer soll sich aus den Bereichen und Lernmitteln sein persönliches Profil erstellen. Er wählt aus den Bereichen Gebiete aus, die ihn interessieren, über die er Wissen hat oder in denen er etwas kann.

Ein Profil für einen Informatikstudenten könnte z.B. die Bereiche Mathematik und Computer enthalten, sei es als *Interesse*, *Können* oder *Wissen*. Zudem wird auch das Lernmittel Computer in seinem Profil enthalten sein. Natürlich muss das Profil noch viele weitere Bereiche und Lernmittel enthalten, die nicht nur aus einem Bereich kommen. Das Profil soll einen Menschen so gut wie möglich auf die Bereiche und Lernmittel abbilden.

In der Abbildung 2.2 wird der Zusammenhang der verschiedenen Begriffe skizziert.

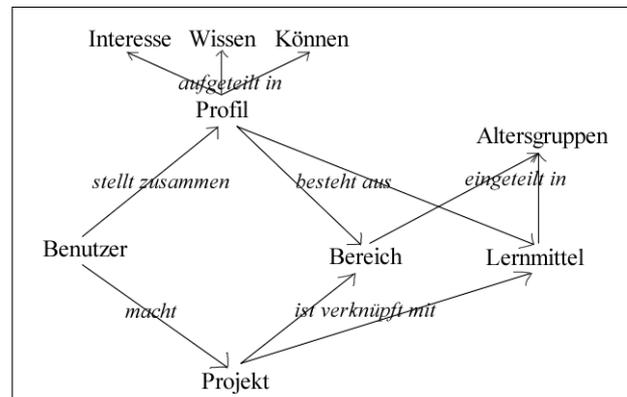


Abbildung 2.2: Zusammenhang der Begriffe

Die Aufteilung nach *Interesse*, *Wissen* und *Können* hat sich im Laufe der Gespräche mit Herrn Helmeth herausgebildet. *Interesse* sollte alle Bereiche bzw. Lernmittel enthalten, die den Benutzer interessieren und über die er mehr erfahren möchte. Die Unterscheidung zwischen *Wissen* und *Können* ist etwas schwieriger. So kann ein Benutzer ein Auto fahren, aber er weiß nicht, warum und wie das Auto funktioniert. Oder der Benutzer weiß, wie ein Auto aufgebaut ist und wie die Teile funktionieren, aber er kann es nicht selbst zusammenbauen, weil ihm z.B. das handwerkliche Geschick fehlt.

Das Ziel des Profils ist es, den Benutzer auf die Bereiche und Lernmittel abzubilden. Wenn nun der Benutzer all seine *Interessen*, sein *Wissen* und sein *Können* eingetragen bzw. ausgewählt hat, soll er sich mit Hilfe von EDEJU autodidaktisch weiterentwickeln können. Hierzu gibt es mehrere Wege. Zum einen sollte der Benutzer ausgehend von einem Bereich, für den er sich interessiert, die Unterbereiche oder Lernmittel dieses Bereichs näher betrachten und dabei noch andere Seiten dieses Bereichs kennen lernen können, an die er noch nie gedacht hat. Der Benutzer interessiert sich z.B. für den Bereich Astronomie: er soll nun die Möglichkeit haben ausgehend vom Bereich Astronomie, Unterbereiche oder Lernmittel zu betrachten. Dort könnte er unter "Lernmittel" vielleicht eine Internetseite finden, die sich ausführlich mit dem Thema Astronomie beschäftigt oder er könnte einen Unterbereich finden, wie z.B. Hintergrundstrahlung im All. Auf diese Weise soll der Benutzer sein *Wissen*, *Können* und seine *Interessen* weiter ausbauen. Die breite Auffächerung der Bereiche hilft dabei, die Neugier zu wecken und bisher unbekannte Wege zu gehen.

Ein anderer Weg wäre es, sich vom System Vorschläge anhand des eigenen Profils geben zu lassen. Dabei sind 3 Vorschläge zu unterscheiden. Die erste Art von Vorschlägen wären *Ähnliche Vorschläge*, d.h. solche die den *Interessen*, *Wissen* und *Können* des Benutzers entsprechen.

Die andere Art von Vorschlägen wären *Weißer Flecken*, d.h. Gebiete innerhalb der Struktur der Bereiche die noch nicht durch das Profil abgedeckt sind. Dies könnte dem Benutzer die Möglichkeit geben, "über den Tellerrand zu blicken" und ihm helfen, sich nicht nur in eine Richtung z.B. Technik weiterzuentwickeln, sondern ihn auch dazu animieren sich in eine andere Richtung, die bisher vernachlässigt wurde (z.B. Kultur) zu entwickeln.

Eine vom Profil unabhängige Art Vorschläge zu machen wären *Willkürlichen Vorschläge*, die randomisiert mehrere Bereiche auswählen und vorschlagen.

Personen zusammenzubringen, die die gleichen Interessen haben, ist ein weiteres Ziel von EDEJU. Dies sollte auf 2 Arten geschehen: Zum einen sollte nach Personen gesucht werden können, die bestimmte oder gleiche Interessen haben. Zum anderen sollte jeder Benutzer die Möglichkeit haben, Projekte innerhalb von EDEJU einzutragen und diese mit Bereichen und Lernmitteln zu verknüpfen. So sollte ein Benutzer innerhalb eines Bereiches auch die Projekte betrachten, die mit diesem Bereich verknüpft sind. Falls er sich für dieses Projekt interessiert, soll es möglich sein mit dem Benutzer, der dieses Projekt eingetragen hat, Kontakt aufzunehmen. Was ein Projekt sein kann, ist unbestimmt. Ein Projekt könnte ein Wochenendausflug mit dem Fahrrad genauso wie ein Volkshochschulkurs sein.

Innerhalb von EDEJU soll es möglich sein, dass ein Benutzer anderen Benutzern Lernmittel anbietet, sei es zum Leihen oder Kaufen. Ein Lernmittel z.B. eine Drechselmaschine ist nicht für jeden verfügbar, durch die Eintragung dieser Maschine wäre es möglich, mit dem Besitzer Kontakt aufzunehmen und diese vielleicht auszuleihen und auszuprobieren.

Durch Vergleiche mit Profilen wäre es möglich, sich z.B. mit Berufsgruppen zu vergleichen. Da Berufsgruppen wie Bäcker, Informatiker usw. ähnliche *Interessen*, *Können* und *Wissen* haben, soll der Benutzer das eigene Profil mit dem einer Berufsgruppe vergleichen können. Dadurch könnte er Übereinstimmungen oder Unterschiede herausfinden und untersuchen, ob vielleicht dieser Beruf zu ihm passen würde. Auch ist es denkbar, sich mit einer bestimmten Kultur zu vergleichen. Mithilfe eines solchen Vergleichs könnte überprüft werden, ob der Benutzer in ein bestimmtes "Klischee" von der Kultur eines Landes bzw. einer Region passt. Diese Vergleiche sollen es dem Benutzer möglich machen, sich mit anderen Personen zu vergleichen. Diese Vergleiche könnten dazu beitragen, dass man z.B. einem Vorbild ähnlicher wird oder sieht ob irgendwelche Gemeinsamkeiten bestehen.

Um dem Benutzer die Suche nach bestimmten Bereichen oder Lernmitteln zu erleichtern ist es notwendig eine geeignete Suchfunktion aufzubauen, die es dem Benutzer ermöglichen soll sich innerhalb von EDEJU zielgenau zu bewegen.

Es ist ersichtlich, dass EDEJU eine Vielzahl von Verwaltungsaufgaben unterstützen muss. Angefangen von der Benutzerverwaltung hin zu einem Redaktionssystem für Bereiche, Lernmittel, Projekte, Zusätzlich müsste die Struktur von EDEJU flexibel genug sein für Erweiterungen, da in Zukunft mit Sicherheit weitere Ideen entstehen werden, die in EDEJU integriert werden sollen. Da die Benutzer bei der Verwaltung unterschiedliche Aufgaben haben sollen ist es notwendig eine Rechteverwaltung zu entwickeln, die flexibel genug auf Veränderungen bzw. Erweiterungen von EDEJU reagieren kann.

2.3 Probleme

Ein Problem bei der Entwicklung des Grundgerüsts von EDEJU ist, dass die Struktur der Bereiche erst wachsen wird, wenn das System einsatzbereit ist. Es gibt keine Bereiche bzw. Lernmittel, die wie in 2.1 beschrieben, bereits strukturiert sind. Daher ist es notwendig künftig die Struktur der Daten so flexibel wie möglich zu gestalten, um auf eventuelle Veränderungen vorbereitet zu sein. Die Ermittlung und Eingabe der Daten in EDEJU wurde, nachdem eine geeignete Struktur implementiert wurde, von Herrn Helmeth vorgenommen. Allerdings müssen die Daten zukünftig noch erweitert und mit mehr Informationen angereichert werden.

Ein weiteres Problem ist, dass EDEJU ein lebendiges dynamisches System sein soll. Dies bedeutet, dass es immer wieder erweitert und verändert werden kann. Es ist ersichtlich, dass ein solches System nie "fertig" sein wird. EDEJU wurde nach dem evolutionären Modell³ entwickelt. Zuerst wurden die Kernanforderungen an das System realisiert. Während der ganzen Entwicklung wurden dann durch Feedback und ausprobieren immer neue Ideen und Erweiterungen zu EDEJU gefunden. Nachteil bei diesem Modell ist, dass durch neue Ideen und gewünschte Erweiterungen bisherige Lösungen überflüssig werden bzw. ganz oder teilweise ersetzt werden müssen. Da aber am Anfang der Entwicklung noch nicht feststand und am Ende dieser Arbeit nicht feststehen wird, welche Erweiterungen noch kommen, ist die evolutionäre Entwicklung die einzig mögliche. Ersichtlich ist, dass dieser Prozess iterativ ist, da die verschiedenen Entwicklungsstufen für partielle Anforderungen mehrmals durchlaufen werden müssen.

2.4 Vergleichbare Systeme

In diesem Abschnitt werden zwei vergleichbare im Internet vorhandene Systeme und deren Unterschiede zu EDEJU dargestellt.

Planet-Wissen⁴ bietet mit seinem *Wissens-Planetarium* einen ähnlichen Ansatz, wie EDEJU. Die Bereiche werden dort Themen genannt und sind durch eine Netzstruktur miteinander verbunden. Dem Benutzer werden dort zu jedem Thema Informationen angeboten.

³ausführlicher dazu H.Balzert (2001) S.56

⁴<http://www.planet-wissen.de/>

Wikipedia⁵ ist eine freie Enzyklopädie im Internet, in der jeder sein Wissen mit anderen teilen kann. Jeder Benutzer hat die Möglichkeit als Redakteur oder Autor Beiträge einzutragen bzw. vorhandene zu bearbeiten.

Der Unterschied der beiden Systemen zu EDEJU liegt darin, dass keine Möglichkeit besteht sich ein Profil zu erstellen. Zusätzlich sind die Themen nicht strikt in Kategorien eingeteilt, wie bei EDEJU, wo die Bereiche in die 4 Anfangsbereiche eingeteilt sind. Ähnlichkeiten bestehen zu Wikipedia darin, dass jeder Benutzer Beiträge, die das System weiterentwickeln, beisteuern kann. Planet-Wissen ist durch seine Vernetzung ähnlicher Themen den Bereichen bei EDEJU sehr ähnlich. Der Hauptunterschied zu bestehenden Systemen ist, dass bei EDEJU der Mensch und dessen Weiterentwicklung im Mittelpunkt steht. Beide vorgestellten Systeme dienen hauptsächlich als Nachschlagewerke. Neu an EDEJU ist, dass versucht wird dem Benutzer Möglichkeiten aufzuzeigen, sich weiterzuentwickeln, z.B. durch Vergleiche und Vorschläge.

⁵<http://de.wikipedia.org/>

Kapitel 3

Definitionsphase

In diesem Kapitel wird die in der Anforderungsanalyse erstellte Struktur konkretisiert und die Aufgaben von EDEJU genauer beschrieben. Da der Hauptbestandteil dieser Arbeit die Erstellung einer geeigneten Datenbank ist, wird zunächst Schritt für Schritt ein Entity-Relationship-Modell¹, kurz ER-Modell, erstellt, in dem die Daten und Beziehungen zwischen den Daten beschrieben werden. In einem zweiten Schritt wird eine Benutzerverwaltung erstellt und das Datenmodell um diese erweitert. Im dritten Schritt werden die Geschäftsprozesse von EDEJU konkretisiert.

Um die Grafiken nicht zu groß und unübersichtlich werden zu lassen, wurde auf die Darstellung der Attribute innerhalb der ER-Modelle verzichtet. Diese werden im Text erwähnt und genauer beschrieben.

3.1 Entity-Relationship Modell

Durch die Fülle von Daten, wie Benutzerdaten, Bereiche, Lernmittel, Projekte, usw. die EDEJU zu verwalten hat, ist es notwendig, eine Datenbank zu nutzen, die diese Verwaltung übernimmt. Innerhalb dieses Abschnittes wird Schritt für Schritt ausgehend von den Benutzern ein ER-Modell erstellt, in dem das Datenmodell genauer erläutert und definiert wird.

3.1.1 Kompetenzträger

Ein Benutzer kann eine natürliche Person, eine Firma, eine Institution oder ein Verein sein. Daher ist es notwendig eine Generalisierung des Begriffs Benutzer zu verwenden, die dieser Vielfalt gerecht werden kann. In Abbildung 3.1 wird ein Benutzer aufgeteilt in einen Obertyp "Benutzer" und die Untertypen "Firma", "Verein", "Institution" und "Person".

Ein Benutzer, gleich welcher Art er ist, ist immer auch ein "Kompetenzträger". Dies bedeutet: er hat ein eigenes Profil mit *Interessen*, *Können* und *Wissen* bzgl. Lernmitteln und Bereichen.

¹Vgl. H.Balzert (2001) S.224ff; Vgl. G.Matthiessen/M.Unterstein (2000) S.91ff; Vgl. A.Kemper/A.Eikler (1999) S.33ff

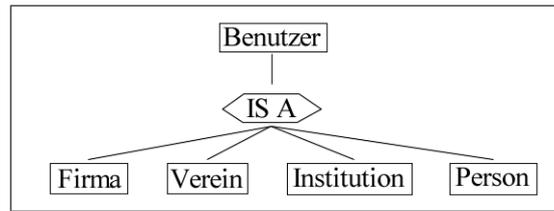


Abbildung 3.1: ER-Modell:Benutzer

Der Benutzer ist allerdings nicht der einzige Kompetenzträger. Berufsgruppen oder Kulturen haben auch ein Profil. Diese werden unter dem Begriff des "Muster-Kompetenzträger" zusammengefasst. Diese werden z.B. für Vergleiche mit dem eigenen Profil benötigt, um herauszufinden, ob ein bestimmter Beruf zum Benutzer passen könnte. Diese beiden Gruppen von Kompetenzträgern sind in Abbildung 3.2 dargestellt.

Innerhalb dieser Abbildung ist eine weitere Beziehung zwischen Benutzer und dem Muster-Kompetenzträger ersichtlich. Ein Muster-Kompetenzträger sollte immer mindestens einen Ansprechpartner unter den Benutzern, für eventuelle Fragen von anderen Benutzern, haben.

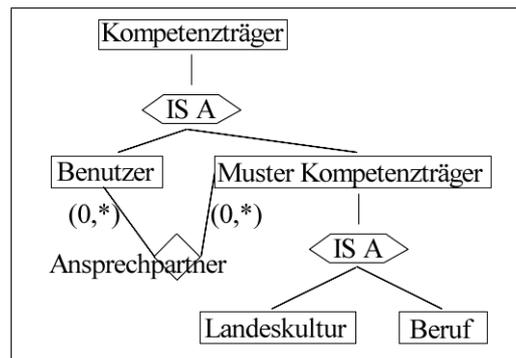


Abbildung 3.2: ER-Modell: Kompetenzträger

Um einen Benutzer zu beschreiben, benötigt man folgende Attribute: *Straße, Postleitzahl, Ort, Land, Telefon, Fax, eMail, Internetseite, Beschreibung* und *Bild*. Die Untertypen des Benutzers benötigen zusätzliche Attribute zu den schon vom Benutzer geerbten. Die Person benötigt noch *Vorname, Nachname, Nationalität, Geburtstag* und *Beruf*. Die anderen Untertypen benötigen noch die Attribute *Bezeichnung*, d.h. z.B. der *Firmen-* oder *Vereinsname* und den *Vor-* und *Nachnamen* des Ansprechpartners des Vereins, der Firma oder der Institution.

Der Muster-Kompetenzträger benötigt um beschrieben zu werden folgende Attribute: *Bezeichnung, Beschreibung, eine Internetseite* und ein *Bild* als Attribute.

Die Beziehung zwischen Benutzer und Muster-Kompetenzträger beinhaltet nur die beiden Schlüssel aus den beteiligten Entitäten. Wie der Schlüssel genau aussieht, wird in Kapitel 4 genauer erläutert.

3.1.2 Bereiche, Lernmittel und Altersgruppen

Der Aufbau der Bereiche (siehe Abbildung 3.3), welche die Kernstruktur von EDEJU bilden, ist eine Baumstruktur, die sich ausgehend von den 4 Basis-Bereichen weiter in Unterbereiche auffächert. Diese Beziehung ist rekursiv aufgebaut. So ist z.B. Elektronik ein Unterbereich von Technik, wobei Elektronik und Technik jeweils eigene Bereiche sind.

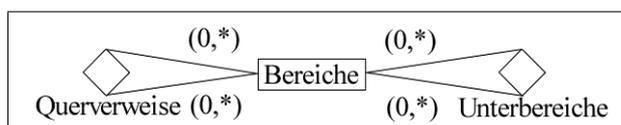


Abbildung 3.3: ER-Modell: Bereiche

Da es oft mehrere Möglichkeiten gibt einen Bereich einem anderen zuzuordnen, gibt es eine weitere Beziehung der Bereiche mit sich selbst, die Querverweise. So ist z.B. Auto ein Unterbereich von Technik verknüpft mit "der Geschichte des Autos", einem Unterbereich von Kultur. Auf diese Weise wird es möglich einen Bereich mit mehreren Bereichen zu verknüpfen.

Ein weiterer Kernpunkt der Struktur von EDEJU sind die Lernmittel. Lernmittel sind mit Bereichen verknüpft. Diese Beziehung zusammen mit der Beziehung zu den Altersgruppen ist in Abbildung 3.4 genauer dargestellt. Ein Lernmittel kann mit beliebig vielen Bereichen verknüpft sein. Dies entspricht der Realität, wo z.B. ein Brettspiel sowohl der Konzentrationfähigkeit, der Kontaktpflege oder der Unterhaltung dienen kann.

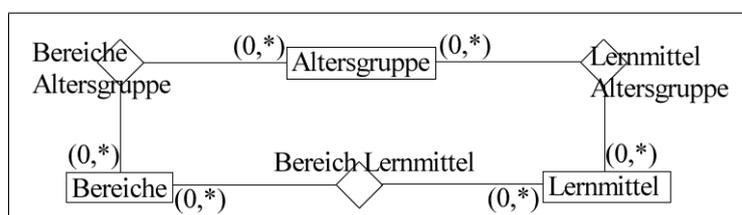


Abbildung 3.4: ER-Modell: Lernmittel - Altersgruppen - Bereiche

Bei der Altersgruppenunterteilung kann ein Lernmittel genauso wie ein Bereich mehreren Altersgruppen zugeordnet werden. Da z.B. aufgrund von unterschiedlichen Begabungen von Kindern die Altersangaben nicht immer zutreffen, sind die Altersgruppen zusätzlich zum Alter mit einer *Beschreibung* zu versehen. Damit klar wird, dass, wenn ein Kind bestimmte Begabungen hat (z.B. im Bereich Technik) dieses Lernmittel schon früher verwendet werden kann. Für andere Benutzer, die nicht technisch begabt sind, eignet sich dieses Lernmittel vielleicht erst später oder nie.

Um einen Bereich genau zu beschreiben, werden folgende Attribute benötigt: *Bezeichnung*, *Beschreibung* und ein *Bild*. Natürlich können in Zukunft weitere Attribute hinzugefügt

werden. Es hat sich im Rahmen dieser Arbeit aber herausgestellt, dass diese 3 Attribute momentan ausreichen. Die Beziehung Querverweise benötigt das Attribut *Beschreibung*. Die Beziehung Unterbereiche benötigt keine weiteren Attribute ausser den Schlüsseln der in Beziehung stehenden Bereiche.

Lernmittel haben *Bezeichnung*, *Beschreibung*, einen *Link* und ein *Bild* als Attribute. Die Beziehung zwischen Lernmitteln und Bereichen hat keine weiteren Attribute als die beiden Schlüssel.

Altersgruppen werden durch Altersangaben (*von-*, *bis-Alter*) und einer *Beschreibung* charakterisiert. Hier wird die Beziehung sowohl zu Bereichen, wie auch zu Lernmitteln nur durch die Schlüssel der Altersgruppe und des Bereichs bzw. des Lernmittels genauer definiert.

3.1.3 Profil und Projekte

Um ein Profil zu erstellen, müssen nun die oben entwickelten ER - Modelle in ein Modell überführt werden und mit den benötigten Beziehungen zu den anderen Entitäten erweitert werden. In Abbildung 3.5 ist ersichtlich, dass ein Kompetenzträger immer *Interessen*, *Können* und *Wissen* in Bezug auf Lernmittel oder Bereiche hat. Diese Beziehungen werden zusätzlich zu den jeweiligen Schlüsseln um eine *Beschreibung* und ein *Bild* erweitert. Mithilfe dieser Beziehungen wird das Profil eines Kompetenzträgers beschrieben. Zwischen Bereiche und dem Untertyp von Kompetenzträger Benutzer gibt es die Beziehung Ansprechpartner. Diese Beziehung dient dazu, Rückfragen zu Bereichen zu unterstützen. Diese Beziehung besteht auch zwischen Lernmitteln und Benutzern, zusätzlich dazu gibt es dort eine weitere Beziehung Angebote. Diese definiert die Möglichkeit des Benutzers, ein Lernmittel innerhalb von EDEJU anzubieten. Das Angebot wird durch die Attribute *Beschreibung* und *Bild* genauer definiert. Dadurch ist es möglich, vielfältige Formen von Angeboten einzutragen. So kann das Lernmittel zum Verleihen, zum Verkaufen oder zum Lernen in EDEJU eingetragen werden.

Zusätzlich zum Profil, den Angeboten und Ansprechpartner-Beziehungen ist in der Abbildung 3.5 noch die Entität Projekte und die dazugehörigen Beziehungen eingezeichnet. Ein Projekt wird immer genau von einem Benutzer (dem Projekt-Initiator) in EDEJU eingetragen und betreut. Zusätzlich soll es möglich sein, dass andere Benutzer an einem Projekt teilhaben und mitmachen können. Auf diese Weise ist es möglich, zwischen Initiator und Benutzer zu unterscheiden und dementsprechend Anfragen an EDEJU zu stellen. Die Attribute der Projekte sind die Folgenden: *Bezeichnung*, *Beschreibung*, *Link*, Veranstaltungsdaten (*von-Datum*, *bis-Datum*), *Ort*, *Land*, *Region* und ein *Bild*. Ein Projekt kann mit mehreren Benutzern in Beziehung stehen, da viele Projekte keine "Ein-Mann-Projekte" sind. Ein Projekt kann beliebig viele Lernmittel benutzen, z.B. könnten im Projekt "Kanu fahren" die Lernmittel Paddel, Spritzdecke, Kanu, usw. enthalten sein. Ausserdem ist es möglich, ein Projekt mehreren Bereichen zuzuordnen, so ist z.B. das Projekt "EDEJU-Atlas erstellen" in den Bereichen Computer, Lernen, Autodidaktik, usw. einzuordnen, da diese Bereiche mit dem Projekt thematisch verbunden sind. Die Attribute dieser Beziehungen beschränken sich auf die jeweiligen Schlüssel und können aber wenn nötig noch

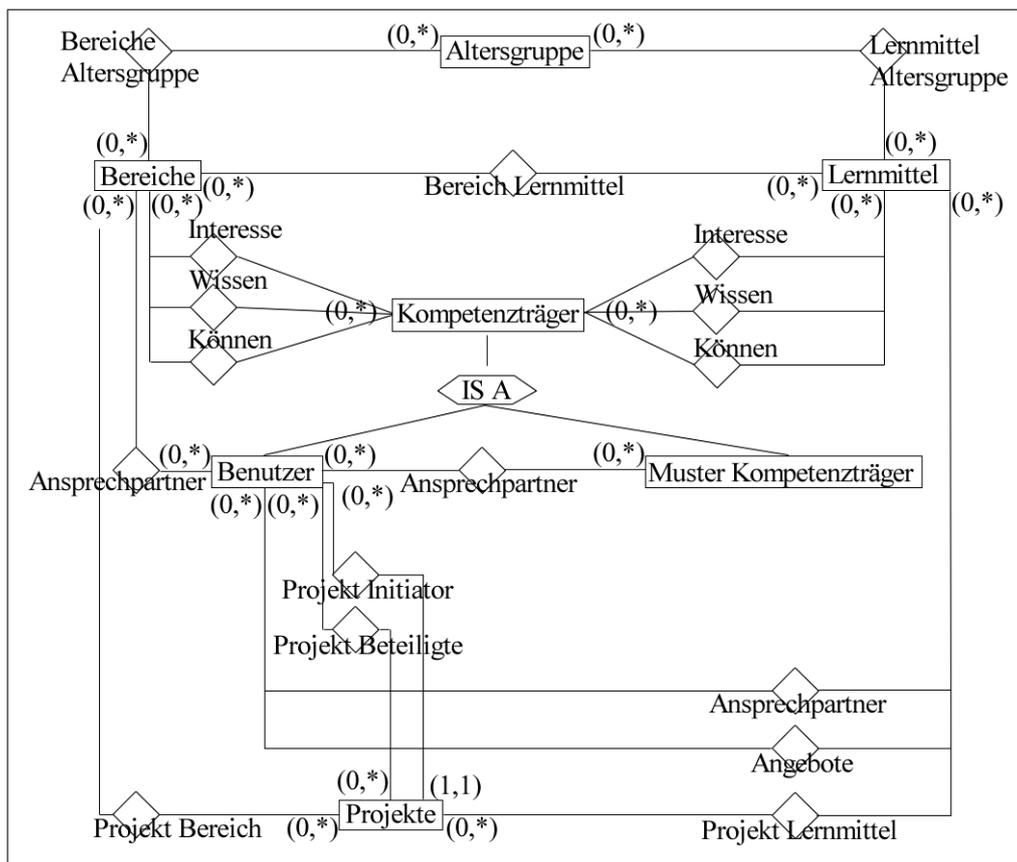


Abbildung 3.5: ER-Modell: Lernmittel - Altersgruppen - Bereiche

ausgebaut werden.

3.2 Rechte- und Benutzerverwaltung

Im Folgenden wird das oben entwickelte ER-Modell um eine einfache Benutzerverwaltung und Rechteverwaltung erweitert.

Zuerst wird das ER-Modell um eine Rechteverwaltung erweitert, durch die es möglich wird den Benutzern unterschiedliche Rechte für die Benutzung von EDEJU zuzuteilen. Z.B. ist es notwendig, dass bestimmte Benutzer das Recht haben, die Bereichsstruktur zu erweitern oder Lernmittel einzutragen. Im Anhang D werden die verschiedenen Rechte aufgeführt und genauer beschrieben.

Um die Rechteverwaltung einzubauen wird eine Entität Rechte hinzugefügt und eine Beziehung zu Benutzer. Die Entität Rechte enthält die Attribute *Name*, die *Bezeichnung* des Rechts und eine *Beschreibung*. Die Beziehung enthält nur die jeweiligen Schlüssel. Durch diese Anordnung ist es möglich, die Rechte beliebig zu erweitern. In der Abbildung 3.6 sind die Erweiterungen um die Rechteverwaltung und Benutzerverwaltung im ER-Modell erkenntlich. Jedes Recht muss mindestens einmal mit einem Benutzer in Beziehung stehen. Jeder Benutzer muss das Recht, welches er vergeben will auch besitzen. Es ist notwendig,

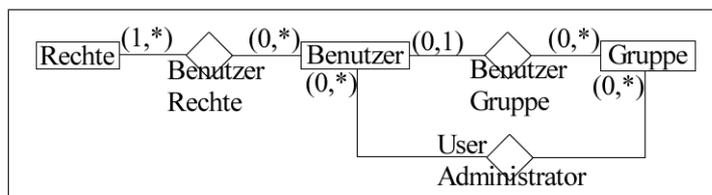


Abbildung 3.6: ER-Modell: Benutzerverwaltung

dass jedes Recht mindestens einem Benutzer zugeordnet ist, da ansonsten diese Rechte nur noch "manuell" einem Benutzer übertragen werden können. Manuell heißt, dass die Datenbank direkt geändert werden muss.

Um den Überblick über die Benutzer von EDEJU zu behalten, werden Gruppen gebildet. Ein Benutzer kann dabei maximal in einer Gruppe enthalten sein. Die Gruppen können dabei frei gestaltet werden, z.B. Verteilung nach dem Alphabet oder nach dem Wohnort bzw. der Wohnregion. Benutzer, die keiner Gruppe angehören, sind auch erlaubt, da z.B. nach einer Anmeldung eines Benutzers, dieser noch keiner Gruppe zugeteilt wurde.

Diese verschiedenen Gruppen werden von sogenannten *User-Administratoren* verwaltet. Diese sind Benutzer, die das Recht haben, die Daten anderer Benutzer zu bearbeiten. Dabei können diese das Recht haben, eine, aber auch mehrere Gruppen zu verwalten. Die User-Administratoren bilden eine eigene Gruppe, die von den *Administratoren* verwaltet werden. Die Administratoren sind Benutzer, die ganz oben in der Hierarchie der Benutzerverwaltung stehen.

Es ist klar, dass die Benutzerverwaltung noch weiter ausgebaut werden muss. Im Rahmen dieser Diplomarbeit wird allerdings nur eine grundlegende Benutzerverwaltung aufgebaut, da der Hauptaugenmerk auf der Entwicklung des EDEJU - Atlas und dessen Nutzung liegt.

3.3 Geschäftsprozesse

Innerhalb dieses Abschnitts werden die Geschäftsprozesse genauer definiert. Diese sind unterteilt in die Verwaltungsprozesse, d.h. die Prozesse, die nur von bestimmten Benutzern ausgeführt werden. Darunter fallen die Benutzerverwaltung, die Bearbeitung der Bereiche, der Lernmittel, der Altersgruppen und der Muster-Kompetenzträger.

Die Nutzungsprozesse, die im öffentlichen Bereich auftreten erstrecken sich auf die "normale" Nutzung der nicht angemeldeten Benutzer von EDEJU und auf die Benutzung durch die angemeldeten Benutzer, denen eine Vielzahl von zusätzlichen Funktionen zur Verfügung stehen.

Im Anhang B sind die verschiedenen Geschäftsprozessdiagramme dargestellt.

3.3.1 Verwaltung

Verwaltungsaufgaben sind unterteilt in die Verwaltung der Daten und der Benutzerverwaltung.

Die Datenverwaltung beinhaltet Lernmittel und Bereiche. Diese bilden den Kern von EDE-JU. Es ist notwendig Schnittstellen zur Verfügung zu stellen, um diese Daten einfügen, bearbeiten und löschen zu können. Während des Einfügens müssen nicht nur die Daten eingegeben werden, es müssen auch die Beziehungen zu anderen Bereichen bzw. Lernmitteln hergestellt werden. Die Möglichkeit, Beziehungen nachträglich zu ändern oder zu löschen, muss gegeben sein. Bei der Eingabe von Bereichen wird z.B. unter Kultur Popmusik eingetragen. Erst später wird der Bereich Musik unter Kultur eingefügt. Jetzt muss die Möglichkeit gegeben sein, Popmusik mit allen seinen Beziehungen unter den neuen Bereich Musik verschieben zu können.

Die Datenverwaltung beinhaltet zusätzlich noch die Eingabe von Altersgruppen und die Erstellung von Muster-Kompetenzträgern. Die Beziehung zwischen Altersgruppen und Lernmitteln bzw. Bereichen erfolgt bei der Eintragung bzw. Änderung des Lernmittels bzw. des Bereichs. Der Benutzer sollte dort die Möglichkeit haben Altersgruppen einem Lernmittel bzw. einem Bereich hinzufügen zu können.

Muster-Kompetenzträgern muss neben den allgemeinen Daten noch ein Profil hinzugefügt werden. Es muss daher möglich sein, Bereiche bzw. Lernmittel als *Interesse*, *Wissen* und *Können* dem Profil des Muster-Kompetenzträgers hinzuzufügen.

Innerhalb der Benutzerverwaltung müssen Benutzer von Dritten, d.h. User-Administratoren oder Administratoren bearbeitet werden und gegebenenfalls mit weiteren Rechten versehen werden. So muss z.B. einem Benutzer, um Lernmittel bearbeiten zu können, vorher das Recht "Lernmittel eintragen und verändern" erteilt werden.

Um die Gruppen zu verwalten, werden sog. User-Administratoren benötigt. Diese haben die Aufgaben die Daten der Benutzer zu überprüfen und als Ansprechpartner zur Verfügung zu stehen. Es ist sinnvoll, einem User-Administrator nicht alle Gruppen zur Verwaltung zu übergeben, sondern nur einige wenige. Auf diese Weise verlieren diese nicht den Überblick, über die zu verwaltenden Benutzer. Diese Übergabe muss von den Administratoren vollzogen werden, die die Gruppe der User-Administratoren verwalten. Die verwendete Benutzerhierarchie ist in Abbildung 3.7 dargestellt. Aus dieser Abbildung geht hervor, dass ein Administrator gleichzeitig auch ein User-Administrator und ein normaler Benutzer ist. Dadurch wird es z.B. möglich, dass ein Administrator, der eine Gruppe verwaltet, bei einer Herabstufung auf einen User-Administrator diese weiterhin verwalten kann.

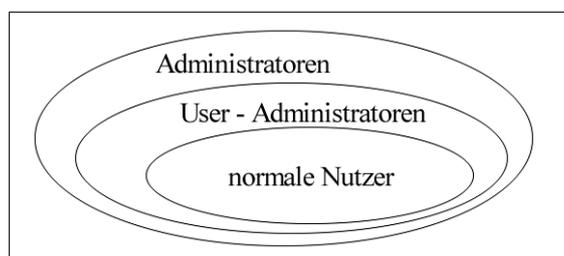


Abbildung 3.7: Benutzerhierarchie

Weiterhin muss die Benutzerverwaltung die Möglichkeit bieten, mit den Benutzern Kontakt aufzunehmen, um Fragen bzw. Probleme zu diskutieren.

3.3.2 Rechte

Da nicht jedem Benutzer alle Funktionen von EDEJU offenstehen, ist es notwendig, Benutzern Rechte zu vergeben bzw. wieder zu nehmen, falls diese missbraucht werden. Im Anhang D sind die bisher benötigten Rechte aufgeführt. Bei der Weitergabe von Rechten ist zu beachten, dass nur die Rechte weitergegeben werden dürfen, die der vergebende Benutzer auch selbst besitzt.

EDEJU sollte offen sein, die Rechteverwaltung auszubauen, da mit jeder Erweiterung z.B. der Benutzerverwaltung neue Rechte hinzukommen können. Dies ist gewährleistet durch den Aufbau der Rechteverwaltung (Abbildung 3.6), da es möglich ist, eine beliebige Anzahl von Rechten zu verwalten.

3.3.3 Mitglieder-Bereich

Im Mitglieder-Bereich kann der Benutzer seine eigenen Daten bearbeiten. Dazu gehören seine persönlichen Daten wie seine Adresse oder sein Beruf aber auch sein Profil. Es muss möglich sein, das eigene *Interesse*, *Wissen* und *Können* zu bearbeiten und gegebenenfalls auch zu löschen. Zusätzlich soll der Benutzer dort eigene Projekte anlegen und bearbeiten können. Der Benutzer soll die Möglichkeit haben, den eigenen Account und die gespeicherten Daten selbst zu löschen.

3.3.4 Öffentlicher Bereich

Im öffentlichen Bereich werden alle Daten, die in EDEJU vorhanden sind, ausgegeben. Aus diesem Grund ist es wichtig diesen übersichtlich und benutzerfreundlich zu gestalten.

Im Mittelpunkt von EDEJU stehen die Bereiche und Lernmittel. Ausgehend von den wenigen Startbereichen ist es möglich, Schritt für Schritt alle Daten zu erschließen. Der Benutzer soll die Möglichkeit haben, die Unterbereiche, Querverweise, Lernmittel, Projekte, usw., die mit einem bestimmten Bereich verknüpft sind, zu sehen und sich auf diese Weise in den Daten von EDEJU "fortzubewegen". Während der Benutzer sich nur durch die Daten "bewegt", muss es dem angemeldeten Benutzer zusätzlich möglich sein, Bereiche oder Lernmittel dem eigenen Profil hinzuzufügen.

Eine Suche, die direkt nach Lernmitteln, Bereichen, Projekten oder Personen sucht, hilft dem Benutzer zielgenau, nach Themen zu suchen, die ihn interessieren. Dabei hilft eine "Bereichsmap", die z.B. alle Unterbereiche bis zu einer gewissen Tiefe ausgibt. Allerdings ist dies durch die zu erwartende Größe einer solchen Bereichsmap sehr problematisch und kann das Layout von EDEJU sprengen.

Um in EDEJU zu stöbern, sind Vorschläge des Systems hilfreich. Bei EDEJU sind drei Arten von Vorschlägen angedacht. Willkürliche Vorschläge, z.B. 5 aus der Datenbank willkürlich herausgesuchte Bereiche, sind sowohl für angemeldete Benutzer mit einem Profil, wie auch für Benutzer, die noch nicht angemeldet sind und nur schnuppern wollen, eine

sinnvolle Möglichkeit Bereiche zu finden. Auf diese Weise ist es möglich, Bereiche zu finden, die neue Ideen für eine Weiterentwicklung bringen und normalerweise nicht entdeckt worden wären.

Die beiden anderen Arten von Vorschlägen sind nur für Benutzer mit einem bereits erstellten Profil nutzbar. Im ersten Fall soll EDEJU anhand des Profils "ähnliche Vorschläge" machen. Dies bedeutet, EDEJU versucht Bereiche anhand des Profils zu bewerten und gibt dann die Vorschläge mit der höchsten Wertung aus. Wie diese Wertung realisiert wurde, wird in Kapitel 5.2.2 genauer erläutert.

Die Ausgabe von Muster-Kompetenzträgern und der Vergleich des eigenen Profils mit dem einer anderen Person oder mit einem Muster-Kompetenzträger ist eine weitere Funktion, die im öffentlichen Bereich realisiert wird.

Natürlich gibt es neben den hier aufgeführten Arten der Ausgabe der Daten noch unzählige andere Ausgabeformen. Innerhalb der Suche kann es z.B. viele verschiedene Kombinationsmöglichkeiten geben, um die gesuchten Daten schneller zu finden. Im Kapitel 6 werden einige mögliche Erweiterungen, die nicht im Rahmen dieser Diplomarbeit implementiert wurden, genauer beschrieben.

Kapitel 4

Entwurfsphase

Innerhalb dieses Kapitels werden verschiedene Grundsatzentscheidungen z.B. über die benutzte Programmiersprache und die verwendete Datenbank getroffen. Anschließend wird die Benutzerverwaltung und die Rechteverwaltung konkretisiert. Dann wird der Kernbereich von EDEJU (die Struktur der Bereiche) genauer betrachtet und mögliche Techniken diese zu speichern erörtert. Nachdem diese Struktur feststeht, wird das Aussehen der übrigen Daten und Funktionen, die im Mitglieder- und Verwaltungsbereich benötigt werden, entworfen. Während dieses Entwurfs wird das Datenbankschema aufbauend auf dem ER-Modell, welches in Kapitel 3 definiert wurde, festgelegt. Schließlich werden die Geschäftsprozesse, die im öffentlichen Bereich benötigt werden, entwickelt und im letzten Abschnitt dieses Kapitels wird noch auf das Design der Ausgabe eingegangen.

4.1 Grundsatzentscheidungen

4.1.1 Architektur

Wie schon der Name dieser Arbeit "Entwicklung des Webportals www.edeju-atlas.de" sagt, basiert EDEJU auf einer Webarchitektur. Im Rahmen der ersten Gespräche mit Herrn Helmeth zur Entwicklung und zum möglichen Aussehen von EDEJU wurden auch andere Architekturen, wie die Client/Server Architektur oder die Verteilung von EDEJU als Anwendungsprogramm durchdacht, allerdings wurden diese wieder verworfen.

Das lebendige, sich dauernd verändernde Wesen von EDEJU schließt beispielsweise die Programmierung als Anwendungsprogramm aus, da jedesmal, wenn EDEJU gestartet würde Updates in 2 Richtungen notwendig wären. Zuerst müssten über ein Update die aktuellen Datenbestände eingespielt werden, eventuelle Änderungen müssten dann an eine zentrale Stelle, z.B. einen Server, übermittelt werden. Die daraus resultierenden Probleme der Datenredundanz und der Kollision von Änderungen verhindern die Entwicklung von EDEJU als Anwendungsprogramm.

Eine Client/Server Architektur¹ kann diese Kollisionen durch die dauerhafte Verbindung während einer Session umgehen. Allerdings sind Client/Server Architekturen sehr aufwendig in der Programmierung, da sowohl der serverseitige Teil, wie auch der clientseitige

¹vgl. H.Balzert S. 703ff

Teil programmiert werden muss. Ausserdem müssen für den clientseitigen Teil mehrere Betriebssysteme berücksichtigt werden. Ein weiterer Nachteil ist die nötige Verbreitung der Clientsoftware.

Die Webarchitektur² bietet die Möglichkeit EDEJU, schnell und unkompliziert weltweit anzubieten. Voraussetzung ist natürlich ein Internetanschluss. Vorteile sind, dass keine Verbreitung von clientseitiger Software notwendig ist, da ein Web Browser genügt. Ausserdem wird so die Möglichkeit geboten, neue Daten direkt und ohne spezielle Software-Updates dem Benutzer zugänglich zu machen.

Die Beschränkung auf die grafischen Möglichkeiten eines Web Browsers sprechen nicht gegen diese Architektur. Die grafischen Beschränkung auf die Möglichkeiten eines Web Browsers sind nicht mehr so gravierend, da diese z.B. durch Flash³ oder VRML⁴ erweitert werden können. An der Entwicklungsgeschichte der Browser ist erkenntlich, dass diese sich ausgehend von reinen textbasierten Browser(wie Lynx⁵) zu solchen entwickelt haben, die mittlerweile fast jede Gestaltungsmöglichkeiten zulassen, wie z.B. Mozilla⁶ oder Internet Explorer⁷.

Da HTTP⁸ ein zustandsloses Protokoll ist, stellt eine Sessionverfolgung ein Problem dar und lässt sich auf 2 Arten lösen. Zum einen kann ein Cookie⁹ beim Web Browser hinterlegt werden, in dem die Session-Informationen eingetragen sind. Ein Nachteil dabei ist, dass der Web Browser das Anlegen von Cookies explizit erlauben muss. Viele Internetnutzer haben aber diese Funktion ihres Browsers entweder eingeschränkt oder aber ganz abgestellt, aus Angst, es könnte z.B. ein "Surf Profil" erstellt werden. Aus diesem Grund wird bei EDEJU die zweite Möglichkeit verwendet: das sog. URL-rewriting. Dabei werden die Informationen, die für eine Session Verfolgung benötigt werden, an die URL angehängt und entweder per HTTP-GET Anfrage oder aber per HTTP-POST Anfrage verschickt.

Um eine Session zu verfolgen, wird bei EDEJU jedem Benutzer dynamisch eine eindeutige Session-ID zugeteilt, die dann jeweils per URL-rewriting weitergegeben wird. Der Nachteil bei dieser Methode, ist die Beschränkung auf 256 Zeichen bei der HTTP-GET Anfrage. Diese Beschränkung ist notwendig, da ansonsten ältere Browser die GET Anfragen nicht mehr richtig versenden. Für größere Anfragen, die mehr Daten versenden, kann auch eine HTTP-POST Anfrage verwendet werden, die nicht in der Größe beschränkt ist. Allerdings ist die HTTP-GET Anfrage durch das URL-rewriting, durch das direkte Anhängen von Übergabewerten, komfortabler.

²vgl. H.Balzert S. 703ff

³siehe <http://www.macromedia.com/>

⁴"Virtual Reality Modeling Language" siehe <http://www.web3d.org/vrml/vrml.htm>

⁵siehe <http://lynx.browser.org/>

⁶siehe <http://www.mozilla.org/>

⁷siehe <http://www.microsoft.com/>

⁸nähere Informationen unter A.S.Tanenbaum S.726

⁹Ein Cookie bezeichnet Informationen, die ein Webserver zu einem Browser sendet, um dem zustandslosen HTTP-Protokoll die Möglichkeit zu geben, Information zwischen Aufrufen zu speichern.

4.1.2 Datenbank

Um die Daten von EDEJU effizient verwalten zu können, ist eine Datenbank nötig. Die Datenmenge, die zu erwarten ist macht diesen Einsatz notwendig. Aufgrund des ER-Modells, welches in Kapitel 3 definiert wurde, bietet es sich an, eine Relationale Datenbank zu benutzen. Die Wahl, welche relationale Datenbank bei EDEJU benutzt wird, wurde unter Kostengesichtspunkten getroffen. Unter dieser Betrachtung fielen z.B. die Oracle Datenbank, MS SQL oder DB2 heraus. Im Internet gibt es eine Vielzahl von frei verfügbaren Datenbanken, die zwar teilweise für den kommerziellen Gebrauch lizenzpflichtig, aber ansonsten lizenzfrei sind. Darunter fallen MaxDB, Firebird, PostgreSQL und MySQL. Für EDEJU wurde MySQL ausgewählt, da dieses Datenbanksystem erstens sehr kostengünstig ist und zweitens im Internet sehr verbreitet ist. Ein weiterer Vorteil ist die Schnelligkeit und die vorhandene Verbindung mit der Programmiersprache PHP. EDEJU ist neben der Ausgabe, die immer durch HTML geschieht, mit PHP implementiert, da viele Internetprovider das Bundle MySQL-Datenbank mit PHP Unterstützung kostengünstig anbieten. Natürlich gibt es bei MySQL nicht die Funktionalität, die andere Datenbanksysteme haben, aber diese Nachteile können teilweise überwunden werden oder werden durch neue Versionen von MySQL behoben. EDEJU wurde für die Version MySQL 3.23 programmiert. Hauptnachteil ist dabei das Fehlen von referentiellen Integritäten. Es ist nicht möglich innerhalb der Datenbank referentielle Integritäten zu definieren. Dadurch müssen z.B. beim Löschen von Datensätzen immer mehrere Updates ausgeführt werden. Ein weiteres Problem ist das Fehlen der UNION Anweisung, dies kann aber durch das erstellen einer temporären Tabelle gelöst werden.

Diese Lösungen gehen natürlich zu Lasten der Performanz und sind fehleranfällig. Aber in den neuen Versionen von MySQL 4.0 und MySQL 5.0 sind diese fehlenden Funktionalitäten aufgenommen worden. Diese Versionen werden im Rahmen dieser Diplomarbeit nicht benutzt, da diese während der Entwicklung dieser Arbeit neu herauskamen. Es ist allerdings möglich, die Datenbank auf neue Versionen umzustellen und dadurch die Performanz zu steigern. Die benötigten Schritte werden in Kapitel 6 näher erläutert.

Mithilfe dieser Grundsatzentscheidungen, die Wahl der Architektur, der Datenbank und der Programmiersprache ist es nun möglich die benötigten Teile von EDEJU zu entwerfen.

4.2 Infrastruktur

In Abschnitt 4.1.1 wurde schon erläutert, wie trotz Zustandslosigkeit des HTTP Protokolls über URL-rewriting eine Session simuliert werden kann. Zuerst muss der Benutzer identifiziert werden. Dazu muss er sich mit einem Benutzernamen und seiner Email Adresse bei EDEJU anmelden, dann bekommt er per Email ein Passwort zugeschickt. Diese Prozedur wird benötigt, um "Spassanmeldungen", die keine korrekte Email Adresse angeben, zu vermeiden. Passwort, Benutzername und Email werden in die Relation *rechte_user* eingetragen. Bei der Anmeldung werden zuerst die Zugangsdaten überprüft, dann wird eine Session-ID erzeugt oder bei falschen Daten der Zugriff abgelehnt. Die Session-ID wird in die Relation *rechte_user* eingetragen. Nachdem eine Session erzeugt wurde, muss nun an jeden Link, der innerhalb von EDEJU verzweigt die Session-ID gehängt werden:

"dateixy.php?sid=SessionID".

Die Sessionerzeugung ¹⁰ erfolgt folgendermaßen:

$$SessionID = substr(userid + md5(uniqid(rand())), 0, 30).$$

Die Session-ID sollte eindeutig sein, da die Userid innerhalb der Datenbank eindeutig ist. Falls zufällig doch eine Session-ID erzeugt wird, die schon vergeben wurde, wird dies abgefangen und eine neue Session-ID erzeugt. Damit die Session-ID Erzeugung nicht nachvollziehbar ist, wird mithilfe der Funktion *uniqid(rand())* eine Zufallszeichenfolge auf Basis der Mikrosekunden erzeugt, die dann mit *md5()* verschlüsselt wird. Die ersten 30 Zeichen stellen nun die Session-ID dar.

Zusätzlich wird ein Timeout hinzugefügt, das den Benutzer, wenn innerhalb eines bestimmten Zeitraumes z.B. 20 Minuten keine Eingaben erfolgen automatisch ausloggt. Jedesmal, wenn der Benutzer eine Seite neu lädt oder eine andere Seite aufruft wird die Session-ID überprüft und gleichzeitig der Timeout in der Relation *rechte_user* neu gesetzt.

Mithilfe dieser Session Verfolgung ist es möglich, den Benutzer immer eindeutig zu identifizieren.

Funktionen, die immer wieder auftauchen werden ausgelagert und dann jeweils, falls nötig, eingebunden. Zu diesen Funktionen gehören alle Session Funktionen *Session erzeugen*, *Session überprüfen* oder *Timeout überprüfen* und eine Funktion, die überprüft, ob der Benutzer ein bestimmtes Recht hat oder nicht. Zusätzlich werden auch Funktionen, die bestimmte Daten, wie Benutzername, Name oder Vorname liefern, ausgelagert.

4.3 Datenbankschema

Das ER-Modell, welches in der Definitionsphase entwickelt wurde, wird in ein logisches Datenbankmodell transformiert. Jeder Relation wird ein zusätzliches Feld *ID INT(11)* hinzugefügt. MySQL bietet eine Funktion (*Auto_Increment*) an, die automatisch beim Einfügen eines neuen Tupels in eine Relation einem Feld eine eindeutige Zahl zuordnet. Diese Funktion¹¹ wird für die Schlüsselvergabe verwendet. Beim Einfügen eines neuen Datensatzes trägt MySQL automatisch einen Wert ein, der um eins größer ist als der jemals in der Relation gespeicherte höchste Wert. Das zusätzliche Feld *ID INT(11)*, welches den Primary Key darstellt, ist mit dieser Autoincrement Funktion versehen. Durch dieses Feld ist es nun möglich, Datensätze eindeutig miteinander in Beziehung zu setzen. Die ID eines Datensatzes ist gleichzeitig der Schlüssel, der in anderen Relationen als Fremdschlüssel eingetragen ist.

Um das ER-Modell zu transformieren werden in einem ersten Schritt für jeden Entitätstyp ein Relationschema mit den jeweiligen Attributen erstellt. Die Umwandlung der ISA-Beziehungen der Kompetenzträger werden in Abschnitt 4.4 beschrieben. Schließlich werden

¹⁰Vergleiche J.Krause S.250

¹¹Vergleiche P.Dubois S.72

alle Beziehungen zwischen den Entitäten in Relationsschemata transformiert. Die *IDs* der beteiligten Entitäten werden als Fremdschlüssel benutzt. Eine Ausnahme bildet dabei die Beziehung *Unterbereiche*, diese wird in 4.5 genauer beschrieben. Die übrigen Transformationen werden im Abschnitt 4.6 genauer erläutert.

Im Anhang A ist das dabei entstandene Datenbankschema dargestellt.

4.4 Rechte- und Benutzerverwaltung

Als erstes ist es notwendig, eine Benutzerverwaltung mit einer effizienten Rechteverwaltung aufzubauen. Im Abschnitt 3.2 wurden die 3 Arten eines möglichen Benutzers (normaler Benutzer, User-Administrator und Administrator) schon genauer erläutert. In der Abbildung 3.6 ist die Benutzerverwaltung aufgeteilt in die 3 Entitäten Benutzer, Rechte und Gruppen und die Beziehungen Benutzerrechte, Benutzergruppe und User-Administrator. Da offensichtlich ist, dass jeder Benutzer jedes Recht genau einmal haben kann, ist es sinnvoll, die Entität Rechte mit der Beziehung Benutzerrechte und Benutzer zusammenzulegen. Bei EDEJU heißt diese Relation *rechte_user*. Innerhalb dieser Relation ist jedes Recht als *ENUM (0,1)* Feld vorhanden, d.h. in diesem Fall kann das Feld entweder den Wert 0 oder den Wert 1 haben. Falls das Feld gleich dem Wert 1 ist, besitzt der Benutzer das Recht, ansonsten nicht. Da die Session-ID in dieser Relation auch vorhanden ist, kann ohne zusätzliche Abfragen überprüft werden, ob ein eingeloggter Benutzer das Recht hat, eine Aktion auszuführen oder nicht. Zusätzlich zu den Feldern der Relation, die die Rechte beinhalten, gibt es noch eine zusätzliche Relation *rechte_rights*, die eine genaue Beschreibung und den Namen der Rechte enthält. Diese Relation wird nur für die Verwaltung benötigt, um die Rechte, die vergeben werden können, zu erläutern. Um ein neues Recht einzufügen, muss in *rechte_rights* ein neuer Eintrag vorgenommen werden und in *rechte_user* ein neues *ENUM (0,1)* Feld hinzugefügt werden mit dem Namen des jeweiligen Rechtes.

Rechte, die unabhängig von der Benutzerverwaltung sind, werden in den nächsten Abschnitten dieses Kapitels aufgeführt und erläutert.

Die Einteilung in die 3 Benutzerarten wird durch die Vergabe von bestimmten Rechten vorgenommen. Ein User-Administrator ist ein Benutzer, der das Recht *rig_useradmin* hat. Ein Administrator hat sowohl dieses Recht, als auch das Recht *rig_admin*. Zusätzliche Rechte die für die Benutzerverwaltung benötigt werden, sind z.B. *rig_admin_change*. Es gibt einem Administrator das Recht andere Administratoren zu bearbeiten. *Rig_userdelete* gibt dem User-Administrator oder dem Administrator das Recht andere Benutzer zu löschen. Schließlich gibt *rig_giverights* das Recht, Rechte an andere Benutzer weiterzugeben, natürlich nur an Benutzer, die der jeweilige Benutzer auch verwaltet. Es muss durch die Software gewährleistet werden, dass die jeweiligen Beziehungen zwischen den Rechten erfüllt sind, z.B. muss ein Benutzer der das Recht *rig_admin* hat, gleichzeitig immer auch das Recht *rig_useradmin* haben.

Bei der Zugehörigkeit eines Benutzers zu einer Gruppe wird die Beziehung zu einer Gruppe in die Relation Benutzer überführt. Dies ist möglich, da ein Benutzer maximal nur in einer Gruppe enthalten sein kann. Daher wird in die Relation *rechte_user* das Feld *gruppe INT(11)* hinzugenommen, das den Schlüssel der entsprechenden Gruppe enthält. Falls ein

Benutzer keiner Gruppe angehört, ist dieser Wert gleich 0. Da es möglich sein soll, Benutzer Gruppen zuzuteilen und diese Zuteilung vom jeweiligen User-Administrator akzeptiert oder abgelehnt werden können sollen, ist es notwendig, 2 zusätzliche Felder hinzuzufügen. Zum einen kommt das Feld *oldgruppe INT(11)* hinzu, welches die alte Gruppe des Benutzers enthält. Zum anderen gibt es das Feld *transfer ENUM(0,1,2)*. Der Wert 0 gibt an, dass der Benutzer gerade nicht verschoben wird. Der Wert 1 bedeutet, dass der Benutzer von *oldgruppe* nach *gruppe* verschoben werden soll, der Wert 2 gibt an, dass der User-Administrator der neuen Gruppe diesen Benutzer abgelehnt hat und dieser wieder in die alte Gruppe übernommen werden sollte. Durch diese Transfermechanismen ist es möglich, für die User-Administratoren einen Überblick über die Benutzer zu behalten und sofort neue von alten Benutzern unterscheiden zu können.

4.5 Bereiche

Ausgehend von den vier Bereichen (Natur, Technik, Organisation, Kultur) ist es das Ziel, möglichst alle Themen und Wissensgebiete, zu katalogisieren und in einer Struktur zu vereinen.

Es hat sich im Laufe des Entwicklungsprozesses gezeigt, dass eine Baumstruktur¹² dafür am Besten geeignet ist, da bei jedem Bereich klar sein muss, zu welchem der 4 Ausgangsbereiche der Bereich gehört.

Aus dem ER-Modell (Abbildung 3.3) geht hervor, dass die Bereiche zwei rekursive Beziehungen haben. Die Beziehung der Unterbereiche stellt eine Baumartige Struktur dar, welche ausgehend von den vier Bereichen sich in viele andere Bereiche auffächert.

Um eine Baumstruktur zu bilden, muss eine Wurzel als besonderer Knoten definiert werden. An diese Wurzel können nun die Ausgangsbereiche gehängt werden. Zu beachten ist dabei, dass diese Wurzel nicht gelöscht werden darf, da ansonsten die 4 Ausgangsbereiche nicht mehr auf der gleichen Tiefe innerhalb der Baumstruktur stehen würden. Dieser Knoten kann natürlich, wie ein normaler Bereich, mit einer Beschreibung und einem Bild versehen werden, z.B. kann innerhalb der Beschreibung eine kurze Einführung in die Struktur der Bereiche gegeben werden.

Anhand dieser Bereichsstruktur wird nun eine Navigation durch das System möglich, da dadurch eine Hierarchie innerhalb der Bereiche erstellt wird. Um diese Baumstruktur innerhalb einer Datenbank zu speichern und effizient zu nutzen, wird das Nested Sets Model benutzt. Im Abschnitt 4.5.2 wird hierauf näher eingegangen.

Natürlich ist es nicht möglich, jeden Bereich genau einem anderen Bereich zuzuordnen. Hier kommt nun die andere rekursive Beziehung der Bereiche, die Querverweise ins Spiel. So kann z.B. das Musikinstrument Digeridoo der Kultur zugeordnet werden, doch um ein solches Instrument herzustellen ist es notwendig auch eine Verbindung zu Technik zu ziehen. Daher werden in einer zusätzlichen Relation Querverbindungen innerhalb der Baumstruktur abgespeichert.

Im Folgenden wird zuerst die Art der Baumstruktur, die für die Beziehung Unterbereiche

¹²nähere Informationen unter T.Ottmann/P.Widmayer S.235ff

benötigt wird, besprochen, danach wird auf die benutzte Tabellenstruktur der Bereiche genauer eingegangen.

4.5.1 Modelle zur Abspeicherung von Baumstrukturen in Datenbanken

Es gibt mehrere Möglichkeiten Baumstrukturen in einer Datenbank zu speichern. Drei von diesen werden hier vorgestellt, das Parent-Modell, das Pfad-Modell und das Nested Sets Modell. Diese Modelle haben gewisse Vor- und Nachteile.

Parent-Modell

Beim Parent-Modell auch Adjazenzlisten Modell¹³ genannt wird der Vater eines Knotens in der Knotenrelation mit einem zusätzlichen Feld gespeichert. Es ist natürlich auch möglich, Kanten des Baumes in eine zusätzliche Relation auszulagern. Dies ist z.B. zur Speicherung von allgemeinen Graphen der Fall. Da aber bei Bäumen jeder Knoten nur einen Vorgänger hat, kann die zusätzliche Information in der Knotenrelation mit gespeichert werden.

Ein Problem bei dieser Speicherung sind Teilbaumanfragen, da diese nur rekursiv beantwortet werden können. So können für einen Knoten zwar problemlos alle Kinder bestimmt werden, dann müssen aber für jedes dieser Kinder wieder alle Kinder ermittelt werden, usw.. Dies führt, da der Baum sehr tief werden kann, zu einem großen rekursiven Aufwand. Auch Anfragen, die ausgehend von einem Knoten z.B. alle Knoten der selben Tiefe oder alle Vorgänger ermitteln sollen, sind problematisch. Im ersten Fall muss ausgehend von der Wurzel der Teilbaum bis zu der gewünschten Tiefe ermittelt werden, im zweiten Fall liegt der Aufwand in der Größe der Tiefe des Knotens. Die Einfüge-, Entferne- und Updateoperationen sind im Gegensatz dazu einfach zu bewerkstelligen. Zum Einfügen wird nur der Vater des neuen Knotens benötigt, zum Entfernen müssen alle Datensätze geändert werden, die den zu entfernenden Knoten als Vorgänger eingetragen haben. Update Operationen wie das Verschieben von Teilbäumen sind sehr einfach, da nur der Vorgänger des zu verschiebenden Wurzelknotens geändert werden muss.

Vorteile des Parent-Modell sind eindeutig die einfachen Einfüge-, Entferne- und Updateoperationen. Da bei EDEJU Abfragen und die Struktur der Bereiche eine sehr bedeutende Rolle spielen und diese Update - Operationen nur am Rande ausgeführt werden, wird das Parent-Modell aber nicht verwendet.

Pfad-Modell

Das Pfad-Modell verfolgt einen anderen Ansatz. Es speichert für jeden Knoten alle Vorgänger bis zur Wurzel in einem zusätzlichen Feld in der Relation. Dies geschieht z.B. durch Zahlen, die die Knoten eindeutig identifizieren, die durch ein Zeichen wie ";" oder "-" voneinander getrennt werden. Ein Vorteil ist, dass dadurch Verwandtschaftsgrade ohne Rekursion beantwortet werden können. Um die Vorgänger eines Knotens zu ermitteln, muss nur dieses Feld ausgegeben werden. Teilbäume lassen sich mit dieser Methode leicht ermitteln. So haben alle Knoten, die im gleichen Teilbaum liegen, ausgehend von der Wurzel den gleichen

¹³siehe J.Celko S.432

Listen-Anfang.

Diese Möglichkeit eignet sich aber nur dann, wenn die Tiefe des Baumes begrenzt ist, da ansonsten zusätzlich zu den anderen Daten eine riesige Liste bei jedem Knoten gespeichert werden muss und der Overhead sehr groß wird. In MySQL kann z.B. das Feld vom Typ "varchar" genutzt werden, dieses ist aber in der Länge begrenzt. Um den verfügbaren Speicherplatz zu erhöhen, kann ein Feld des Typs "text" benutzt werden. Ein Index über das Feld muss aber wieder in der Länge begrenzt sein. Ein Index auf diesem Feld ist sinnvoll, um den Zugriff auf die Struktur des Baumes zu beschleunigen. Dies ist notwendig, da die Struktur der Bereiche eine große Rolle innerhalb von EDEJU spielen und Anfragen möglichst schnell beantwortet werden sollten.

Weitere Probleme treten bei den Einfüge-, Entferne- und Update-Operationen auf. Während das Einfügen keine Probleme bereitet, da die Liste des Vaters mit der ID des Vaters verwendet werden kann, müssen beim Entfernen viele Datensätze geändert werden. Bei allen Knoten, die im Teilbaum dieses Knoten enthalten sind, muss der Pfad angepasst werden. Dies ist auch bei Verschiebeoperationen der Fall.

Dieser entscheidende Nachteil, die Länge der Liste und der damit verbundene Speicheraufwand, verhindert die Verwendung dieses Modells bei EDEJU. Da bei EDEJU die Größe und Tiefe des Baumes nicht vorhergesehen werden kann und damit auch nicht in der Tiefe begrenzt werden kann, kann auch dieses Modell nicht verwendet werden.

4.5.2 Vorstellung des Nested Sets Modell

Struktur

Die Baumstruktur beim Nested Sets Modell¹⁴ wird mithilfe von zwei weiteren Feldern in der Tabelle abgespeichert. Für jeden Knoten im Baum werden 2 zusätzliche Felder *lft* und *rght* vom Typ INT gespeichert. Bei EDEJU wird zusätzlich noch die Tiefe bzw. der Level des Knotens abgespeichert, da dies bei einigen Abfragen einen großen Vorteil bietet. Es ist aber auch möglich, den Level anhand der beiden Felder *lft* und *rght* zu berechnen.

In Abbildung 4.1 ist eine solche Nested Sets Baumstruktur dargestellt. In Abbildung 4.2 ist diese Struktur als Tabelle dargestellt. Die Nummerierung eines solchen Baumes erfolgt wie folgend: beginnend bei der Wurzel A, deren *lft* Wert immer 1 ist, geht es weiter mit dem ersten Kind B. *B.lft* wird 2, der rechte Wert wird vorerst nicht belegt, wie schon bei der Wurzel. Zuerst müssen die Kinder von B abgearbeitet werden. E, als Kind von B bekommt *E.lft* = 3. Da E ein Blatt ist wird *E.rght* = 4. Falls E kein Blatt gewesen wäre, hätten erst die Kinder von E abgearbeitet werden müssen, bevor *E.rght* gesetzt wird. Nun wird das nächste Kind von B behandelt. *F.lft* wird 5, da F wiederum ein Blatt ist, wird *F.rght* = 6. Da B keine weiteren Kinder hat, wird *B.rght* = 7. Nun werden die anderen Kinder von A entsprechend behandelt. Zuerst wird *C.lft* = 8 und *C.rght* = 9, das nächste Kind D bekommt die Werte *D.lft* = 10, *D.rght* = 11. Da A nun keine weiteren Kinder mehr hat, wird *A.rght* = 12. Da A die Wurzel des Baumes ist, ist der Baum nun fertig

¹⁴Nähere Informationen zum Nested Sets Modell finden sich im Internet unter <http://www.intelligententerprise.com/001020/celko.shtml>, in den Artikeln "Weltenbaum" im PHP Magazin 02.2002 und "Wurmloch" im PHP Magazin 04.2003. Diese basieren auf dem Buch von J.Celko (S.431ff).

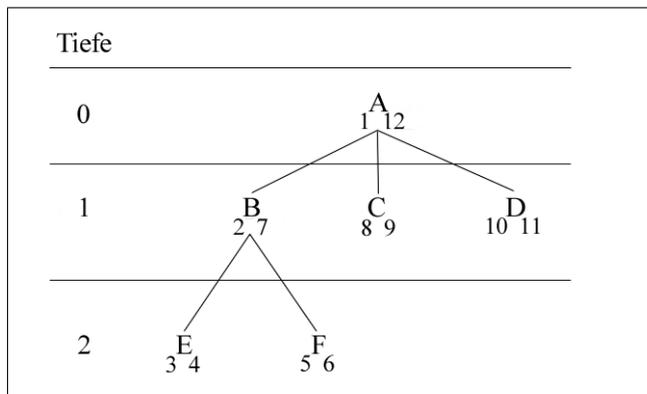


Abbildung 4.1: Beispiel Nested Sets Struktur

Knoten	lft	rght
A	1	12
B	2	7
E	3	4
F	5	6
C	8	9
D	10	11

Abbildung 4.2: Beispiel Nested Sets Tabellenform

durchnummeriert.

Jetzt wird ersichtlich, warum das Modell Nested Sets wörtlich übersetzt "verschachtelte Mengen" heißt. Jeder Knoten bildet mit seinen Kindern eine Menge und ist eine Teilmenge seiner Vorgänger. Die *lft* und *rght* Werte eines Knotens schließen alle Knoten ein, die innerhalb des untergeordneten Teilbaumes liegen.

Die Nachteile der Rekursivität bei Abfragen und der Tiefenbegrenzung, die beim Parent-Modell und Pfad-Modell auftreten, sind die Vorteile des Nested Sets Modells. So wird ersichtlich, dass die Tiefe keine Rolle spielt, da nur die Anzahl der Knoten eine Rolle spielt, die begrenzt ist durch den Typ des *lft* und des *rght* Feldes. Wenn der Typ der Felder INT und UNSIGNED ist, liegt die Begrenzung bei $\frac{2^{32}-1}{2}$ vielen Knoten. Bei BIGINT und UNSIGNED, liegt die Begrenzung bei $\frac{2^{64}-1}{2}$ vielen Knoten. Da schon beim Typ INT über 2 Milliarden viele Knoten gespeichert werden können, ist dies keine wirkliche Begrenzung.

Im Folgenden werden die bei EDEJU benötigten Strukturveränderungen und Abfragen besprochen, im Anschluß daran noch die Nachteile und Vorteile des Nested Sets Modell.

Knoten einfügen

Um einen Knoten einzufügen ist es notwendig, alle *lft* bzw. *rght* Werte, die größer als die Werte des einzufügenden Knotens sind, um 2 zu erhöhen, da um einen Knoten in die Struktur einzufügen 2 Werte (*lft*, *rght*) benötigt werden. Zuerst wird der *rght* Wert des Vaters ermittelt, dann werden alle *lft* Werte, die größer als dieser Wert sind um 2 erhöht. Anschließend werden die *rght* Werte erhöht die größer oder gleich dem *rght* Wert sind. Am Schluß wird der neue Knoten eingefügt, der *lft* Wert ist der alte *rght* Wert des Vaters und der neue *rght* Wert ist der alte *rght* Wert + 1.

Ein kleines Beispiel macht dies klarer: in Abbildung 4.3 soll an B der Knoten F gehängt werden. Bevor jedoch ein Knoten in die Datenbank eingefügt wird, muss erst die Struktur des bestehenden Baumes verändert werden. In einem ersten Schritt (Abbildung 4.4) werden die *lft* Werte um 2 erhöht, die größer als B.*rght*, d.h. die größer als 5 sind. Im nächsten Schritt (Abbildung 4.5) werden die *rght* Werte um 2 erhöht, die größer gleich B.*rght*, d.h. 5 sind. Erst dann (Abbildung 4.6) kann der neue Knoten in die Baumstruktur

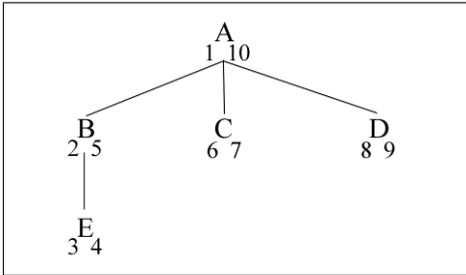


Abbildung 4.3: Knoten einfügen

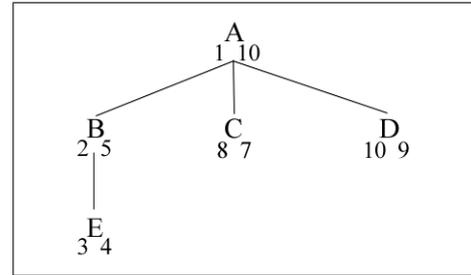


Abbildung 4.4: Schritt 1

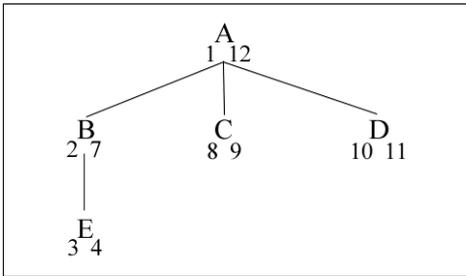


Abbildung 4.5: Schritt 2

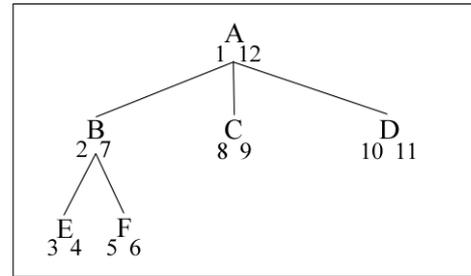


Abbildung 4.6: Schritt 3

eingefügt werden, mit $F.lft = B.right$ und $F.rght = B.rght + 1$ und $level = F.level + 1$. Diese Update-Operationen sind notwendig, damit nach dem Einfügen eines Knotens eine korrekte Nested Sets Struktur entsteht.

Mithilfe dieser 3 SQL-Queries wird ein neuer Knoten in die Datenbank eingefügt und die Struktur angepasst.

```
UPDATE tree SET lft=lft+2
WHERE lft > B.rght
```

```
UPDATE tree SET rght=rght+2
WHERE rght >= B.rght
```

```
INSERT INTO tree (... , level , lft , rght)
VALUES (... , B.level + 1 , B.rght , B.rght + 1)
```

Knoten löschen

Wenn ein Knoten A gelöscht werden soll, muss geklärt sein, was mit den Kindern von A passieren soll. Es gibt 2 Möglichkeiten: zum einen kann der gesamte Teilbaum gelöscht werden oder nur A wird gelöscht und die Kinder von A werden an den Vater von A gehängt. Bei EDEJU wird nur A gelöscht, die Kinder von A werden an den Vater von A angehängt. Ein Sonderfall ist das Löschen der Wurzel, da nicht klar ist welcher Knoten die neue Wurzel sein soll. Dieser Sonderfall tritt bei EDEJU aber nicht auf, da die Wurzel einen besonderen Knoten darstellt.

Zuerst wird der Knoten A gelöscht, das entstandene Loch in der Nested Sets Struktur muss nun wieder "gestopft" werden. Zuerst müssen die Kinder von A an den Vater von A gehängt werden. Die lft , $rght$ und $level$ Werte der Kinder, d.h. aller Knoten deren lft bzw. $rght$ Wert zwischen A.lft und A.rght liegen, werden um 1 erniedrigt. Dann müssen die

Knoten deren *lft* bzw. *rght* Werte größer als A.*rght* sind, geändert werden. Die *lft* Werte dieser Knoten werden um 2 und die *rght* Werte um 2 erniedrigt.

Die benötigten SQL - Queries dazu sehen folgendermaßen aus:

```
DELETE FROM tree
WHERE Knoten = A

UPDATE tree SET level=level-1, lft=lft-1, rght=rght-1
WHERE lft BETWEEN A.lft AND A.rght

UPDATE tree SET lft=lft-2
WHERE lft > A.rght

UPDATE tree SET rght=rght-2
WHERE rght > A.rght
```

Teilbäume verschieben

Eine weitere Änderung der Struktur ist das Verschieben von Teilbäumen. Das Verschieben von Teilbäumen beim Nested Sets Modell ist etwas komplexer.

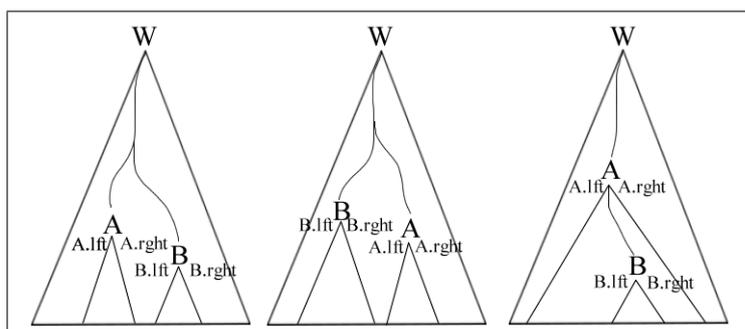


Abbildung 4.7: 3 Fälle, Teilbäume zu verschieben

Es soll als Beispiel der Teilbaum mit der Wurzel B verschoben werden. Der Knoten, an den dieser Teilbaum gehängt werden soll ist A. Es gibt nun 3 verschiedene Fälle von möglichen Verschiebungen. In Abbildung 4.7 sind diese dargestellt.

Beim ersten Fall liegt der Knoten A, an den der Teilbaum gehängt werden soll links vom Vater B des Teilbaumes. D.h. der Wert A.*rght* ist kleiner als B.*lft*. Entsprechend ist der zweite Fall: der Knoten A liegt rechts vom Teilbaum mit dem Vater B, d.h. A.*lft* ist größer als B.*rght*.

Beim dritten Fall ist der Teilbaum mit Vater B eine Teilmenge des Teilbaumes von A. Dies bedeutet, dass der Wert A.*lft* kleiner als B.*lft* und der Wert A.*rght* größer als B.*rght* ist. Diese 3 Fälle sind beim Update der Struktur zu beachten. Ausserdem ist es notwendig, auszuschließen, dass ein Teilbaum in seinen eigenen Teilbaum gehängt wird, da dies ein Sonderfall ist und nicht klar ist, ob der Knoten, an den der Teilbaum gehängt wird, die neue Wurzel des Teilbaumes sein soll oder nicht.

Um doppelte *lft*, *rght* Werte zu vermeiden, wird zuerst der *rght* Wert der Wurzel *W* benötigt. Der zu verschiebende Teilbaum wird zuerst nach links ausserhalb der Struktur verschoben (siehe Abbildung 4.8). Alle *lft* bzw. *rght* Werte der Knoten, die im Teilbaum von *B* liegen, werden um $(W.rght - B.lft + 1)$ erhöht. Zusätzlich werden die *level* Werte des Teilbaumes gleich richtig eingetragen. Dies geschieht mit $level = level + (A.level - B.level + 1)$. Die Verschiebung ist notwendig, da zuerst die übrige Struktur angepasst werden muss und ansonsten doppelte *lft* bzw. *rght* Werte zu großen Problemen führen und die Struktur zerstören würden. Das "Loch" in der Struktur, welches durch das Verschieben der Teilbaumes von *B* entsteht, muss an die entsprechende Stelle im Teilbaum von *A* verschoben werden, damit der Teilbaum *B* an der richtigen Stelle wieder eingefügt werden kann. Mit diesem SQL-Query wird der Teilbaum *B* aus der Struktur herausgenommen.

```
UPDATE tree
SET lft = lft + W.rght - B.lft + 1,
    rght = rght + W.rght - B.lft + 1,
    level = level + A.level - B.level + 1
WHERE lft >= B.lft AND rght <= B.rght
```

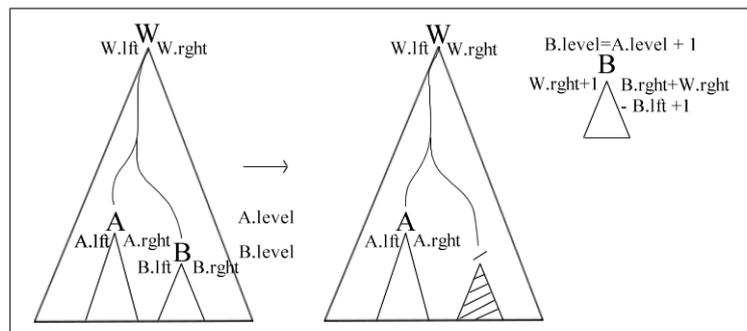


Abbildung 4.8: Teilbaum B aus der Struktur entfernen

Nachdem der Teilbaum von *B* verschoben wurde, kann die restliche Struktur abhängig von der Art der Verschiebung geändert werden.

Im ersten Fall werden zuerst die *lft* Werte geändert. Alle Knoten, deren *lft* Wert größer als *A.rght* und kleiner als *B.lft* ist, werden um $(B.rght - B.lft + 1)$ erhöht. Dann werden die *rght* Werte, die größer gleich *A.rght* und kleiner *B.lft* sind, um $B.rght - B.lft + 1$ erhöht. Nun wird der zu verschiebende Teilbaum, für den nun Platz geschaffen wurde, wieder in die Struktur eingefügt. Alle *lft*, *rght* Werte der Knoten, die größer als *W.rght* sind, werden um $(A.rght - W.rght - 1)$ erniedrigt. Damit wird die Baumstruktur wieder hergestellt. Die benötigten SQL-Queries für den ersten Fall sind im Folgenden dargestellt:

```
UPDATE tree SET lft = lft + (B.rght - B.lft + 1)
WHERE lft > A.rght AND lft < B.lft
```

```

UPDATE tree
SET  rght = rght + (B.rght - B.lft + 1)
WHERE rght >= A.rght AND rght < B.lft

UPDATE tree SET lft = lft + A.rght - W.rght - 1
        rght = rght + A.rght - W.rght - 1
WHERE lft > W.rght

```

Im zweiten Fall wird zuerst der Teilbaum, analog zum ersten Fall, ganz nach rechts verschoben und der Level angepasst. Dann wird die restliche Struktur entsprechend angepasst. Zuerst werden die *lft* Werte bei allen Knoten um $(A.rght - B.lft + 1)$ erniedrigt, deren *rght* Wert kleiner gleich $A.rght$ und deren *lft* Wert größer gleich $B.lft$ ist. Die *rght* Werte werden entsprechend angepasst. Dann wird der Teilbaum von A um $(A.rght - B.rght - W.rght + B.lft - 2)$ erniedrigt. Auf diese Weise wird der Teilbaum an die Stelle geschoben, die vorher für ihn frei gemacht wurde.

Die benötigten SQL-Queries für den zweiten Fall sind im Folgenden dargestellt:

```

UPDATE tree SET lft = lft - (B.rght - B.lft + 1)
WHERE rght <= A.rght AND lft > B.lft

UPDATE tree
SET  rght = rght + (B.rght - B.lft + 1)
WHERE rght < A.rght AND rght > B.rght

UPDATE tree SET lft = lft + A.rght - B.rght - W.rght + B.lft - 2
        rght = rght + A.rght - B.rght - W.rght + B.lft - 2
WHERE lft > W.rght

```

Im dritten Fall wird der Teilbaum analog zu den beiden anderen Fällen nach links verschoben und der Level angepasst. Dann werden die *lft* Werte der Knoten um $(B.rght - B.lft + 1)$ erniedrigt, deren *rght* Wert kleiner $A.rght$ ist und deren *lft* Wert größer $B.rght$ ist. Die *rght* Werte der Knoten, deren *rght* Wert kleiner $A.rght$ und deren *rght* Wert größer $B.rght$ ist, werden entsprechend erniedrigt. Der Teilbaum wird entsprechend dem zweiten Fall wieder in die Struktur eingliedert.

Die benötigten SQL-Queries für den dritten Fall sind im Folgenden dargestellt:

```

UPDATE tree SET lft = lft - (B.rght - B.lft + 1)
WHERE rght < A.rght AND lft > B.rght

UPDATE tree SET rght = rght - (B.rght - B.lft + 1)
WHERE rght < A.rght AND rght > B.rght

UPDATE tree SET rght = rght + A.rght - B.rght - W.rght + B.lft - 2
        lft = lft + A.rght - B.rght - W.rght + B.lft - 2
WHERE lft > W.rght

```

Anhand des folgenden kleinen Beispiels wird dies deutlicher. Ausgehend von der Abbildung 4.9 soll der Knoten B samt Teilbaum unter den Knoten A verschoben werden. Dazu wird zunächst Teilbaum B aus der Struktur herausgenommen und die *level* Werte geändert. Danach werden die *lft* und *rght* Werte in der Struktur geändert (siehe Abbildung 4.10). Schließlich wird der Teilbaum B wieder in das verschobene "Loch" in der Struktur eingefügt (siehe Abbildung 4.11).

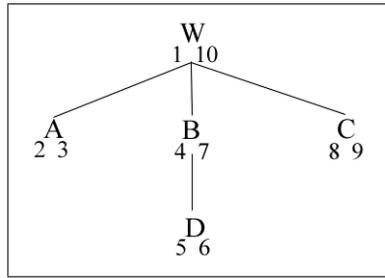


Abbildung 4.9: Beispiel: Teilbaum verschieben

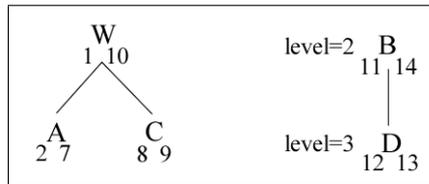


Abbildung 4.10: Beispiel: Teilbaum verschieben, Schritt 1,2

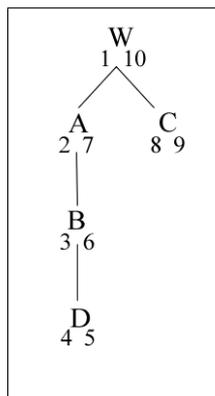


Abbildung 4.11: Beispiel: Teilbaum verschieben, Schritt 3

Abfragen

Während die Änderungen der Datenbank, auf deren Probleme im nächsten Abschnitt eingegangen wird, unter Umständen einen großen Aufwand machen, sind Abfragen sehr geschickt zu realisieren.

Eine Abfrage, die bei EDEJU sehr oft vorkommt, ist, die Vorgänger eines Knotens *A* auszugeben. Mit folgendem Query lassen sich alle Vorgänger sortiert ausgehend von der Wurzel aus der Datenbank holen.

```
SELECT *
FROM tree
WHERE lft < A.lft AND rght > A.rght
ORDER BY lft
```

Um nur den Vater des Knotens *A* zu erhalten, wird an den WHERE Block folgendes angehängt:

```
AND level = A.level - 1.
```

Die Tiefe eines Knotens lässt sich mit Hilfe von *lft* und *rght* berechnen. Da aber die Tiefe eines Knotens sehr leicht während der Einfüge-, Entferne- und Updateoperationen angepasst werden kann und bei EDEJU der Level für viele Abfragen benötigt wird, wird der Level als zusätzliches Feld *level* zu *lft* und *rght* mit abgespeichert. Zur Vervollständigung wird die Berechnung hier angegeben.

```
SELECT n.*, COUNT(*)-1 AS level
FROM tree n, tree o
WHERE n.lft
BETWEEN o.lft AND o.rght
GROUP BY n.lft
ORDER BY n.lft
```

Teilbäume lassen sich mithilfe des Nested Sets Modell sehr einfach berechnen. Um alle Knoten eines Teilbaumes im Teilbaum *A* zu ermitteln, ist der folgende Query ausreichend:

```
SELECT *
FROM tree
WHERE lft > A.lft AND rght < A.rght.
```

Um alle Kinder eines Knotens *A* auszugeben, kann die obige Berechnung eines Teilbaumes benutzt werden. Zusätzlich muss nur eine weitere Bedingung an den WHERE Block gehängt werden:

```
AND level = A.level + 1.
```

Nachteile und Vorteile

Ein großer Nachteil des Nested Sets Modell ist, dass die Datenbank für die Einfüge-, Entferne- und Update-Operationen, die Struktur der Bereiche verändern, von keinem anderen Benutzer gleichzeitig verändert werden darf. Die Baumstruktur kann ansonsten zerstört werden. Da Transaktionen in der verwendeten Version von MySQL fehlen, ist es notwendig, die Tabelle zu sperren. Ansonsten wäre es nicht möglich, zu garantieren, dass die SQL-Queries als Einheit ausgeführt werden. Dies würde die ACID¹⁵ (atomar, konsistent, isoliert, dauerhaft) Eigenschaften, welche die Transaktionen erfüllen müssen, verletzen. Allerdings sind diese Eigenschaften durch das Sperren der Relation allein nicht gewährleistet. Die Eigenschaft Atomarität und damit auch die Konsistenz kann durch das Sperren, nicht gewährleistet werden, da es kein Rollback bzw. Commit in der aktuellen Version gibt. Aufgetretene Fehler können somit nicht mehr rückgängig gemacht werden. Allerdings sind Transaktionen ab der MySQL Version 4.0.11 möglich und können und sollten dann anstatt der Sperren verwendet werden.

Ein weiterer Nachteil ist, dass im schlechtesten Fall alle Einträge verändert werden müssen und dies einen großen Rechenaufwand darstellt. Da aber bei EDEJU die Struktur nur selten geändert wird und für die Änderung von Bereichen nur wenige Benutzer zuständig sein werden, sind diese Nachteile zu verkraften. Allerdings wäre z.B. bei Diskussionsforen, die eine Baumstruktur haben, diese Art der Speicherung aufgrund der Vielzahl von Änderungen der Struktur nicht sinnvoll.

Die Vorteile dieses Modells überwiegen für diesen Anwendungsfall die Nachteile. Der Hauptbestandteil von EDEJU sind Ausgaben der Daten, die in der Datenbank eingetragen sind. Es ist daher notwendig Abfragen an die Struktur so schnell wie möglich beantworten zu können¹⁶. Aus diesem Grund ist dieses Modell besser geeignet als z.B. das Parent-Modell, da keine rekursiven Abfragen notwendig sind. Ein weiterer Vorteil ist die Flexibilität in der Größe und Tiefe des Baumes, da im Gegensatz zum Pfad-Modell nur die Anzahl der Knoten eine Rolle spielt und nicht die Größe eines Feldes.

4.5.3 Tabellenstruktur der Bereiche bei EDEJU

Die Bereiche werden in 2 Tabellen in der Datenbank gespeichert. In der Tabelle *edeju_bereiche* werden alle Informationen über die Bereiche in den Feldern *bezeichnung*, *beschreibung*, *bild* und *datum* gespeichert. Gleichzeitig ist noch die Baumstruktur mit den Feldern *lft*, *rght* und *level* abgespeichert.

Die Querverweise werden in der Tabelle *edeju_bereiche_querverweise* abgelegt. Zusätzlich zu den zwei IDs der verbundenen Bereiche *bereich1ID* und *bereich2ID* kann noch eine Beschreibung im Feld *beschreibung* hinzugefügt werden. Dies ist sinnvoll, da der Zusammenhang der beiden Bereiche darin genauer erläutert werden kann, da er in vielen Fällen ansonsten nicht immer erkennbar ist.

¹⁵vgl. R.Ramakrishnan/J.Gehrke S.523ff

¹⁶siehe Vergleich der Modelle im PHP Magazin 04.2003 S.87

4.6 Datenverwaltung

Im Abschnitt 4.5 wurde die Datenstruktur der Bereiche genauer erläutert. Diese Struktur bildet den Kern von EDEJU. Nun ist es möglich, anhand der Bereiche Profile zu erstellen und Projekte bzw. Lernmittel den Bereichen zuzuordnen.

4.6.1 Verwaltung

Zu den Daten der Verwaltung gehören neben den schon besprochenen Bereichen die Lernmittel, Altersgruppen und Muster-Kompetenzträger.

Um Lernmittel beschreiben zu können, wird zunächst die Basisrelation mit den Informationen benötigt. Die Relation *edeju_lernmittel* enthält die Felder *bezeichnung*, *beschreibung*, *link* und *bild*. Zusätzlich zu dieser Relation sind noch 3 Relationen notwendig, um Lernmittel in EDEJU einzuordnen. Die Beziehung zu den Bereichen wird in der Relation *edeju_lernmittel_bereiche* festgelegt. Enthalten sind die Schlüssel des jeweiligen Bereichs bzw. des Lernmittels. Die Ansprechpartner zu einem Lernmittel werden in der Relation *edeju_lernmittel_ansprechpartner* gespeichert. Darin werden die jeweiligen Schlüssel gespeichert und es wird vermerkt, wer das Lernmittel in die Datenbank eingetragen hat. Zusätzlich kann ein Lernmittel noch verschiedenen Altersgruppen zugeordnet werden, dies geschieht in der Relation *edeju_lernmittel_altersgruppe*. Es gibt 2 verschiedene Rechte in EDEJU, die den Benutzern erlauben, Lernmittel einzutragen. Zum einen gibt es *rig_lernmittel*, welches dem Benutzer erlaubt, alle Lernmittel zu bearbeiten und neue einzutragen. Das andere Recht *rig_eigene_lernmittel* erlaubt dem Benutzer, Lernmittel einzutragen und nur diese zu bearbeiten.

Mit dem Recht *rig_altersgruppe* kann der Benutzer Altersgruppen in der Relation *edeju_altersgruppe* eintragen und bearbeiten. In dieser Relation enthalten sind ein *von*- und ein *bis*-Alter und eine kurze Beschreibung. Ein Sonderfall ist eine Altersgrenze, die nach oben offen ist. Dazu muss einfach das *bis*-Alter kleiner sein als das *von*-Alter, damit unterschieden werden kann, dass hier bewusst die Grenze nach oben offen ist.

Ein weiterer Punkt, der zur Verwaltung der EDEJU Daten gehört, ist das Bearbeiten von Muster-Kompetenzträgern. Dazu wird das Recht *rig_musterkompetenz* benötigt. In der Abbildung 3.2 ist das ER-Modell von Kompetenzträgern zu sehen. Im Laufe des Aufbaus von EDEJU hat sich gezeigt, dass nicht nur Landeskultur und Berufe Muster-Kompetenzträger sein können, sondern auch Vorbilder, wie Personen aus der Politik oder dem Sport. Da es nicht möglich war, diese verschiedenen Arten von Muster-Kompetenzträgern zu begrenzen, gibt es eine zusätzliche Relation *edeju_muster_art*. Diese Relation enthält die verschiedenen Arten von Muster-Kompetenzträgern. Mit den Attributen *Bezeichnung* und *Beschreibung* kann die Art genauer erläutert werden. Die allgemeinen Daten eines Muster-Kompetenzträgers werden in der Relation *edeju_muster* eingetragen. Mit Hilfe der Attribute *Bezeichnung*, *Beschreibung*, *Link*, *Bild* und *Datum* werden die Informationen gespeichert. Zusätzlich wird das Feld *musterID* hinzugefügt. Dieses Feld enthält den Schlüssel aus der Relation *edeju_muster_art* und gibt an, welche Art von Muster gemeint ist. Das Profil eines Musters wird in folgenden Relationen eingetragen:

- *edeju_muster_bereich_interessen*,

- *edeju_muster_bereich_wissen*,
- *edeju_muster_bereich_koennen*,
- *edeju_muster_lernmittel_interessen*,
- *edeju_muster_lernmittel_wissen*
- *edeju_muster_lernmittel_koennen*.

Diese Relationen sind genauso aufgebaut, wie das Profil eines Benutzers. Daher wird das Aussehen dieser Relationen erst im nächsten Abschnitt näher erklärt. Ansprechpartner zu Muster-Kompetenzträgern werden in der Relation *edeju_muster_ansprechpartner* eingetragen. Diese ist genauso aufgebaut, wie die Ansprechpartnerrelation der Lernmittel.

4.6.2 Benutzer

Im ER-Modell (siehe Abbildung 3.2) kann ein Benutzer eine Firma, ein Verein, eine Institution oder eine natürliche Person sein. Dies wurde reduziert auf zwei Arten von Benutzern, die innerhalb einer Relation gespeichert werden können. Innerhalb der Relation *edeju_user_daten* werden die persönlichen Daten und die Art des Benutzers eingetragen. Die Auswahl geschieht über ein ENUM(0,1)¹⁷ Feld *firma*, welches zwischen natürlichen Personen und Institutionen unterscheidet. Eine Institution kann dabei ein Verein, eine Firma usw. sein. Falls das Feld *firma* gleich 1 ist, kann ein Firmen- oder Vereinsname eingetragen werden.

Zu den Benutzerdaten gehören auch die persönlichen Daten, die eigenen Projekte und das eigene Profil. Die persönlichen Daten eines Benutzers werden in der Relation *edeju_user_daten* gespeichert. Der Benutzer muss diese Daten nicht alle ausfüllen. Um als angemeldeter Benutzer zu gelten, reicht es aus, dass er in EDEJU einen Benutzernamen und eine gültige Emailadresse angegeben hat, die in der Relation *rechte_rights* gespeichert sind. Zusätzlich dazu kann er folgende Felder eintragen: *Vorname*, *Nachname*, *Strasse*, *Plz*, *Ort*, *Land*, *Nationalität*, *Telefon*, *Fax*, *Internetseite*, *Geburtstag*, *Beruf*, *Beschreibung* und er hat Möglichkeit ein *Bild* hochzuladen. Falls der Benutzer eine Institution ist, enthält der Vor- und Nachname den Ansprechpartner innerhalb der Firma bzw. des Vereins der für den Eintrag in EDEJU verantwortlich ist. Felder, wie Nationalität und Geburtstag sind in diesem Fall irrelevant.

Um Projekte eintragen zu können, muss der Benutzer das Recht *rig_projekte* besitzen, welches ihm von einem User-Administrator zugewiesen wurde. Projekte werden in der Relation *edeju_projekte* gespeichert. Die Felder *Bezeichnung*, *Beschreibung*, *Link*, *Termin_von*, *Termin_bis*, *Ort*, *Land*, *Region* und *Bild* ermöglichen eine umfassende Beschreibung von Projekten. Zusätzlich gibt es das ENUM(0,1) Feld *oeffentlich*, welches dem Benutzer erlaubt Projekte einzutragen, die noch kein anderer Benutzer sehen darf. Dies ist z.B. sinnvoll, wenn das Projekt noch im Aufbau ist und noch nicht alle Daten eingetragen werden können. Da an Projekten mehrere Benutzer beteiligt sein können gibt es noch ein

¹⁷ENUM ist ein Spaltentyp für Felder, die nur bestimmte Werte annehmen sollen. In diesem Fall enthält das Feld entweder eine 0 oder eine 1.

ENUM(0,1) Feld *mitmachen*. Damit kann der Benutzer zulassen, dass andere Benutzer sich bewerben können, an diesem Projekt teilzunehmen. Der Benutzer soll allerdings die Möglichkeit haben, Bewerber abzulehnen. Die Beziehung von Projekten mit Benutzern wird in der Relation *edeju_projekte_user* gespeichert. Darin werden die jeweiligen Schlüssel des Projekts und des Benutzers gespeichert. Zusätzlich gibt es das Feld ENUM(0,1) *owner*, das auf 1 steht, falls der Benutzer der Eigentümer ist. Ein zusätzliches ENUM(0,1) Feld *anfrage* markiert Benutzer, die sich an einem Projekt beteiligen wollen, die aber noch nicht akzeptiert wurden. Dem Benutzer soll, wenn sich Personen an seinem Projekt beteiligen wollen, immer automatisch eine Benachrichtigungsmail geschickt werden. Das Profil eines Benutzers wird in den folgenden Relationen gespeichert:

- *edeju_user_bereich_wissen*,
- *edeju_user_bereich_interessen*,
- *edeju_user_bereich_koennen*,
- *edeju_user_lernmittel_wissen*,
- *edeju_user_lernmittel_interessen*,
- *edeju_user_lernmittel_koennen*.

Diese Relationen sind genauso aufgebaut, wie die Relationen der Muster-Kompetenzträger, die das Profil enthalten. Die Relationen enthalten die jeweiligen Schlüssel des Benutzers und des Bereichs bzw. Lernmittels. Zusätzlich kann der Benutzer jeweils noch eine Beschreibung einfügen. In den Feldern *eingefuegt* und *geupdatet* werden jeweils die Daten hinzugefügt, wann der Eintrag erfolgt ist und wann er zuletzt geändert wurde.

Ein Benutzer kann zusätzlich Lernmittel anbieten. Diese Angebote werden in der Relation *edeju_user_lernmittel_angebote* eingetragen und enthalten neben den Schlüsseln, des Benutzers und des Lernmittels, eine Beschreibung. Zusätzlich soll es möglich sein, zu jedem Eintrag in diese Relationen noch ein Bild hinzuzufügen.

Dem Benutzer muss ermöglicht werden, seine Einträge bearbeiten und löschen zu können. Ob die persönlichen Daten oder das Profil eines Benutzers freigegeben werden soll, d.h. dass auch andere Benutzer Einsicht in diese Daten haben, kann jeder selbst entscheiden. Dazu existieren in der Relation *edeju_user_daten* zwei zusätzliche ENUM(0,1) Felder *freigabe_adresse* und *freigabe_profil*. Durch diese Trennung von Profil und eigenen Daten kann der Benutzer beispielsweise sein Profil freigeben, aber nicht seine Adresse.

4.7 Funktionen im öffentlichen Bereich

Funktionen im öffentlichen Bereich dienen der Ausgabe jener Daten, die in der Datenbank enthalten sind und öffentlich zugänglich sind. Die Funktionen sind strukturiert in "Allgemeine Funktionen", "Vorschläge", "Vergleiche" und die "Suche innerhalb der Datenbank".

4.7.1 Allgemeine Funktionen

Die Bereiche, als zentrale Basis von EDEJU, spielen bei der Navigation durch die Daten eine große Rolle. Denkbar sind viele Ausgaben, wie diese Struktur ausgegeben werden kann. Zwei Arten der Ausgabe wurden im Rahmen dieser Diplomarbeit ausgearbeitet.

Zum einen kann die Datenbank "durchstöbert" werden, d.h. ausgehend von der Wurzel der Bereiche ist es dem Benutzer möglich, über die Verknüpfungen mit untergeordneten Bereichen, je nach Interesse tiefer in die Struktur der Bereiche vorzudringen. Dabei ist zu beachten, dass es eine Möglichkeit geben muss, wieder rückwärts, d.h. über den Vorgänger von Bereichen, sich wieder nach "oben" zu bewegen.

Die andere berücksichtigte Form der Ausgabe ist eine "Bereichsmap". Im Prinzip ist diese Ausgabe ähnlich, allerdings hat der Benutzer einen größeren Überblick, wohin er sich bewegt. In der Bereichsmap kann der Benutzer auswählen, bis zu welcher Tiefe, ausgehend von einem bestimmten Bereich, er die Bereiche betrachten will. Diese Form der Ausgabe wird durch die große Anzahl der Bereiche sehr schnell unübersichtlich.

Die Ausgabe eines Bereichs ist aufgeteilt in die Daten, die den Bereich charakterisieren, wie z.B. die Bezeichnung und die Beschreibung. Zusätzlich müssen neben den Unterbereichen und den Querverweisen auch die Lernmittel, Altersgruppen und Projekte, die mit diesem Bereich verknüpft sind, ausgegeben werden. Ausserdem kann der Benutzer nach Personen suchen, die in diesem Bereich *Interessen*, *Wissen* oder *Können* haben. Dabei ist zu beachten, dass nur Personen ausgegeben werden, die ihr Profil freigegeben haben. Auf diese Weise kann der Benutzer mit anderen Personen Kontakt aufnehmen, die z.B. die gleichen Interessen haben.

Um den Benutzern die Möglichkeit zu geben die Informationen, die einen Bereich beschreiben ausserhalb des Computers zu nutzen, gibt es eine Druckversion der Bereiche, in der die benötigten Informationen druckerfreundlich aufbereitet sind. Druckerfreundlich bedeutet, dass Bilder und Menüs entfernt werden.

Bei der Ausgabe der Lernmittel werden zusätzlich zu den Informationen des Lernmittels noch die Altersgruppen, Bereiche und Projekte mit denen das Lernmittel verknüpft ist ausgegeben. Angebote rund um das Lernmittel, welche die Benutzer von EDEJU eingetragen haben werden hier ausgegeben.

Bei der Ausgabe von Benutzern muss immer überprüft werden, ob der Benutzer seine Daten freigegeben hat. Es gibt 4 Arten der möglichen Freigabe: der Benutzer hat nichts freigegeben, weder sein Profil noch seine Daten, darum kann dieser Benutzer nicht öffentlich ausgegeben werden. Bei den anderen 3 Freigabeformen werden nur die freigegebenen Teile ausgegeben. Der Benutzer kann alle Daten freigeben, dann werden seine persönlichen Daten und sein Profil öffentlich ausgegeben. Darunter fallen alle persönlichen Daten aus der Relation *edeju_user_daten* und sein *Interesse*, *Wissen* und *Können*.

Bei Projekten und Angeboten wird immer der Name der Person veröffentlicht. Mit Hilfe des Kontaktsystems ist es möglich, ohne die Veröffentlichung von weiteren Daten, mit dem Benutzer direkt Kontakt aufzunehmen.

Um mit anderen Benutzern von EDEJU in Kontakt zu kommen, wird ein Kontaktsystem verwendet, bei dem der Benutzer, der zuerst mit einem anderen Benutzer Kontakt aufnimmt, keinerlei Kontaktdaten ausser dem Namen sehen kann. Auf diese Weise können Unbefugte nicht an Emailadressen von Benutzern gelangen. Das Kontaktsystem funktioniert wie folgt: bei der Ausgabe einer Person gibt es immer einen "Kontakt-Link". Beim Klicken auf diesen Link öffnet sich ein Fenster, in dem ein HTML Formular mit einer Betreffzeile, einem Email Feld und einem Feld für die eigentliche Nachricht vorhanden ist. Der Benutzer kann diese Felder ausfüllen. Da der Benutzer nun seine Emailadresse eintragen muss (falls der Benutzer eingeloggt ist geschieht dies automatisch), ist der Empfänger dieser Nachricht in der Lage, mit dem anfragenden Benutzer Kontakt aufzunehmen. Auf diese Weise wird eine Verwendung der Emailadresse z.B. für Spam verhindert. Dadurch, dass die Emailadresse nicht direkt ausgegeben wird, können z.B. Spiderprogramme nicht in Besitz der Email Adresse kommen.

Dieses Kontaktsystem wird sowohl im öffentlichen, wie auch im Mitglieder-Bereich eingesetzt.

4.7.2 Vorschläge

Vorschläge dienen dem Benutzer dazu, Bereiche zu finden, die für ihn interessant sind und eine sinnvolle Ergänzung seines Profils sein können. Es gibt 3 Arten von Vorschlägen, die bei EDEJU angedacht sind:

- "Willkürliche Vorschläge"
- "Ähnliche Vorschläge"
- "Weiße Flecken".

"Willkürliche Vorschläge" lassen sich mit Hilfe von MySQL sehr einfach bewerkstelligen. Mit dem folgendem SQL-Query werden 5 willkürliche Bereiche als Vorschläge ausgegeben.

```
SELECT *  
FROM edeju_bereiche  
ORDER BY RAND()  
LIMIT 0,5
```

Natürlich ist es möglich, auch Lernmittel auf diese Weise auszugeben, darauf wird bei EDEJU aber verzichtet. Es werden automatisch 5 Lernmittel, die mit den ausgegebenen Bereichen verknüpft sind, ausgegeben. Auf diese Weise ist ein gewisser Zusammenhang innerhalb eines Willkürlichen Vorschlags vorhanden.

Ähnliche Vorschläge sollen Bereiche ausgeben, die für den Nutzer relevante Themen darstellen könnten. Bei EDEJU kommen alle Bereiche in Frage, die mit einem Bereich, der im Profil, sei es als *Wissen*, *Können* oder *Interesse* eingetragen ist, "näher verwandt" sind.

Das bedeutet, dass alle Kinder der eingetragenen Bereiche, alle Vorgänger plus deren Kinder und alle Querverweise plus deren Kinder dafür in Frage kommen können. Natürlich kann diese Auswahl noch um Enkel, den Vorgänger des Vorgängers oder andere Bereiche erweitert werden, es besteht dann allerdings die Gefahr, dass die Vorschläge nichts mehr mit dem Profil des Benutzers zu tun haben und nicht mehr "ähnlich" sind.

Alle in Frage kommenden Bereiche werden gewichtet. Die Bewertung ist abhängig davon, ob der Bereich mit *Interesse*, *Wissen* oder *Können* verwandt ist. Verwandte mit *Interesse* bekommen den höchsten Wert 3, da dadurch die Wahrscheinlichkeit größer ist, einen Bereich zu finden, der für den Benutzer interessant ist. Verwandte mit *Können* bekommen den Wert 2 und mit *Wissen* den Wert 1. Da natürlich ein Bereich mehrmals in der Menge auftauchen kann, kann dieser verschiedene Bewertungen erhalten. Diese Bewertungen werden am Schluß summiert und die Bereiche mit der höchsten Bewertung werden ausgegeben. In Abbildung 4.12 ist ein Beispiel für die Bewertung dargestellt. Die Bereiche, welche als Vorschlag in Frage kommen sind A, D, E und H. Diese bekommen jeweils Werte zugewiesen, abhängig davon, mit welchen Bereichen aus dem Profil diese verwandt sind. Anschließend werden diese Werte addiert und sortiert ausgegeben. In diesem Fall wäre das Ranking: E mit 10 Punkten, A mit 7, D mit 6 Punkten und schließlich H mit 4 Punkten. Wie schon in der Grafik ersichtlich ist, ist E mit allen Bereichen aus dem Profil direkt verbunden. E wird daher wahrscheinlich eine sinnvolle Erweiterung zum vorhandenen Profil sein.

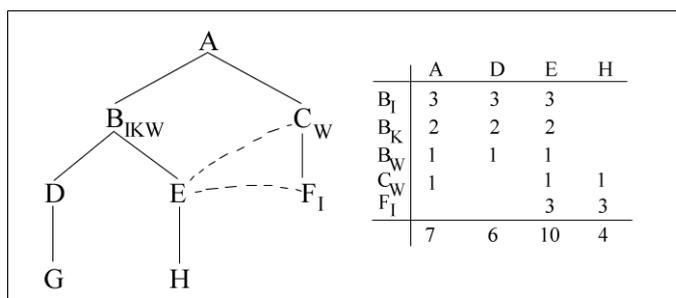


Abbildung 4.12: Ähnliche Vorschläge Bewertung

Die Berechnung der Vorschläge ist unter MySQL kompliziert und wird in Kapitel 5.2.2 genauer erläutert. Zusätzlich zu den Bereichen werden noch Lernmittel, die mit diesen Bereichen zu tun haben, wie bei den "Willkürlichen Vorschlägen", ausgegeben.

Weißer Flecken im Profil eines Benutzers sind Bereiche oder Teilbäume innerhalb der Bereichsstruktur, die im Profil nicht auftauchen. Für diese weißen Flecken kommen alle Bereiche, die nicht näher mit einem Bereich aus dem Profil verwandt sind, in Frage. Näher verwandt heißt, alle Bereiche, die nicht Vorgänger, Kind oder Enkel eines Bereichs aus dem Profil sind, gehören zu den in Frage kommenden Bereichen. Auf die Entfernung von Querverweisen wird verzichtet, da diese meist zu Bereichen aus anderen Themenblöcken verzweigen. Aus dieser Menge werden nun wieder randomisiert 5 Bereiche und dazugehörige Lernmittel ausgegeben. Bei dieser Berechnung treten wie bei den "Ähnlichen Vorschlägen" die gleichen Probleme auf. Diese werden in Kapitel 5.2.2 genauer erläutert.

Im Laufe der Nutzung von EDEJU wird sich zeigen, ob diese Heuristiken sinnvoll sind oder

erweitert bzw. durch neue ersetzt werden müssen. Da die Datenbasis noch sehr klein ist und erst im Laufe der Zeit anwachsen wird, ist es nicht möglich, im Rahmen dieser Arbeit diese Heuristiken zu optimieren.

4.7.3 Vergleiche

Jeder Benutzer, der ein eigenes Profil hat, kann dieses mit dem Profil anderer Benutzer oder Muster-Kompetenzträgern vergleichen. Sobald ein Benutzer eingeloggt ist und Personen oder Muster-Kompetenzträger betrachtet, hat er die Möglichkeit einen Vergleich zu starten.

Ein Vergleich ist aufgeteilt in drei Teile. Es wird jeweils das *Interesse*, das *Wissen* oder das *Können* mit dem Profil der anderen Person verglichen. Dabei wird für jeden Bereich ausgegeben, ob es Gemeinsamkeiten oder Ähnlichkeiten gibt. Wenn z.B. ein Benutzer als *Interesse* den Bereich Musik eingetragen hat und der andere Benutzer *Wissen*, wird hinter dem Bereich Musik *Wissen* ausgegeben. Dadurch wird klar, dass der andere Benutzer im Bereich Musik, in welchem der Benutzer *Interesse* aufweist, *Wissen* besitzt. Auf diese Weise kann der Benutzer z.B. bei bestimmten Fragen Kontakt zu anderen Benutzern aufnehmen. Falls keine Gemeinsamkeiten gefunden werden, werden verwandte Bereiche gesucht. Dabei gelten alle Kinder der Vorgänger und die Querverweise als verwandt. Für alle diese verwandten Bereichen wird überprüft, ob diese beim anderen Benutzer im Profil vorhanden sind. Falls dies der Fall ist, werden diese Bereiche als Ähnlichkeiten ausgegeben.

In der Abbildung 4.13 sind die Profile von 2 Benutzern dargestellt. Die zugrundeliegende Bereichsstruktur ist in Abbildung 4.12 dargestellt. In diesem Fall werden die Interessen von Benutzer 1 mit dem Profil von Benutzer 2 verglichen. Beim Bereich A gibt es Übereinstimmungen, so hat Benutzer 2 dort *Können* und *Wissen*. Bei Bereich B gibt es keine solchen Übereinstimmungen, daher wird nach Ähnlichkeiten gesucht. In Frage kommen da nur Bereich A und E, da diese beiden Bereiche in beiden Profilen vorkommen.

Benutzer 1		Benutzer 2		Vergleich	
A	I K W	A	K W	Interesse	Benutzer 2
B	I	D	I	A	K W
E	K	E	K	B	A, E
		C	I		

Abbildung 4.13: Vergleich zwischen 2 Benutzern

Natürlich ist es möglich und wahrscheinlich, dass in vielen Fällen für einige Bereiche keinerlei Ähnlichkeiten bzw. Gemeinsamkeiten gefunden werden. Allerdings kann der Benutzer das Profil des anderen betrachten und so vielleicht auf interessante Bereiche stoßen.

4.7.4 Suche

Die Anzahl der Suchmöglichkeiten, die bei EDEJU allein schon durch die Vielzahl von Verknüpfungen zwischen den Relationen möglich sind, werden durch die Miteinbeziehung von Attributen der einzelnen Relationen um ein Vielfaches erhöht. So könnte eine Suche

nach einem Projekt so aussehen: das Projekt sollte in einem bestimmten Zeitraum stattfinden, in einer bestimmten Region, sollte mit bestimmten Bereichen verknüpft sein und bestimmte Lernmittel verwenden.

Der Benutzer soll immer nach Bereichen, Lernmitteln, Personen und Projekten suchen können. Dazu muss er spezifizieren, nach was gesucht werden soll. Bei Projekten, Bereichen und Lernmitteln werden die Felder Bezeichnung und Beschreibung mit den Suchbegriffen verglichen und gegebenenfalls ausgegeben. Bei der Suche nach Personen werden Name, Vorname oder Firmenname mit den Suchbegriffen verglichen. Dabei werden in der Ergebnismenge nur Personen ausgegeben werden, die auch ihr Profil oder die persönlichen Daten freigegeben haben. Zusätzlich dazu kann bei den Bereichen nach Benutzern gesucht werden, die zu diesem Bereich *Interesse*, *Wissen* oder *Können* eingetragen haben.

Im Kapitel 6 werden weitere sinnvolle Suchmöglichkeiten beschrieben, die aber nicht im Rahmen dieser Arbeit realisiert wurden, da ansonsten der Zeitrahmen gesprengt worden wäre.

4.8 Design von EDEJU

Nachdem nun die Datenbank und die Ausgaben von EDEJU genauer betrachtet wurden, muss geklärt werden, wie die Daten im Browser dargestellt werden sollen. Dabei sind Darstellungsunterschiede zwischen den gängigen Internetbrowsern Internet Explorer, Opera, Netscape oder Mozilla in der Darstellung bei bestimmten HTML Tags zu beachten. Die meisten Internet Nutzer benutzen laut Statistiken den Internet Explorer¹⁸.

Die Internetseiten von EDEJU sind programmiert für eine Auflösung von 1024*768 Pixel, da diese Auflösung momentan die Gängigste ist.

Wichtige Kriterien, die eine Internetseite ausmachen, sind der Wiedererkennungseffekt, der durch ein einheitliches Layout gewährleistet werden soll. Zusätzlich ist eine sinnvolle und übersichtliche Navigation notwendig. Zusätzlich zu diesen Kriterien gibt es noch unzählige andere, die hier nicht weiter aufgeführt werden.

Alle Internetseiten von EDEJU werden innerhalb eines Framesets ausgegeben. Die Internetseiten sind aufgeteilt in ein Menü und die Ausgabe der jeweiligen Informationen. In Abbildung 4.14 ist ein Screenshot von EDEJU-Atlas dargestellt. Im Menübalken, der links auf der Seite angeordnet ist, hat der Benutzer die Möglichkeit, sich einzuloggen. Zusätzlich sind das EDEJU Logo und Links auf die verschiedenen Ausgaben der Daten vorhanden. Dazu gehören "Datenbank durchstöbern", "Bereichsmap", "Tips holen" (Vorschläge) und "Muster bzw. Vorbilder anschauen". Eine einfache Suche nach Bereichen, Lernmitteln, Projekten oder Personen stehen dem Benutzer dort auch zur Verfügung. Zusätzlich zu den Links, die sich auf die Inhalte der Datenbank von EDEJU beziehen, sind auch Links auf nähere Informationen zu EDEJU, wie "Mitmachen" und "Über uns" vorhanden. Um den rechtlichen Bestimmungen zu genügen ist hier gut sichtbar das Impressum angesiedelt. Eine sinnvolle Erweiterung, die aber im Rahmen dieser Arbeit nicht realisiert wurde, wäre

¹⁸Laut einer Statistik von <http://www.onestat.com/> im Januar 2004 benutzen etwa 94% aller Nutzer den Internetexplorer in verschiedenen Versionen gefolgt vom Mozilla mit 1,8% und Opera mit 0,8%

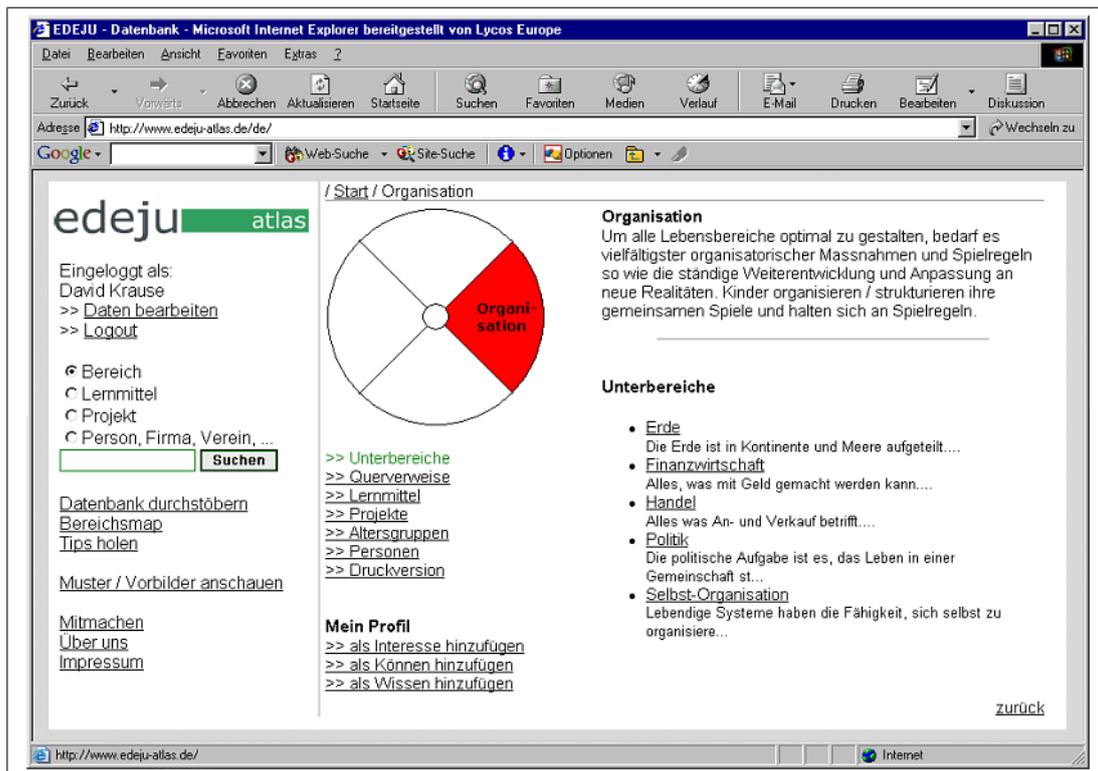


Abbildung 4.14: EDEJU-Atlas Screenshot: Bereiche

ein Hilfesystem bzw. eine "Erste Schritte Anleitung". Näheres dazu ist im Kapitel 6 zu finden.

Die Ausgabe der Informationen auf der rechten Seite der Internetseiten ist aufgeteilt in 3 Teile. In der Abbildung 4.14 ist diese Aufteilung dargestellt. In der ersten Zeile ist jeweils zu erkennen, wo sich der Benutzer befindet. Z.B. bei der Betrachtung von Lernmitteln steht dort "Lernmittel: Bezeichnung des Lernmittels". Bei der Betrachtung von Bereichen werden dort zusätzlich die Vorgänger des ausgewählten Bereichs ausgegeben. Diese sind mit Links versehen, damit der Benutzer direkt zu den Vorgängern springen kann. Der Untere Teil ist in einen linken und einen rechten Teil aufgeteilt. Im linken Teil sind neben dem Bild auch Informationen und falls notwendig ein Untermenü vorhanden. Bereiche haben dort ein Untermenü, in dem Links zu den Unterbereichen, Querverweisen, Lernmitteln, Projekten, Altersgruppen und Personen die mit diesem Bereich verknüpft sind. Zusätzlich ist dort ein Link zu einer Druckversion der relevanten Informationen zu finden. Falls der Benutzer eingeloggt ist, hat er direkt die Möglichkeit den Bereich oder das Lernmittel seinem Profil hinzuzufügen. Auf der Rechten Seite wird die Beschreibung, darunter je nach Wahl des Benutzers die Verknüpfungen mit weiteren Informationen ausgegeben.

Kapitel 5

Implementierung

Innerhalb dieses Kapitels wird die Implementierung von EDEJU erläutert. Im ersten Abschnitt wird der allgemeine strukturelle Aufbau von EDEJU geklärt. Im darauffolgenden Abschnitt wird auf die Reihenfolge der Implementierungsschritte eingegangen. Im Mittelpunkt steht dabei die Datenbank, die schon in der Entwurfsphase in Kapitel 4 entwickelt wurde. Abschließend werden anhand von einigen Implementierungsbeispielen Kernbereiche von EDEJU genauer erläutert. Darin werden auch Probleme, die aufgrund der Verwendung von MySQL als Datenbanksystem aufgetaucht sind, besprochen.

Diese Arbeit wurde auf einem Windows 98 System mit der Datenbank MySQL¹ 4.23.44 und dem Webserver Xitami² v.2.4d9 entwickelt und implementiert. Für die Implementierung wurde ein Java basierter Texteditor verwendet. Für die Administration der Datenbank wurde phpMyAdmin³ 2.2.6-rc2 verwendet.⁴ Eine CD mit dem gesamten Quellcode ist dieser Arbeit beigelegt. Im Internet ist EDEJU unter <http://www.edeju-atlas.de/de/> zu finden.

5.1 Struktureller Aufbau

5.1.1 Datenbank

Die Relationen, die im Anhang A dargestellt sind, bilden das Datenbankschema. Um Abfragen zu unterstützen wurden diese um Indexstrukturen erweitert. Zusätzlich zum Primärxindex, der den Primärschlüssel *ID* indiziert, wurden in verschiedenen Relationen Indexe hinzugefügt.

Innerhalb der Relation *edeju_bereiche* wird das Attribut *lft* und *rght* indiziert, dadurch soll die Performanz der Datenbankabfragen, welche die Struktur der Bereiche betreffen, gesteigert werden.

¹<http://www.mysql.com/>

²<http://www.imatix.com/>

³<http://www.phpmyadmin.net>

⁴Folgende Literatur wurde als Nachschlagewerk für Html, PHP, Javascript und MySQL verwendet: P.Dubois, J.Krause, S.Münz, MySQL Reference Manual, T.Theis, PHP-Handbuch, C.Wenz und Cascading Style Sheets.

Der Index über den Fremdschlüssel *rechteuserID* in der Relation *edeju_user_daten* beschleunigt Abfragen, die anhand der ID eines Benutzers die persönlichen Daten ermitteln wollen.

5.1.2 Ordnerstruktur

Die Ordnerstruktur bei einem solchen umfangreichen Projekt ist sehr wichtig, um nicht den Überblick zu verlieren. Es ist nicht sinnvoll alle Dateien in einem Verzeichnis unterzubringen.

Die Ordnerstruktur von EDEJU ist folgendermaßen aufgebaut: im Startverzeichnis liegen 4 Ordner *clips*, *css*, *de*, *php-lib*. Im Verzeichnis *clips* werden Bilder und Grafiken gespeichert, die in fast allen HTML-Seiten gebraucht werden. Im Verzeichnis *css* sind die benötigten Stylesheet Dateien enthalten, die z.B. die Schriftart oder Hintergrundfarben von Tabellen zentral verwalten. Dadurch ist es möglich, später die Schriftart zu ändern, ohne dabei alle Dateien verändern zu müssen. Das Verzeichnis *php-lib* enthält 2 Unterverzeichnisse. Zum einen ein Verzeichnis *cfg* in dem Konfigurationsdateien, die die Zugangsdaten für die Datenbank und gängige Fehlermeldungen enthalten gespeichert, zum anderen gibt es noch das Verzeichnis *dump* in dem eine Datenbanksicherung abgespeichert wird.

Im Verzeichnis *de* ist die Verwaltung und der öffentliche Bereich von EDEJU enthalten. Es ist aufgeteilt in 9 Verzeichnisse. 5 davon enthalten die Bilder, die zu den Bereichen, Lernmittel, Projekten, Personen oder Muster-Kompetenzträgern gehören. Das *inc* und das *js* Verzeichnis, enthalten PHP- bzw. Javascript Funktionen, die immer wieder in anderen Dateien benötigt werden und dort eingebunden werden. Mit Hilfe der beiden Verzeichnisse *mitglieder* und *edeju* wird das System logisch in den Mitglieder-Bereich und den öffentlichen Bereich unterteilt.

Der öffentlich Bereich im Verzeichnis *edeju* enthält die Verzeichnisse *bereiche*, *druckversion*, *impressum*, *komplett*, *lernmittel*, *lostoppassword*, *mitmachen*, *muster*, *new*, *personen*, *projekte*, *suche*, *vergleiche* und *vorschlag*. Innerhalb dieser Verzeichnisse liegen die Dateien, die die jeweiligen Informationen ausgegeben.

Das Verzeichnis *mitglieder* enthält die Verwaltung von EDEJU. Die Verzeichnisse *password* und *selfdelete* enthalten die Dateien, mit denen jeder Benutzer sein Passwort ändern kann bzw. seinen Account löschen kann. Das Verzeichnis *kontaktsystem* enthält die Dateien, mit denen die Kontaktaufnahme zwischen den Benutzern ermöglicht wird.

Die restlichen Verzeichnisse enthalten die Verwaltung der EDEJU Daten *system*, die Benutzerverwaltung *useradmin* und die Verwaltung der eigenen Daten bzw. fremder Daten *user*.

Im Verzeichnis *System* werden in den Unterverzeichnissen *altersgruppe*, *bereiche*, *lernmittel*, *muster* und *musterart* die Daten, die unabhängig von den Benutzern in der Datenbank enthalten sind, eingetragen bzw. bearbeitet. Dazu gehören die Altersgruppeneinteilung, die Bereiche, die Lernmittel und die Muster-Kompetenzträger sowie die dazugehörige Einteilung der Muster-Kompetenzträger.

Das Verzeichnis *useradmin* enthält die Dateien, die zur Einteilung der Benutzer in Gruppen benötigt werden. Zusätzlich sind noch die Verzeichnisse *gruppe*, *newsletter*, *transfer* und *user* enthalten. Dort können Gruppen bearbeitet bzw. erstellt, User-Administratoren Gruppen zugeordnet, Newsletter verschickt, Benutzer in andere Gruppen transferiert und Benutzer gelöscht werden.

Das Verzeichnis *user* enthält schließlich alle Dateien, die benötigt werden, um die Daten eines Benutzers zu bearbeiten. Innerhalb der Verzeichnisse *adresse*, *angebote*, *beschreibung*, *bild*, *changemail*, *firma*, *freigabe*, *interessen*, *koennen*, *newsletter*, *projekte* und *wissen* können alle benötigten Daten bearbeitet bzw. eingetragen werden.

Im Anhang C ist eine Auflistung der Dateien und die Ordnerstruktur von EDEJU mit einer kurzen Beschreibung der jeweiligen Dateien beigefügt.

5.1.3 Dateiaufbau

Die zugrundeliegende Struktur der Dateien sind HTML Dateien. PHP wird direkt in den HTML-Code eingebunden. Durch die Endung der Datei z.B. *.php* erkennt der Server, dass diese Dateien noch von einem PHP-Prozessor interpretiert werden müssen. Dabei wird alles, was zwischen

```
<?php
?>
```

steht, vom PHP-Prozessor ausgeführt. Diese PHP-Bausteine können überall im HTML-Code eingebunden werden.

Der typische Aufbau einer Datei ist im Folgenden dargestellt.

Am Beginn jeder Datei steht ein Kommentar, da PHP-Kommentare nicht an den Browser weitergegeben werden, eignet sich dieser dazu, den serverseitigen PHP-Code zu erklären, ohne dass der spätere Nutzer Zugriff auf diese Kommentare hat. Die Datei wird darin benannt und dann beschrieben. Innerhalb der Beschreibung werden u.a. mögliche Übergabeparameter, die benötigt werden, genauer beschrieben. Ein überall verwendeter Übergabeparameter ist z.B. die Session-ID *\$sid*. Diese wird benötigt, um den Benutzer zu identifizieren.

```
1 <?php
2 #####
3 # kurze Dateibeschreibung
4 #####
5 # Beschreibung:
6 # ...
7 # Übergabeparameter:
8 # - $sid: Session ID der Benutzers
9 #####

11 # Include File mit den globalen Datenbankdaten
12 include (" ../../ php_lib/cfg/standard.cfg.php");

14 # Include File mit den Fehlermeldungen
15 include (" ../../ php_lib/cfg/errors.cfg.php");

17 # Include File mit den Rechte Überprüfungen
18 include (" ../../ php_lib/cfg/rechte.cfg.php");
```

Nach diesem Kommentar werden die benötigten Funktionen und Informationen eingebunden, die in jeder Datei, die auf die Datenbank zugreift, benötigt werden. Dazu gehört die

Datei *standard.cfg.php*, die in einem assoziativen Array die benötigten Informationen für die Verbindung zur jeweiligen Datenbank enthält. So ist es möglich, wenn die Datenbank umbenannt wird oder das Passwort geändert wird, zentral den Namen bzw. das Passwort zu ändern. Die zweite Datei *errors.cfg.php* enthält gängige Fehlermeldungen in einem assoziativen Array. Im Mitglieder-Bereich wird eine dritte Datei *rechte.cfg.php* mit Funktionen eingebunden, die überprüfen, ob der Benutzer ein bestimmtes Recht hat oder das Recht hat, eine andere Person zu bearbeiten.

Darauf folgend werden weitere benötigte Dateien eingebunden. An die Variablen *\$errortext* und *\$erfolgtext* werden Fehlermeldungen bzw. Erfolgsmeldungen, die auf der HTML Seite ausgegeben werden sollen, angehängt. Falls Datenbankabfragen erfolgen, wird nun eine persistente Datenbankverbindung geöffnet. Persistent bedeutet, dass diese Verbindung nicht sofort nach Beendigung geschlossen wird, sondern, falls erneut eine Verbindung aufgebaut werden soll, überprüft wird, ob schon eine identische Verbindung vorhanden ist, um diese dann zu benutzen. Allerdings funktioniert diese Art der Verbindung nur, wenn der Server PHP als Modul verwendet und diese Art der Verbindung auch unterstützt. Falls der Server dies nicht tut, wird die Verbindung zur Datenbank bei jeder Seite immer wieder neu aufgebaut. Wenn persistente Datenbankverbindungen möglich sind, wird das ständige Öffnen und Schließen von Datenbankverbindungen verhindert.

Sobald die Verbindung erfolgreich geöffnet wurde, wird die benötigte Datenbank ausgewählt, falls die Datenbankverbindung fehlschlägt, wird eine Fehlermeldung ausgegeben.

```

19 #####
20 # Falls Fehler auftreten werden Fehlermeldungen hier eingetragen
21 $errortext = "";
22 $erfolgtext = "";
23 #####
24 # Datenbank öffnen
25 $db = @mysql_pconnect (
26     $cfg["db_server"],
27     $cfg["db_username"],
28     $cfg["db_password"]
29 );

31 if ($db) {
32     @mysql_select_db ($cfg["db_name"]);
33 } else {
34     $errortext = $errors["database_connect_failure"];
35 }
36 #####

```

Im Mitglieder-Bereich, wird nun anhand der Session-ID überprüft, ob der Benutzer eingeloggt ist und wenn ja, ob der Timeout abgelaufen ist. Dies geschieht in der Datei *sicherheit.inc.php* in der gleichzeitig der Timeout wieder auf das Maximum gesetzt wird. Falls diese Überprüfung fehlschlägt, sei es wegen einer falschen Session-ID oder aufgrund des abgelaufenen Timeout wird der Benutzer auf die Startseite weitergeleitet, mit der Begründung, dass sein Timeout abgelaufen ist.

```

36 #####
37 # Include File zu Überprüfung des Logins
38 include ("../inc/sicherheit.inc.php");
39 #####
40 ?>

```

Im öffentlichen Bereich werden die für diese Seite benötigten Dateien eingebunden, wie die Funktion zum Erzeugen einer Session-ID in der Datei *session.inc.php*, die auf jeder Seite benötigt wird. Danach werden gegebenenfalls Übergabeparameter mit Standardwerten belegt, die noch nicht belegt sind. Falls die Seite mit dem Übergabeparameter "login = 1" aufgerufen wurde, versucht der Benutzer anhand der übergebenen Variablen *password* und *username* sich einzuloggen. Es ist daher notwendig, die Datei *publiclogin.inc.php* einzubinden. Dort erhält der Benutzer eine neue gültige Session-ID. Falls *\$sid* bereits übergeben oder erstellt wurde wird diese auf Gültigkeit überprüft.

Im öffentlichen und im Mitglieder-Bereich werden nun gegebenenfalls Veränderungen in der Datenbank vorgenommen. Durch Parameter wie "step = 2" wird angezeigt, dass diese Seite zur Verarbeitung von Anweisungen aufgerufen wurde.

```

41 #####
42 # Session ID erzeugen
43     include (" ../../ inc/session.inc.php");
44 #####
45 # als Benutzer einloggen
46 if($login == "1"){
47     #####
48     # Include File mit dem Login im oeffentlichen Bereich
49     # wird ausgeführt wenn login == 1 ist.
50     # benötigt werden die Felder username und password
51     #####
52     include (" ../../ inc/publiclogin.inc.php");
53 }
54 #####
55 # Include File zur Überprüfung des Logins
56 # wird nur überprüft, falls eine SessionID übergeben wurde
57 if (isset($sid)){
58     include (" ../../ inc/sicherheit.inc.php");
59 }
60 #####
61 if($step == 2){
62     # Anfragebearbeitung
63 }
64 #####
66 ?>

```

Danach wird die HTML Seite ausgegeben. Zuerst der "Head" mit dem Titel, den Meta-Daten und den benötigten Javascript Funktionen. Dann der "Body" mit dem Inhalt der Seite und gegebenenfalls den Fehlermeldungen oder Erfolgsmeldungen. Zu beachten ist dabei, dass Sonderzeichen im HTML-Code, der durch PHP erzeugt wird, wie das Anführungszeichen " mit einem \ maskiert werden müssen, damit durch die Verwendung von HTML in PHP keine Fehler auftauchen. In Zeile 103 oder 106 des Quelltextes wird dies deutlich.

```

67 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
68     "http://www.w3.org/TR/REC-html40/loose.dtd">
69
70 <html lang="de"
71     dir="ltr">
72
73 <link rel="stylesheet" type="text/css" href="../../css/standard.css">
74
75 <head>
76
77 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
78 <meta name="author" content="David Krause">

```

```

79 <meta name="description" content="EDEJU-Atlas">
80 <meta name="keywords" content="">
81 <meta name="allow-search" content="yes">
82 <meta name="robots" content="index, follow">
83 <meta name="audience" content="all">
84 <meta name="content-language" content="de">
85
86 <title>EDEJU - Atlas</title>
87 </head>
88
89
90 <body
91     topmargin= "0"
92     leftmargin= "0"
93     marginwidth="0"
94     marginheight= "0"
95 >
96 <?php
97 Ausgabe der Internetseite
98 ?>
99 <?php
100 # Fehlerausgabe
101 if ( $errortext != "" ) {
102     echo "<span class=\"red\">". $errortext. "</span>";
103 }
104 if ( $erfolgtext != "" ) {
105     echo "<span class=\"red\">". $erfolgtext. "</span>";
106 }
107
108 </body>
109 </html>

```

Für Datenbankabfragen wird zuerst die SQL-Anfrage definiert, dann wird diese Anfrage an die Datenbank geschickt. Diese schickt das Ergebnis, welches mehrere Tupel enthalten kann, zurück. Das Ergebnis wird Tupel für Tupel in einer Schleife in ein assoziatives Array geparkt und gegebenenfalls ausgegeben. Falls kein Tupel gefunden wurde, wird eine Fehlermeldung oder ein Hinweis ausgegeben.

```

1 $query = "SELECT bezeichnung";
2 $query .= " FROM edeju_bereiche";
3 $query .= " ORDER BY bezeichnung";
4
5 $res = @mysql_query ( $query , $db );
6 $num = @mysql_num_rows ( $res );
7
8 if ( $num != 0 ) {
9     for ( $i=0; $i<$num; $i++) {
10         $pre = @mysql_fetch_assoc ( $res );
11         echo $pre[" bezeichnung "];
12     }
13 } else {
14     # Keine Tupel gefunden , Fehlerbehandlung
15 }

```

5.1.4 Implementierungsschritte

Die Implementierung von EDEJU ist in verschiedene Phasen eingeteilt. Zuerst müssen Funktionen zur Verfügung gestellt werden, die die Session Verfolgung unterstützen. Dies ist notwendig, da alle Dateien bei EDEJU die Session Verfolgung unterstützen sollen. In einem zweiten Schritt ist es notwendig, eine Benutzerverwaltung aufzubauen, welche

die geforderten Funktionen enthält. Dazu gehören die Verteilung von Rechten und die Verwaltung von Benutzern in Gruppen. Nachdem diese beiden Bausteine implementiert sind, kann die Datenbank und die dazugehörigen Masken zur Eingabe von Daten bzw. der gesamte Mitglieder-Bereich erstellt werden. Als letzter Schritt wird der öffentliche Bereich implementiert, der die Informationen, die in der Datenbank gespeichert sind, in geeigneter Weise präsentiert.

5.2 Implementierungsbeispiele

In diesem Abschnitt wird die Implementierung von EDEJU anhand einiger ausgewählter Beispiele genauer erläutert. Dazu gehört die Implementierung des Upload von Dateien bzw. Bildern, der Vorschläge und der Vergleiche. Dabei werden die Beispiele nicht komplett durchgesprochen, sondern nur die Kernpunkte der jeweiligen Implementierung.

Auf die Darstellung der Implementierung der Bereichsstruktur mit Hilfe des Nested Sets Modells, die den Kern von EDEJU bildet, wird hier verzichtet. Diese wurden schon im Abschnitt 4.5.2 vorgestellt und erläutert.

5.2.1 Bilder Upload

Bilder sind bei EDEJU sehr wichtig, da "Bilder mehr sagen als tausend Worte". Daher ist es notwendig, Bilder bei Personen, Bereichen, Lernmitteln, Projekten oder Muster-Kompetenzträgern hinzufügen zu können. Ein Problem ist, dass große Bilder das Layout von EDEJU zerstören können und zu viel Speicherplatz und Transfervolumen einnehmen. Aufgrund dieser Problematik dürfen nur Bilder hinzugefügt werden, die kleiner als 200*200 Pixel sind. Im Beispielcode wird zunächst die Größe und Art des Bildes ermittelt. Dann wird überprüft, ob die Breite und Höhe des Bildes kleiner als 200 Pixel sind. Zusätzlich wird überprüft, ob das Bild im JPEG oder im GIF Format vorliegt. Gegebenenfalls wird eine Fehlermeldung ausgegeben.

```
1 $size = getimagesize($bild);
2 if($size[0] >200 || $size[1] > 200 || ($size["2"] != "1" && $size["2"] != "2")){
3     # Fehlermeldung ausgeben
4 }
```

Wenn das Bild der Überprüfung standgehalten hat, wird das Bild in das richtige Verzeichnis kopiert. Dabei wird das Bild umbenannt, der neue Name im Beispielcode ist die ID des Benutzers in Verbindung mit "_user". Die Variable "bildart" enthält die Endung der Datei in diesem Fall JPEG oder GIF. Auf diese Art werden doppelte Namen verhindert und das Bild ist immer einem Benutzer direkt zugeordnet. Wenn das Bild erfolgreich kopiert wurde, wird die Datenbank aktualisiert und der Name der Datei eingetragen.

```
1 if(copy($bild, "../photos/".$thisuser["ID"]."_user".$bildart)){
2     $query = "UPDATE edeju_user_daten SET bild = \"xy_user".$bildart.\"";
3     $query .= " WHERE rechteuserid = \"xy\"";
4     @mysql_query($query, $db);
5 }else{
6     # Fehlerbehandlung
7 }
```

Natürlich können auch andere Dateien hochgeladen werden. In vielen Fällen liegen z.B. Dokumente wie PDF-Files vor, die nähere Informationen zu einem Projekt, Lernmittel, Bereich oder einer Person enthalten. Diese werden dann auf die gleiche Art und Weise in das Angebot geladen.

5.2.2 Vorschläge

In diesem Abschnitt wird die Implementierung der "Ähnlichen Vorschläge" und der "Weiße Flecken" genauer erläutert.

Während der Berechnung der Vorschläge ist es notwendig, Ergebnismengen einzelner SQL-Queries zu vereinigen. Da bei MySQL 3.23 die UNION Anweisung nicht vorhanden ist und dadurch Ergebnismengen nicht direkt vereinigt werden können, ist es notwendig eine temporäre Tabelle zu erstellen, um diese Ergebnismengen zwischenspeichern zu können. MySQL bietet dazu echte temporäre Tabellen an, die nach Verwendung auch wieder automatisch gelöscht werden. Allerdings können aus diesen Tabellen keine Tupel mehr entfernt werden. Es ist daher notwendig, normale Tabellen zur Berechnung von "Ähnlichen Vorschlägen" und "Weißen Flecken" zu verwenden. Dabei ist zu beachten, dass diese nach der Verwendung wieder gelöscht werden müssen.

Ähnliche Vorschläge

Das Ziel ist es, aus der Menge der Vorgänger, deren Kindern, den Querverweisen und deren Kindern und den Kindern der Bereiche, die im Profil gespeichert sind, geeignete Bereiche zu finden.

Es müssen mehrere Datenbankabfragen gemacht und die Ergebnisse vereinigt werden, um die Menge der in Frage kommenden Bereiche zu erstellen. Ausserdem ist sicherzustellen, dass keine Bereiche enthalten sind, die schon im Profil vorkommen.

In einem ersten Schritt muss die temporäre Tabelle erstellt werden. Damit diese eindeutig ist, wird die Tabelle "temp_UserID" genannt, da die *User ID* eindeutig ist. Diese enthält die Attribute *ID*, *left*, *right*, *level* und *bezeichnung* der Bereiche. Zusätzlich wird noch das Attribut *bewertung* hinzugefügt, welches die jeweilige Bewertung des Bereichs enthält.

Im Quellcode werden in der äußeren For-Schleife die verschiedenen Profiltypen (*Interesse*, *Wissen*, *Können*) durchlaufen. In der inneren For-Schleife werden für jeden Typ die Bereiche, Querverweise und Väter ermittelt. Je nach Art des Typs wird der Wert der Bereiche festgelegt und mit den Kindern in die temporäre Tabelle eingetragen.

```
1 $profiltypen = array ("interessen", "wissen", "koennen");
2 # Profiltypen durchlaufen
3 for ($k=0; $k<3;$k++){
4     # Verwandtschaftsgrade durchlaufen
5     for ($j=0; $j<3; $j++) {
6         # Für jeden Typ werden Querverweise, die Bereiche und Väter ermittelt.
7         $query = "SELECT b.ID, b.left, b.right, b.level";
8         if ($j == 1){
9             # Querverweise aus der Datenbank holen
10            $query .= " FROM edeju_user_bereich_". $profiltypen[$k]. " i, edeju_bereiche b,
                edeju_bereiche_querverweise q";
11            $query .= " WHERE i.userID = ".$userid." AND i.bereichID = b.ID AND (q.
                bereich1ID = b.ID OR q.bereich2ID = b.ID)";
12        } elseif ($j == 0){
13            # Bereiche aus der Datenbank holen
14            $query .= " FROM edeju_user_bereich_". $profiltypen[$k]. " i, edeju_bereiche b";
15            $query .= " WHERE (i.userID = ".$userid." AND i.bereichID = b.ID)";
```

```

16 }elseif($j == 2){
17     # Väter aus der Datenbank holen
18     $query .= " FROM edeju_user_bereich_". $profiltypen[$k]. " i, edeju_bereiche b,
19         edeju_bereiche eb";
20     $query .= " WHERE i.userID = ".$userid." AND i.bereichID = eb.ID AND eb.lft
21         BETWEEN b.lft AND b.rght AND b.level = eb.level - 1 ";
22 }
23 $res = @mysql_query ($query, $db);
24 $num = @mysql_num_rows ($res);
25
26 # Jeder Bereich und seine Kinder werden in die temporäre Tabelle eingetragen
27 # Bewertung
28 # Interessensverwandte bekommen den Wert 3
29 if($k == 0){ $bewert = 3; }
30 # Könnenverwandte bekommen den Wert 2
31 if($k == 2){ $bewert = 2; }
32 # Wissenverwandte den Wert 1
33 if($k == 1){ $bewert = 1; }
34 for ($i=0; $i<$num; $i++) {
35     $pre = @mysql_fetch_assoc ($res);
36     $query = "INSERT INTO temp-".$userid." (bewertung, ID, lft, rght, level,
37         bezeichnung)";
38     $query .= " SELECT ".$bewert.", ID, lft, rght, level, bezeichnung";
39     $query .= " FROM edeju_bereiche";
40     $query .= " WHERE lft BETWEEN ".$pre["lft"]." AND ".$pre["rght"]." AND level
41         <= ".$pre["level"]."+1";
42     @mysql_query ($query, $db);
43 }

```

Im nächsten Schritt müssen dann nur noch die im Profil vorhandenen Bereiche aus dieser Tabelle entfernt werden. Schließlich können die Bereiche sortiert nach der Bewertung ausgegeben werden. Mit Hilfe des nachfolgenden Queries ist dies möglich.

```

1 SELECT DISTINCT *, SUM(bewertung) AS bewertung
2 FROM temp_userID
3 GROUP BY ID
4 ORDER BY bewertung DESC

```

Weißer Flecken

Um weiße Flecken im Profil zu finden, wird zunächst eine Tabelle erstellt, in der alle vorhandenen Bereiche enthalten sind. Aus dieser Menge werden nun alle Bereiche entfernt, die im Profil enthalten sind bzw. verwandt mit diesen Bereichen sind. Anschließend werden 5 Bereiche ausgewählt und ausgegeben. Am Schluss muss die erstellte Tabelle wieder entfernt werden.

5.2.3 Vergleiche

Jeder Benutzer kann sich sowohl mit anderen Benutzern, deren Profil freigegeben ist, wie auch mit Muster-Kompetenzträgern vergleichen. Um den Vergleich nicht zu unübersichtlich werden zu lassen, kann der Benutzer auswählen, ob er sein *Interesse*, sein *Wissen* oder sein *Können* mit dem Profil der anderen Person vergleichen will. Dabei wird anhand des eigenen Profils überprüft, ob der andere Benutzer Gemeinsamkeiten oder Ähnlichkeiten besitzt. Zuerst wird geprüft, ob die andere Person Gemeinsamkeiten besitzt. Als Beispiel hat der Benutzer *Interesse* im Bereich Modellbau, die andere Person hat zu dieser Thematik sowohl *Interesse*, *Können* und *Wissen*. Diese Übereinstimmung wird dann ausgegeben. Falls keine Gemeinsamkeiten bestehen, wird überprüft, ob Ähnlichkeiten bestehen. Dazu

wird überprüft, ob die Vergleichsperson Bereiche im Profil eingetragen hat, die ein Kind, ein Querverweis oder ein Vorgänger des jeweiligen Bereichs sind. Falls dies der Fall ist, werden diese Bereiche ausgegeben. Natürlich kann es auch den Fall geben, dass keine Übereinstimmung gefunden wird, dann wird nichts ausgegeben bzw. ein entsprechender Hinweis ausgegeben.

Im Quellcode wird zunächst das *Interesse*, *Können* oder *Wissen* des Benutzers aus der Datenbank geholt und dann Schritt für Schritt in der ersten For-Schleife durchlaufen. Dann wird eine temporäre Tabelle erstellt, in die Gemeinsamkeiten bzw. Ähnlichkeiten eingetragen werden. Schließlich wird anhand des aktuellen Bereichs, aus dem Profil des Benutzers, überprüft, ob die Vergleichsperson Gemeinsamkeiten beim *Interesse*, *Wissen* oder *Können* hat. Falls ja, wird dies entsprechend ausgegeben. Falls es bei einem Profiltyp keinerlei Übereinstimmung gibt, wird überprüft, ob es Ähnlichkeiten bei den Kindern, Vorgängern oder Querverweisen gibt. Am Ende werden, falls vorhanden, die Ähnlichkeiten ausgegeben.

```

1 $profilarray = array(array("interessen","Interesse"), array("koennen","Können"), array("wissen","Wissen
  "));
2 # Interesse, Wissen oder Können des Benutzers aus der Datenbank holen.
3 $query = "SELECT b.*";
4 $query .= " FROM edeju_user_bereich_". $art. " u, edeju_bereiche b";
5 $query .= " WHERE userID = ". $userid. " AND b.ID = u.bereichID";
6
7 $res = @mysql_query ($query, $db);
8 $num = @mysql_num_rows ($res);
9
10 if ($num != 0) {
11     for ($i=0; $i<$num; $i++) {
12         $uebereinstimmung = 0;
13         $pre = @mysql_fetch_assoc ($res);
14         # Erstellen der temporären Tabelle
15         $query = "CREATE TEMPORARY TABLE temp_". $userid. "(ID INT(11) UNSIGNED, bezeichnung
          VARCHAR(255), art INT(2))";
16         @mysql_query ($query, $db);
17
18         # Überprüfen, ob es Übereinstimmungen mit dem Muster oder der Person gibt
19         # suchen in Interessen, Können und Wissen.
20         for ($z=0; $z<3; $z++) {
21             $query = "SELECT ID";
22             $query .= " FROM edeju_". $type. "_bereich_". $profilarray[$z][0];
23             $query .= " WHERE bereichID = ". $pre["ID"]. " AND ". $type. "ID = ". $id;
24
25             $res2 = @mysql_query ($query, $db);
26             $num2 = @mysql_num_rows ($res2);
27             if ($num2 > 0) {
28                 # Es gibt eine Gemeinsamkeit: diese wird ausgegeben
29                 $pre2 = @mysql_fetch_assoc ($res2);
30                 # Ausgabe von Interesse, Können oder Wissen
31                 $uebereinstimmung = 1;
32             } else {
33                 # Keine Gemeinsamkeiten, Ähnlichkeiten bei Kindern, Vorgängern und
34                 # Querverweisen suchen
35
36                 # Kinder
37                 $query = "INSERT INTO temp_". $userid. " (ID, bezeichnung)";
38                 $query .= "SELECT b.ID, b.bezeichnung";
39                 $query .= " FROM edeju_bereiche b, edeju_". $type. "_bereich_".
40                 $profilarray[$z][0]. " m";
41                 $query .= " WHERE m.". $type. "ID = ". $id. " AND m.bereichID = b.ID";
42                 $query .= " AND b.lft BETWEEN ". $pre["lft"]. " AND ". $pre["right"]. " AND b
43                 .level <= ". $pre["level"]. " + 1";
44
45                 @mysql_query ($query, $db);
46
47                 # Vorgänger
48                 $query = "INSERT INTO temp_". $userid. " (ID, bezeichnung)";
49                 $query .= "SELECT b.ID, b.bezeichnung";
50                 $query .= " FROM edeju_bereiche b, edeju_". $type. "_bereich_".
51                 $profilarray[$z][0]. " m";
52                 $query .= " WHERE m.". $type. "ID = ". $id. " AND m.bereichID = b.ID";
53                 $query .= " AND ". $pre["lft"]. " BETWEEN b.lft AND b.right AND b.level
54                 = ". $pre["level"]. " - 1";

```

```

50
51         @mysql-query ($query , $db);
52
53         # Querverweise
54         $query = "INSERT INTO temp_."$userid." (ID, bezeichnung)";
55         $query .= "SELECT b.ID, b.bezeichnung";
56         $query .= " FROM edeju_bereiche_querverweise q, edeju_bereiche b, edeju_
57         ". $type."_bereich_". $profilarray[$z][0]." m";
58         $query .= " WHERE m."$type."ID = ".$id." AND m.bereichID = b.ID";
59         $query .= " AND b.ID = q.bereich1ID AND ((q.bereich2ID = ".$pre["ID"].")
60         OR (b.ID = q.bereich2ID AND q.bereich1ID = ".$pre["ID"]."))";
61     }
62 }
63 # Ausgeben der Ähnlichkeiten , falls in der temporären Tabelle vorhanden
64
65 # Löschen der temporären Tabelle
66 $query = "DROP TABLE temp_."$userid;
67 @mysql-query ($query , $db);
68 }
69 }

```


Kapitel 6

Erweiterungen

Der Hauptaugenmerk dieser Arbeit lag in der Entwicklung einer Datenbank und der Implementierung der Grundfunktionen von EDEJU. Es gibt eine Vielzahl von möglichen Erweiterungen, die nicht im Rahmen dieser Diplomarbeit implementiert werden konnten. Im Grunde wird ein System wie EDEJU nie fertig sein, da neue Bedürfnisse immer Änderungen bzw. Erweiterungen erfordern werden. Im Folgenden werden mögliche schon jetzt ersichtliche Erweiterungen genauer untersucht und mögliche Ansätze zur Implementierung erläutert.

6.1 Datenbank Upgrade

Momentan ist EDEJU für MySQL 3.23 ausgelegt. Da aber in Zukunft MySQL 4.0 und höher Standard werden wird, sollten Änderungen bei der Programmierung vorgenommen werden.

In Kapitel 4 tauchte das Problem auf, dass in der aktuellen Datenbankversion noch keine Transaktionen vorhanden sind. Für die Struktur der Bereiche ist es sehr wichtig, dass die Einfüge-, Entferne- und Update-Operationen als ein Block ausgeführt werden, da ansonsten die Struktur zerstört werden kann. Momentan ist nur gewährleistet, dass nur ein Benutzer die Nested Sets Struktur verändern kann. Die Frage, ob alle Queries korrekt ausgeführt wurden, kann zwar beantwortet werden, aber bei Problemen und Fehlern nicht rückgängig gemacht werden. Bei der Umstellung auf eine Datenbankversion mit Transaktionen sollten diese Operationen als erstes sicher gemacht werden. Damit dies möglich wird, müssen die jeweiligen Relationen vom Typ *InnoDB* sein, dieser Typ gewährleistet die ACID Eigenschaften. Die benötigten Queries müssen wie folgt eingebunden werden.

```
START TRANSACTION;  
  
... Queries  
  
COMMIT;
```

Falls ein Fehler während der Transaktion auftritt, kann anstatt von *COMMIT* ein *ROLLBACK* die Änderungen wieder rückgängig machen.

Um die Datenbank sicherer zu machen, ist es zudem notwendig, sobald eine neue MySQL

Version Foreign Keys unterstützt, diese nachträglich einzufügen. Auf diese Weise können die momentan mit einzelnen Queries vorgenommenen Löschungen oder Updates direkt von der Datenbank gemacht werden.

Um die Vergleiche und Vorschläge performanter machen zu können, ist bei einem Umstieg des Datenbanksystems die Berechnung zu verändern. So können Ergebnismengen mit UNION zusammengefasst werden und auf die Erstellung von zusätzlichen Tabellen verzichtet werden.

6.2 Abfragekombinationen

Aufgrund der Vielzahl von Relationen und Verbindungen zwischen den Daten gibt es viele Möglichkeiten, Daten abzufragen. Die Suche nach einem Projekt kann z.B. mit einem oder mehreren Lernmitteln bzw. Bereichen verbunden werden, so dass nur Projekte ausgegeben werden, die mit diesen verknüpft sind. Es ist notwendig eine erweiterte Suche zu implementieren, die eine Vielzahl von Kombinationsmöglichkeiten ermöglicht.

Ein möglicher Lösungsansatz wäre, diese Suche in mehreren Schritten zu starten. So kann zuerst ausgesucht werden, nach "was" (Person, Projekt, ...) gesucht werden soll. In einem zweiten Schritt kann der Benutzer auswählen, mit was die Suche verknüpft sein soll, z.B. mit Lernmitteln. In einem dritten Schritt kann der Benutzer die Lernmittel aussuchen die in der Kombination auftreten sollen. Dieses Verknüpfen von Daten Schritt für Schritt kann beliebig erweitert werden. Dabei ist aber darauf zu achten, dass die Anzahl der Schritte, aus Gründen der Benutzerfreundlichkeit möglichst klein bzw. vom Benutzer beeinflusst werden kann.

Zusätzlich zu der Suche können natürlich weitere Standardkombinationen, wie z.B. alle Lernmittel, die einer bestimmten Altersgruppe zugeordnet sind, unabhängig von der Suche einen eigenen Menüpunkt erhalten. So kann ein Logfile, welches die Suchkombinationen festhält, Auskunft geben, welche Kombinationen bevorzugt werden. Die Kombinationen, die am häufigsten auftreten, können dann direkt gesucht werden und einen eigenen Punkt erhalten.

6.3 Darstellungsformen

Es gibt viele Möglichkeiten, wie z.B. die Struktur der Bereiche dargestellt werden kann. Eine Darstellungsform ist die Baumstruktur, wie sie in EDEJU unter dem Punkt "Bereichsmap" zu finden ist. Eine weitere Darstellungsform ist eine Kreisstruktur, in der jeweils ein Viertel des Kreises den 4 Bereichen (Natur, Technik, Organisation, Kultur) entspricht und der sich von innen nach aussen immer weiter in die Unterbereiche auffächert. Eine andere Möglichkeit der Darstellung ist eine Art Planetarium¹. Auf diese Weise könnten sowohl die Bereiche und die jeweiligen Unterbereiche, wie auch Querverweise angezeigt werden. Zusätzlich könnten zu jedem Bereich noch die Lernmittel und Projekte direkt angezeigt werden.

¹Diese Art der Darstellung wird auf der Internetseite <http://www.planet-wissen.de/> unter dem Stichwort Wissens-Planetarium verwendet.

Es ist denkbar, mehrere Layouts für Projekte, Lernmittel oder Bereiche anzubieten und dabei mehr als ein Bild zu verwenden. Der Benutzer könnte dann z.B. bei der Eingabe von Projekten zwischen verschiedenen Layouts wählen und auch beliebig viele Bilder dazu einstellen. Dafür ist es notwendig, zusätzliche Relationen, in der die Bilder zu Projekten, Bereichen, Personen oder Lernmitteln gespeichert werden, zu erstellen.

6.4 Fremdsprachen

Die Ausweitung von EDEJU auf andere Sprachen ist eine weitere Option. Dabei treten viele Probleme auf, die gelöst werden müssen. Probleme treten bei der Eingabe von allen Daten auf. Bei der Eingabe von Bereichen muss die Bezeichnung und die Beschreibung in alle Sprachen, die bei EDEJU implementiert sind, übersetzt und eingetragen werden. Dies tritt natürlich auch bei Personenbeschreibungen, Projekten, Lernmitteln, usw. auf. Nachdem diese Übersetzungsfragen gelöst sind, muss geklärt werden, wie und ob die Datenbankstruktur geändert werden muss oder ob nicht für jede Sprache eine separate Datenbank verwendet wird. Bei der Verwendung von mehreren Datenbanken tritt das Problem von Redundanzen auf, was bei der Größe von EDEJU unüberschaubar wäre. Aus diesem Grund ist es sinnvoll, innerhalb einer Datenbank für alle Felder, die übersetzt werden müssen für jede Sprache ein Feld hinzuzufügen. Um z.B. EDEJU um Englisch zu erweitern wäre es in der Relation Lernmittel notwendig, die Felder *bezeichnungEN* und *beschreibungEN* hinzuzufügen, die die Übersetzung der Felder *bezeichnung* bzw. *beschreibung* enthalten. Innerhalb der Ausgabe der Daten muss dann gewährleistet sein, dass die Sprache ausgewählt werden kann und die richtigen Felder ausgegeben werden.

Bei der Ausgabe gibt es die Möglichkeit, dass innerhalb der bestehenden PHP/HTML - Dateien zwischen den Sprachen ausgewählt wird und die entsprechenden Ausgaben erfolgen. So kann z.B. immer die Variable "sprache" mit der jeweiligen Sprache übergeben werden. Die andere Möglichkeit ist eine Kopie der Dateien, die in der jeweiligen Sprache die Daten ausgibt. Bei Änderungen des Codes oder Erweiterungen müssten dann allerdings immer alle Dateien, der jeweiligen Sprache angepasst werden.

Zur Vereinfachung bei der Entwicklung sind die Dateien für die deutsche Sprache schon im Ordner "de" abgelegt. Bei der Erweiterung um die Sprache Englisch müssten diese alle in den Ordner "en" kopiert werden. Zusätzlich müssen die Erläuterungen und Ausgaben ersetzt werden und die richtigen Felder der Datenbank ausgegeben werden. Die Auswahl der Sprache könnte entweder auf der Startseite erfolgen oder aber auf jeder anderen Seite. Es ist ersichtlich, dass diese Erweiterung um Fremdsprachen einen enormen Aufwand mit sich bringt. Es ist aber sicher sinnvoll, EDEJU später, wenn die deutsche Version mit genügend Daten gefüllt ist, auch ausserhalb des deutschsprachigen Raumes verwenden zu können.

6.5 Level

Um die Entwicklung des eigenen Profils genauer verfolgen zu können ist die Erweiterung, um einen Level des *Wissen*, *Könnens* oder des *Interesse* interessant. Ein Kind, das z.B. *Wissen* über ein Auto hat, d.h. es weiß, dass das Auto fährt, einen Motor hat und Benzin braucht. Ein Automechaniker weiß aber, wie ein Auto aufgebaut ist und funktioniert. Dieser Unterschied, der im *Wissen*, *Können* aber auch im *Interesse* von Menschen vorhanden ist, kann durch eine Klassifizierung genauer erfasst werden. Eine mögliche Klassifizierung ist z.B. die Einteilung in Einsteiger, Fortgeschrittene und Profis. Es kann auch eine andere Art der Klassifizierung verwendet werden z.B. eine Skala von 1 bis 10. Der Benutzer könnte auswählen, wenn er etwas seinem *Wissen*, *Interessen* oder *Können* hinzufügt hat, auf welchem Level er selbst steht. Später, wenn der Benutzer etwas dazugelernt hat und sich als Profi fühlt, könnte er seine Entwicklungsstufe in einem Bereich oder Lernmittel heraufsetzen. Diese Art der Klassifizierung setzt voraus, dass die Benutzer sorgfältig überlegen, welchem Level sie selbst angehören.

Eine andere Art der Klassifizierung wäre, die nach dem jeweiligen Alter des Benutzers. Allerdings ist diese Art der Einteilung fraglich, da z.B. 10jährige im Bereich Computer mehr *Wissen* bzw. *Können* haben, als 40jährige, die zwar mit dem Computer arbeiten, aber keine Ahnung haben, wie er funktioniert bzw. aufgebaut ist. Daher wäre es sinnvoller, dass sich jeder Benutzer selbst klassifizieren kann.

6.6 Hilfesystem bzw. Erste Schritte

Eine weitere sinnvolle Erweiterung wäre ein Hilfesystem, welches auf jeder Seite die möglichen Funktionen näher beschreibt. Es wäre sinnvoll, dieses Hilfesystem zentral aufzubauen und die Textbausteine in einer Datenbank zu speichern. Auf diese Weise könnten Funktionen, die auf mehreren Seiten auftauchen, gleich eingebunden werden und müssen nicht erst neu erstellt oder kopiert werden.

Zusätzlich sollte noch eine "Frequently Asked Questions" Liste aufgebaut werden, damit häufige Fragen sofort und ohne mit einem anderen Benutzer Kontakt aufnehmen zu müssen, gelöst werden können. Solch eine Liste kann erst nach einer Vorlaufzeit erstellt werden, zu erst müssen die relevanten Fragen gesammelt werden.

Eine "Erste Schritt" Anleitung könnte neuen Benutzers helfen sich schnell in EDEJU zu rechtzufinden und alle Optionen zu nutzen.

6.7 Sonstige Erweiterungen

Natürlich gibt es noch viele weitere Erweiterungsmöglichkeiten. Bereits im Rahmen dieser Diplomarbeit sind sehr viele Ideen dazugekommen.

Um mit anderen Benutzern näher in Kontakt zu kommen, könnte EDEJU um einen Chat oder ein Messageboard erweitert werden. Durch einen Chat könnten Erfahrungen oder

Fragen "live" während des Surfens mit anderen Benutzern diskutiert werden. Messageboards könnten z.B. bei Lernmitteln, Projekten oder ganz allgemein zu EDEJU eingesetzt werden. Auf diese Weise können Erfahrungen, Fragen oder Kommentare zu allen Themen hinterlassen werden und dadurch EDEJU bereichern. Natürlich tritt dann das Problem auf, dass diese Chats bzw. Messageboards überwacht werden müssten, damit kein "Müll" oder rechtlich problematische Dinge enthalten sind.

Es ist ersichtlich, dass die Liste der möglichen Erweiterungen sehr lang wird. Die hier aufgeführten Optionen sollen einen Denkanstoß geben, EDEJU weiter auszubauen und zu nutzen.

Kapitel 7

Zusammenfassung

Im Rahmen dieser Arbeit wurde die Grundstruktur des EDEJU-Atlas entwickelt und implementiert. Innerhalb von 3 Abschnitten, der Anforderungsanalyse, der Definitionsphase und der Entwurfsphase wurde ein Modell entwickelt, welches in der Implementierungsphase umgesetzt wurde. Abgerundet wurde diese Arbeit durch einen Ausblick auf mögliche Erweiterungen und Verbesserungen von EDEJU.

In der Anforderungsanalyse wurde aufbauend auf den Vorstellungen von Herrn Helmeth die Struktur und die Funktionen von EDEJU genauer erläutert. Ausgehend von wenigen Grundfunktionen wurde nach und nach ein großes Modell.

Innerhalb der Analyse war es zudem wichtig zuerst die Begriffe genau zu definieren, damit eine gemeinsame begriffliche Grundlage vorhanden war. Innerhalb der Anforderungsanalyse wurde nach möglichen vorhandenen Daten für die Datenbank gesucht, um diese zu verwenden. Allerdings stellte sich recht schnell heraus, dass Daten in dieser Form nicht vorhanden sind und erst zusammengestellt werden müssen.

In der Definitionsphase wurde die Struktur und die Funktionen konkretisiert. Der Kernpunkt innerhalb der Definitionsphase ist das Entity-Relationship Modell der Datenbank. Dieses enthält alle Relationen und Beziehungen, die für EDEJU benötigt werden. Zusätzlich wurden die Geschäftsprozesse genauer dargelegt und erläutert.

In der Entwurfsphase wurden die Grundsatzentscheidungen über zu verwendende Datenbank und Programmiersprache getroffen. Der wichtigste Punkt der Entwurfsphase war die Entscheidung, welche Struktur für die Bereiche genutzt wird. Das Nested Sets Modell wurde gewählt, da es durch seinen flexiblen Aufbau und die einfachen Abfragen den anderen Modellen Pfad-Modell und Parent-Modell überlegen ist. Der einzige große Nachteil ist momentan das fehleranfällige Updaten der Struktur. Allerdings wird durch den Umstieg auf MySQL 4.0 oder höher dieses Problem gelöst. Auch Vergleiche und Vorschläge werden vom Umstieg auf eine höhere Version profitieren.

Bei der Implementierung wurde der allgemeine Dateiaufbau und verschiedene andere Beispiele genauer betrachtet. Dabei wurden die Lösungen der aufgetretenen Probleme, wie das Fehlen der UNION Abfrage genauer erläutert.

Im Kapitel Erweiterungen wurden verschiedene mögliche Optionen von EDEJU genauer betrachtet. Der nächste Schritt, der gemacht werden sollte ist der Umstieg auf MySQL 4.0. Allerdings ist dies abhängig vom Serverprovider, der erst die Datenbank aktualisieren muss.

Die Entwicklung des EDEJU-Atlas wurde, wie wahrscheinlich jedes Projekt einer solchen Dimension, begleitet von vielen Problemen und Schwierigkeiten. Zunächst war es schwierig ohne Daten eine geeignete Datenbankstruktur aufzubauen, da erst nach und nach klar wurde, welche Daten benötigt werden. Zusätzlich mussten die Beziehungen zwischen den verschiedenen Daten geklärt werden, die meist in konkreten Fragen, wie "ist es möglich alle Projekte einer bestimmten Region auszugeben?" erst auftauchten. Während der Gespräche wurden viele Ansätze für die Datenstruktur aufgebaut und dann aufgrund von neu auftauchenden Fragen und Daten wieder verworfen. Die Datenbankstruktur hat sich, nachdem diese festgelegt war als flexibel genug gezeigt, um auf die gewünschten Veränderungen zu reagieren.

Sehr wichtig ist es für EDEJU, dass am Anfang genügend Bereiche und Lernmittel eingetragen sind, ansonsten wird ein neuer Benutzer, der nur wenige Informationen vorfindet EDEJU unter Umständen nicht wieder besuchen. Daher ist der erste Schritt der nun getan werden muss, das Beschaffen von Daten bzgl. der Bereiche und Lernmittel und eine geeignete Strukturierung derselben.

EDEJU kann, sobald genügend informative Daten eingetragen wurden, zu einem sinnvollen Instrument bei der Entwicklung des Menschen werden. Durch das Verknüpfen von Lebensbereichen und Lernmitteln ist es möglich, ohne große Anstrengungen über den Tellerrand seiner Interessen hinauszuschauen und auf neue, bisher unbekannte Dinge zu stoßen. Der Benutzer wird, dadurch dass EDEJU ständig verändert und erweitert wird, animiert immer wieder in EDEJU reinzuschauen, um neue Ideen zu suchen und zu finden. Die Möglichkeit Projekte einzutragen oder Personen mit ähnlichen Interessen zu finden, trägt dazu bei, dass EDEJU eine Plattform zum lebendigen und interkulturellen Wissensaustausch werden könnte.

Anhang A

Datenbank - Schema

edeju_altersgruppe	
<u>ID</u>	INT(11)
von	INT(3)
bis	INT(3)
beschreibung	varchar(40)

edeju_bereiche	
<u>ID</u>	INT(11)
bezeichnung	varchar(255)
beschreibung	text
bild	varchar(100)
datum	date
lft	INT(11)
rght	INT(11)
level	INT(11)
lft	INDEX
rght	INDEX

edeju_bereiche_altersgruppe		
<u>ID</u>	INT(11)	
IDbereiche	INT(11)	
IDaltersgruppe	INT(11)	
IDbereiche -> edeju_bereiche		
IDaltersgruppe -> edeju_altersgruppe		
(IDbereiche, IDaltersgruppe)		UNIQUE

edeju_bereiche_ansprechpartner	
<u>ID</u>	INT(11)
bereicheID	INT(11)
userID	INT(11)
owner	ENUM(0,1)
bereicheID -> edeju_bereiche	
userID -> rechte_rights	

edeju_bereiche_querverweise	
<u>ID</u>	INT(11)
bereich1ID	INT(11)
bereich2ID	INT(11)
beschreibung	text
bereich1ID, bereich2ID -> edeju_bereiche	

edeju_lernmittel	
<u>ID</u>	INT(11)
bezeichnung	varchar(200)
beschreibung	text
link	varchar(150)
bild	varchar(100)
datum	date

edeju_lernmittel_altersgruppe		
<u>ID</u>	INT(11)	
IDlernmittel	INT(11)	
IDaltersgruppe	INT(11)	
IDlernmittel -> edeju_lernmittel		
IDaltersgruppe -> edeju_altersgruppe		
(IDlernmittel, IDaltersgruppe)		UNIQUE

edeju_lernmittel_ansprechpartner	
<u>ID</u>	INT(11)
lernmittelID	INT(11)
userID	INT(11)
owner	ENUM(0,1)
lernmittelID -> edeju_lernmittel	
userID -> rechte_rights	

edeju_lernmittel_bereiche	
<u>ID</u>	INT(11)
lernmittelID	INT(11)
bereichID	INT(11)
lernmittelID -> edeju_lernmittel	
bereichID -> edeju_bereiche	

edeju_muster	
<u>ID</u>	INT(11)
musterID	INT(11)
bezeichnung	varchar(100)
beschreibung	text
link	varchar(150)
bild	varchar(100)
datum	date
musterID -> edeju_muster_art	

edeju_muster_ansprechpartner	
<u>ID</u>	INT(11)
musterID	INT(11)
userID	INT(11)
owner	ENUM(0,1)
musterID -> edeju_muster	
userID -> rechte_rights	

edeju_muster_art	
<u>ID</u>	INT(11)
bezeichnung	varchar(100)
beschreibung	text

edeju_muster_bereich_interessen	
<u>ID</u>	INT(11)
musterID	INT(11)
bereichID	INT(11)
beschreibung	text
bild	varchar(100)
musterID -> edeju_muster	
bereichID -> edeju_bereiche	
(musterID, bereichID) UNIQUE	

edeju_muster_bereich_koennen	
<u>ID</u>	INT(11)
musterID	INT(11)
bereichID	INT(11)
beschreibung	text
bild	varchar(100)
musterID -> edeju_muster	
bereichID -> edeju_bereiche	
(musterID, bereichID) UNIQUE	

edeju_muster_bereich_wissen	
<u>ID</u>	INT(11)
musterID	INT(11)
bereichID	INT(11)
beschreibung	text
bild	varchar(100)
musterID -> edeju_muster	
bereichID -> edeju_bereiche	
(musterID, bereichID)	UNIQUE

edeju_muster_lernmittel_interessen	
<u>ID</u>	INT(11)
musterID	INT(11)
lernmittelID	INT(11)
beschreibung	text
bild	varchar(100)
musterID -> edeju_muster	
lernmittelID -> edeju_lernmittel	
(musterID, lernmittelID)	UNIQUE

edeju_muster_lernmittel_koennen	
<u>ID</u>	INT(11)
musterID	INT(11)
lernmittelID	INT(11)
beschreibung	text
bild	varchar(100)
musterID -> edeju_muster	
lernmittelID -> edeju_lernmittel	
(musterID, lernmittelID)	UNIQUE

edeju_muster_lernmittel_wissen	
<u>ID</u>	INT(11)
musterID	INT(11)
lernmittelID	INT(11)
beschreibung	text
bild	varchar(100)
musterID -> edeju_muster	
lernmittelID -> edeju_lernmittel	
(musterID, lernmittelID)	UNIQUE

edeju_projekte	
<u>ID</u>	INT(11)
bezeichnung	varchar(200)
beschreibung	text
termin_von	date
termin_bis	date
ort	varchar(100)
land	varchar(100)
region	varchar(100)
link	varchar(150)
bild	varchar(100)
oeffentlich	ENUM(0,1)
mitmachen	ENUMm(0,1)
datum	date

edeju_projekte_bereiche	
<u>ID</u>	INT(11)
projektID	INT(10)
bereichID	INT(10)
projektID -> edeju_projekte	
bereichID -> edeju_bereiche	

edeju_projekte_lernmittel	
<u>ID</u>	INT(11)
IDprojekte	INT(11)
IDlernmittel	INT(11)
IDprojekte -> edeju_projekte	
IDlernmittel -> edeju_lernmittel	

edeju_projekte_user	
<u>ID</u>	INT(11)
personID	INT(11)
projektID	INT(11)
owner	ENUM(0,1)
anfrage	ENUM(0,1)
personID -> rechte_rights	
projektID -> edeju_projekte	

edeju_user_bereich_interessen	
<u>ID</u>	INT(11)
userID	INT(11)
bereichID	INT(11)
beschreibung	text
bild	varchar(100)
eingefuegt	date
geupdatet	date
userID -> rechte_rights	
bereichID -> edeju_bereiche	
(userID, bereichID)	UNIQUE

edeju_user_bereich_koennen	
<u>ID</u>	INT(11)
userID	INT(11)
bereichID	INT(11)
beschreibung	text
bild	varchar(100)
eingefuegt	date
geupdatet	date
userID -> rechte_rights	
bereichID -> edeju_bereiche	
(userID, bereichID)	UNIQUE

edeju_user_bereich_wissen	
<u>ID</u>	INT(11)
userID	INT(11)
bereichID	INT(11)
beschreibung	text
bild	varchar(100)
eingefuegt	date
geupdatet	date
userID -> rechte_rights	
bereichID -> edeju_bereiche	
(userID, bereichID)	UNIQUE

edeju_user_daten	
<u>ID</u>	INT(11)
rechteuserID	int(11)
vorname	varchar(150)
nachname	varchar(150)
strasse	varchar(150)
plz	varchar(20)
ort	varchar(100)
land	varchar(100)
nationalitaet	varchar(100)
telefon	varchar(100)
fax	varchar(100)
homepage	varchar(150)
geburtstag	date
beruf	varchar(255)
beschreibung	text
bild	varchar(100)
firma	ENUM(0,1)
firmentname	varchar(255)
freigabe_adresse	ENUM(0,1)
freigabe_profil	ENUM(0,1)
rechteuserID -> rechte_rights	
rechteuserID	INDEX

edeju_user_lernmittel_angebote	
<u>ID</u>	INT(11)
userID	INT(11)
lernmittelID	INT(11)
beschreibung	text
bild	varchar(100)
eingefuegt	date
geupdatet	date
userID -> rechte_rights	
lernmittelID -> edeju_lernmittel	
(userID, lernmittelID)	UNIQUE

edeju_user_lernmittel_interessen	
<u>ID</u>	INT(11)
userID	INT(11)
lernmittelID	INT(11)
beschreibung	text
bild	varchar(100)
eingefuegt	date
geupdatet	date
userID -> rechte_rights	
lernmittelID -> edeju_lernmittel	
(userID, lernmittelID)	UNIQUE

edeju_user_lernmittel_koennen	
<u>ID</u>	INT(11)
userID	INT(11)
lernmittelID	INT(11)
beschreibung	text
bild	varchar(100)
eingefuegt	date
geupdatet	date
userID -> rechte_rights	
lernmittelID -> edeju_lernmittel	
(userID, lernmittelID)	UNIQUE

edeju_user_lernmittel_wissen	
<u>ID</u>	INT(11)
userID	INT(11)
lernmittelID	INT(11)
beschreibung	text
bild	varchar(100)
eingefuegt	date
geupdatet	date
userID -> rechte_rights	
lernmittelID -> edeju_lernmittel	
(userID, lernmittelID)	UNIQUE

edeju_gruppe	
<u>ID</u>	INT(11)
gruppenname	varchar(100)
beschreibung	varchar(255)
gruppenname	UNIQUE

edeju_gruppe_admin	
<u>ID</u>	INT(11)
admin	INT(11)
gruppe	INT(11)
admin -> rechte_rights	
gruppe -> edeju_gruppe	

rechte_rights	
<u>ID</u>	INT(11)
rig	varchar(255)
name	varchar(255)
beschreibung	varchar(255)
sortierung	INT(11)
rig	UNIQUE

rechte_user	
<u>ID</u>	INT(11)
gruppe	INT(11)
oldgruppe	INT(11)
transfer	ENUM(0,1,2)
name	varchar(255)
passwort	varchar(255)
email	varchar(150)
sessionID	varchar(30)
last_update	varchar(255)
datum	date
newsletter	ENUM(0,1)
rig_passwort	ENUM(0,1)
rig_user	ENUM(0,1)
rig_useradmin	ENUM(0,1)
rig_selfdelete	ENUM(0,1)
rig_giverights	ENUM(0,1)
rig_admin	ENUM(0,1)
rig_admin_change	ENUM(0,1)
rig_userdelete	ENUM(0,1)
rig_lernmittel	ENUM(0,1)
rig_projekte	ENUM(0,1)
rig_altersgruppe	ENUM(0,1)
rig_bereiche	ENUM(0,1)
rig_musterkompetenz	ENUM(0,1)
rig_eigene_lernmittel	ENUM(0,1)
rig_newsletter	ENUM(0,1)
name	UNIQUE

Anhang B

Geschäftsprozesse

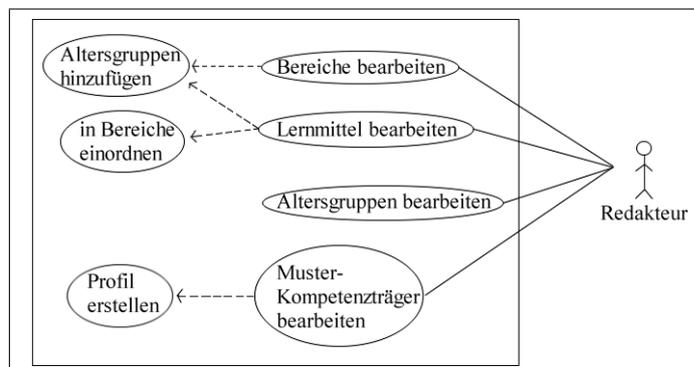


Abbildung B.1: Geschäftsprozessdiagramm: Redakteur

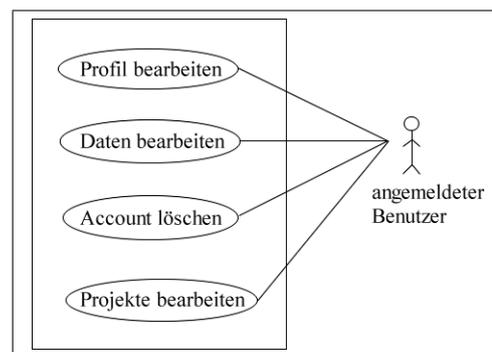


Abbildung B.2: Geschäftsprozessdiagramm: Mitgliederbereich

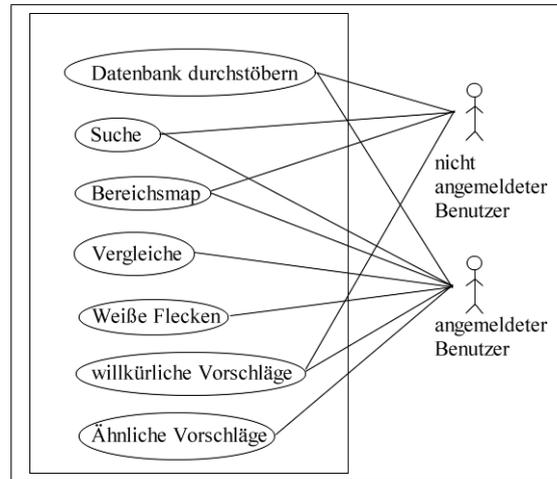


Abbildung B.3: Geschäftsprozessdiagramm: Öffentlicher Bereich

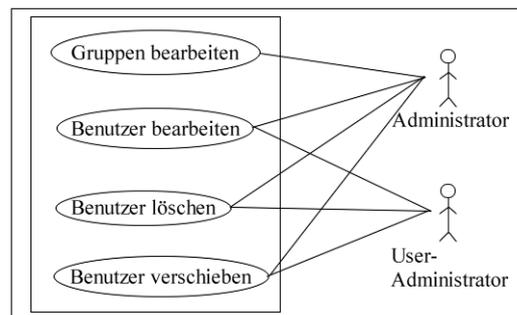


Abbildung B.4: Geschäftsprozessdiagramm: Benutzerverwaltung

Anhang C

Ordnerstruktur & Dateien

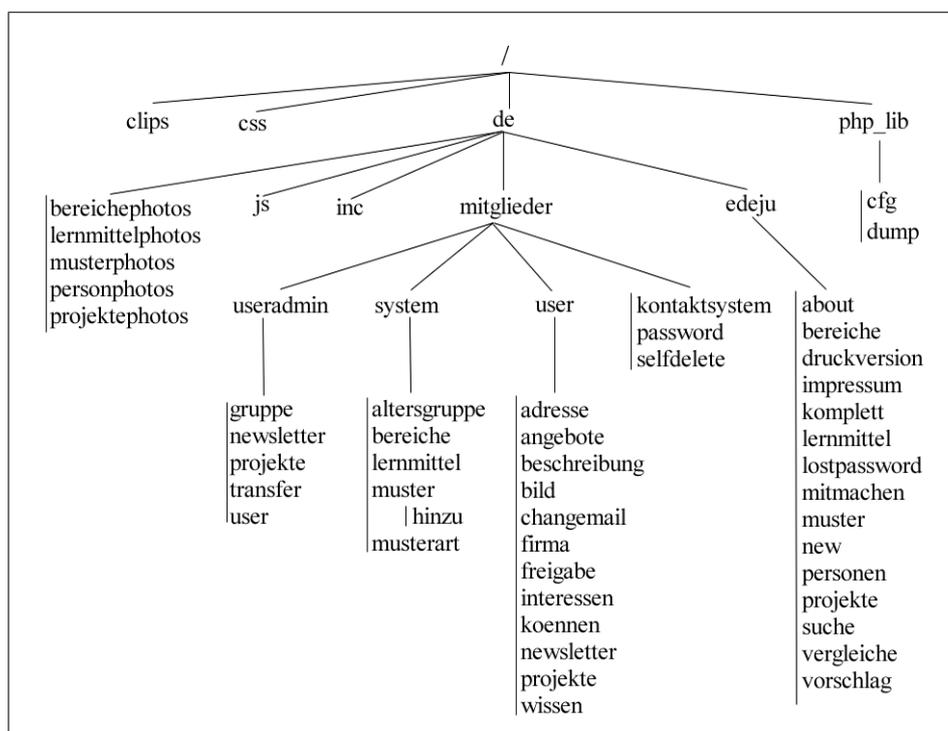


Abbildung C.1: Ordnerstruktur

<i>/clips (häufig benötigte Bilder und Grafiken)</i>
<i>/css (Stylesheets)</i> <i>standard.css: zentrale Stylesheet Datei</i>
<i>/de (Mitglieder-, Verwaltungs-, öffentlicher Bereich)</i> <i>index.php, nix.htm: Definition Frameset</i>
<i>/de/bereichphotos (Bereiche Bilder)</i> <i>/de/lernmittelphotos (Lernmittel Bilder)</i> <i>/de/musterphotos (Muster Bilder)</i> <i>/de/personphotos (Personen Bilder)</i> <i>/de/projektephotos (Projekte Bilder)</i> <i>/de/edeju (Öffentlicher Bereich)</i>

<p><i>/de/inc (PHP-Includes)</i> <i>datum.inc.php: Funktionen zur Datum Formatierung</i> <i>insertansprechpartner.inc.php: allgemeine Funktion zum Ansprechpartner einfügen</i> <i>isinterest.inc.php: Funktion zum Überprüfen, ob ein Bereich, Lernmittel im Profil vorhanden</i> <i>nestedsetsfunctions.inc.php: Nested Sets Funktionen zum Einfügen, Entfernen, Verschieben von Bereichen</i> <i>publiclogin.inc.php: einloggen</i> <i>publicmenu.inc.php: Menü im öffentlichen Bereich</i> <i>rechte.inc.php: Funktionen zur Überprüfung von Rechten</i> <i>selfdelete.inc.php: Benutzer löschen</i> <i>session.inc.php: Session erzeugen</i> <i>sicherheit.inc.php: Session überprüfen</i> <i>textbearbeiten.inc.php: Text Überarbeitung</i> <i>userfunctions.inc.php: Funktionen die Daten über den Benutzer ermitteln</i></p>
<p><i>/de/edeju (Öffentlicher Bereich)</i> <i>/de/edeju/about/index.php (Über uns)</i> <i>/de/edeju/bereiche/index.php (Datenbank durchstöbern anhand der Bereiche)</i> <i>/de/edeju/druckversion/index.php (Druckversion)</i> <i>/de/edeju/impressum/index.php (Impressum)</i> <i>/de/edeju/komplett/index.php (Bereichsmap)</i> <i>/de/edeju/lernmittel/index.php (Lernmittel ausgeben)</i> <i>/de/edeju/lostpassword/index.php (Neues Passwort anfordern)</i> <i>/de/edeju/mitmachen/index.php (Mitmachen)</i> <i>/de/edeju/muster/index.php (Ausgabe von Musterkompetenzträgern)</i> <i>/de/edeju/new/index.php, step2.php (Als neuer Benutzer anmelden)</i> <i>/de/edeju/personen/index.php (Personen ausgeben)</i> <i>/de/edeju/projekte/index.php (Projekte ausgeben)</i> <i>/de/edeju/suche/index.php (Suchen)</i> <i>/de/edeju/vergleiche/index.php (Vergleiche mit Musterkompetenzträgern oder Personen)</i> <i>/de/edeju/vorschlag/index.php (Willkürliche, Ähnliche, Weiße Flecken Vorschläge)</i></p>
<p><i>/de/js (Javascript-Includes)</i> <i>mailfunction.js, printfunction.js: Javascript Funktionen für Kontaktsystem und Druckversion</i></p>
<p><i>/de/mitglieder (Verwaltungs- und Mitgliederbereich)</i> <i>logout.php, uebersicht.php: ausloggen und Übersichtsseite des Mitglieder- bzw. Verwaltungsbereich</i> <i>/de/mitglieder/kontaktsystem/index.php (EDEJU - Kontaktsystem)</i> <i>/de/mitglieder/password/index.php (Passwort ändern)</i> <i>/de/mitglieder/selfdelete/index.php (eigenen Account löschen)</i></p>
<p><i>/de/mitglieder/system (Datenverwaltung)</i> <i>/de/mitglieder/system/altersgruppe (Altersgruppen bearbeiten)</i> <i>index.php, alter.php, delete.php, new.php: Übersicht, bearbeiten, erstellen, löschen</i> <i>/de/mitglieder/system/bereiche (Bereiche bearbeiten)</i> <i>index.php, alter.php, delete.php, new.php: Übersicht, bearbeiten, erstellen, löschen</i> <i>querverweise.php, verschiebe.php, verweis.php: Querverweise hinzufügen, Bereich verschieben, Querverweise bearbeiten</i> <i>/de/mitglieder/system/lernmittel (Lernmittel bearbeiten)</i> <i>index.php, alter.php, delete.php, new.php: Übersicht, bearbeiten, erstellen, löschen</i> <i>newbereich.php Lernmittel einem Bereich zuordnen</i> <i>/de/mitglieder/system/muster (Muster Kompetenzträger bearbeiten)</i> <i>index.php, alter.php, delete.php, new.php: Übersicht, bearbeiten, erstellen, löschen</i> <i>alter2.php: vorhandenes Profil bearbeiten</i> <i>/de/mitglieder/system/muster/hinzu/index.php (Muster Profil erstellen)</i> <i>/de/mitglieder/system/musterart (Musterarten bearbeiten)</i> <i>index.php, alter.php, new.php: Übersicht, bearbeiten, erstellen</i></p>
<p><i>/de/mitglieder/user (Benutzerdaten)</i> <i>index.php, beschreibung.php: Übersicht, Rechte Erläuterungen</i> <i>/de/mitglieder/user/adresse/index.php (Benutzerdaten bearbeiten)</i> <i>/de/mitglieder/user/angebote/index.php, alter.php (Lernmittelangebote, bearbeiten)</i> <i>/de/mitglieder/user/beschreibung/index.php (Beschreibung bearbeiten)</i> <i>/de/mitglieder/user/bild/index.php (Bild ändern)</i> <i>/de/mitglieder/user/changemail/index.php (Email Adresse ändern)</i> <i>/de/mitglieder/user/firma/index.php, ansprech.php (Firmenname, Ansprechpartner bearbeiten)</i> <i>/de/mitglieder/user/freigabe/index.php (Daten freigeben)</i> <i>/de/mitglieder/user/interessen/index.php, alter.php (Interessen, bearbeiten)</i> <i>/de/mitglieder/user/koennen/index.php, alter.php (Können, bearbeiten)</i> <i>/de/mitglieder/user/newsletter/index.php (Newsletter abonnieren)</i> <i>/de/mitglieder/user/projekte (Projekte)</i> <i>index.php, alter.php, delete.php, new.php: Übersicht, bearbeiten, erstellen, löschen</i></p>

<p><i>/de/mitglieder/user/wissen/index.php, alter.php (Wissen, bearbeiten)</i></p>
<p><i>/de/mitglieder/useradmin (Benutzerverwaltung und administrative Aufgaben)</i> <i>index.php, show.php: Gruppenübersicht, Gruppe näher betrachten</i> <i>/de/mitglieder/useradmin/gruppe (Gruppenverwaltung)</i> <i>index.php, alter.php, delete.php, new.php: Übersicht, bearbeiten, erstellen, löschen</i> <i>admindelete.php, adminnew.php: User-Administrator einer Gruppe löschen oder neu hinzufügen</i> <i>/de/mitglieder/useradmin/newsletter/index.php (Newsletter verschicken)</i> <i>/de/mitglieder/useradmin/projekte/index.php (Übersicht Projekte)</i> <i>/de/mitglieder/useradmin/transfer/index.php (Benutzer in andere Gruppe verschieben)</i> <i>/de/mitglieder/useradmin/user/delete.php (Benutzer löschen)</i></p>
<p><i>/php_lib</i> <i>/php_lib/cfg</i> <i>errors.cfg, standard.cfg: zentrale Fehlermeldungen, Datenbankzugangsdaten</i> <i>/php_lib/dump (Datenbanksicherung)</i></p>

Anhang D

Rechte

- **rig_password**: Der Benutzer kann sein eigenes Passwort zu ändern.
- **rig_user**: Der Benutzer kann seine Daten bearbeiten.
- **rig_useradmin**: Der Benutzer ist ein User-Administrator.
- **rig_selfdelete**: Der Benutzer kann seinen Account löschen.
- **rig_admin**: Der Benutzer ist ein Administrator.
- **rig_giverights**: Der Benutzer kann Rechte vergeben.
- **rig_admin_change**: Der Benutzer kann andere Administratoren bearbeiten.
- **rig_userdelete**: Der Benutzer kann andere Benutzer löschen.
- **rig_lernmittel**: Der Benutzer kann Lernmittel eintragen, bearbeiten und löschen.
- **rig_projekte**: Der Benutzer kann Projekte eintragen, bearbeiten und löschen.
- **rig_altersgruppe**: Der Benutzer kann Altersgruppen bearbeiten.
- **rig_bereiche**: Der Benutzer kann Bereiche eintragen, bearbeiten und löschen.
- **rig_musterkompetenz**: Der Benutzer kann Musterkompetenzträger eintragen, bearbeiten und löschen.
- **rig_eigene_lernmittel**: Der Benutzer kann Lernmittel erstellen, diese bearbeiten und löschen.
- **rig_newsletter**: Der Benutzer kann Newsletter verschicken.

Literaturverzeichnis

- [1] H.Balzert *Lehrbuch der Software - Technik* 2.Auflage 2001 Spektrum Akademischer Verlag
- [2] J.Celko *Joe Celko's SQL For Smarties* 2.Auflage 2000 Academic Press
- [3] P.Dubois *MySQL entwicklung, implementierung und referenz* 1.Auflage 2000 Markt+Technik Verlag
- [4] A.Kemper/A.Eikler *Datenbanksysteme Eine Einführung* 3.Auflage 1999 R.Oldenbourg Verlag
- [5] J.Krause *PHP 4 Grundlagen und Profiwissen* 1.Auflage 2000 Carl Hanser Verlag
- [6] G.Matthiessen/M.Unterstein *Relationale Datenbanken und SQL* 2.Auflage 2000 Addison-Wesley Verlag
- [7] S.Münz *Selhtml* Internet: <http://selfaktuell.teamone.de/> Version 8.0
- [8] T.Theis *PHP4* 1. Auflage 2000 Galileo Press GmbH
- [9] *MySQL Reference Manual* Internet: <http://www.mysql.com/documentation/>
- [10] T.Ottmann/P.Widmayer *Algorithmen und Datenstrukturen* 3.Auflage 1996 Spektrum Akademischer Verlag
- [11] *PHP Handbuch* Internet: <http://www.php.net/manual/de/>
- [12] PHP Magazin Artikel von Stephan Schmidt *Weltenbaum* Ausgabe 02.2002 Software&Support Verlag GmbH
- [13] PHP Magazin Artikel von Arne Klempert *Wurmloch* Ausgabe 03.2003 Software&Support Verlag GmbH
- [14] R.Ramakrishnan/J.Gehrke *Database Management Systems* 2.Auflage 2000 McGraw-Hill
- [15] A.S.Tanenbaum *Computernetzwerke* 3.revidierte Auflage 2000 Pearson Studium

- [16] C.Wenz *Javascript Das umfassende Handbuch* 5.Auflage 2003 Galileo Press GmbH
- [17] *Cascading Style Sheets, level 1* W3C Recommendation 17 Dec 1996, revised 11 Jan 1999 Internet: <http://www.w3.org/TR/CSS1>