Fachbereich 4, Informatik - Institut für Management

Eine multimediale Lernumgebung für Datenbanken - Konzeption und Realisierung -

Bachelorarbeit

Zur Erlangung des Grades eines Bachelor of Science im Studiengang Informationsmanagement

vorgelegt von Daniel Reiser

Betreuer: Prof. Dr. Carlo Simon, Fachbereich 4, Institut für Management

Erstgutachter: Prof. Dr. Carlo Simon, Fachbereich 4, Institut für Management

Zweitgutachter: Prof. Dr. Klaus G. Troitzsch, Fachbereich 4, Institut für Wirtschafts- und Verwaltungsinformatik

Koblenz, 14. September 2007

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Koblenz, 14. September 2007

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Kapitel 1

Einleitung

1.1 E-Learning und Blended Learning

Mit dem Begriff *E-Learning* werden alle Formen computer- und internetbasierter Lehr- und Lernangebote bezeichnet.
Als Teilmengen des E-Learning können sowohl das *computer based training* als auch das *web based training* angesehen werden: Computer based training (CBT) umfasst dabei diejenigen Lehr- und Lernangebote, die unter Verwendung von Computern angeboten werden. Web based training (WBT) fasst schließlich solche Lehr- und Lernangebote zusammen, bei denen das Internet als Bildungsmedium verwendet wird (Kerres, 2001, Seite 14).
Multimediale Lehrangebote, in denen mit Hilfe von technischen Übertragungsmedien wie dem Internet Informationen in Form von Texten, Bildern oder Videos über weite Distanzen hinweg vermittelt werden können, bieten vielfältige Einsatzmöglichkeiten von Computern zur Aus- und Weiterbildung. Das Konzept des *Blended Learning*, eine Kombination der klassischen Präsenzlehre mit elektronischen Lehrformen, stellt nach Ostheimer und Freisler (2006) in diesem Zusammenhang eine große Herausforderung dar. Das Ziel dabei ist es, eine Balance zwischen klassischen und modernen, multimedialen Lehrangeboten herzustellen um so die Vorteile beider Formen optimal nutzen zu können.
Speziell im Bereich der unterstützenden Lehrangebote gibt es bereits eine Vielzahl an sogenannten webbasierten Lernplattformen, deren Verbreitung in den letzten Jahren rasant zugenommen hat (Schulmeister, 2001, vgl.:). Portale und Kursmanagementsystem wie *BSCW*, *eCollege.com* oder *WebCT* sind sehr gute Beispiele hierfür und finden vermehrt in der universitären Ausbildung eine weite Verbreitung.

1.2 Ausgangssituation und Zielsetzung

Der Bedarf an ergänzenden Lernangeboten zum Inhalt der Vorlesung *Datenbanken für Informationsmanager* am Fachbereich 4, Informatik der Universität Koblenz-Landau bildete die Ausgangssi-

tuation zur Entwicklung der webbasierten Lernsoftware *Lernmodul Normalisierung*.
Speziell zum Themenkomplex *Funktionale Abhängigkeiten* und *Normalisierung relationaler Datenbanken* wurde ein computerunterstütztes Lernprogramm benötigt, das den Studenten ergänzend zur Vorlesung die Möglichkeit bieten sollte, praktische Fertigkeiten in der Anwendung der im Themenkomplex behandelten Algorithmen zu festigen und auszubauen.
Da zum Zeitpunkt des Projektbeginns noch keine vergleichbaren Lerntools existierten, die den verlangten Anforderungen gerecht wurden, wurde das Ziel festgesetzt, ein webbasiertes, multimediales Lernprogramm zu entwickeln, mit dem die Anwendung der Algorithmen über eine grafische Benutzeroberfläche anhand von Übungsaufgaben geübt werden kann.
Mit Hilfe eines zu implementierenden Bewertungsmechanismus sollten die vom Benutzer eingegebenen Lösungen bewertet und Hinweise zu fehlerhaften Lösungsschritten ausgegeben werden können. Desweiteren sollte das Lernmodul theoretische Inhalte enthalten, mit denen das nötige Wissen zur Funktionsweise und Anwendung der Algorithmen in Text- und Videoform vermittelt wird.

1.3 Inhalt und Aufbau dieser Arbeit

Thematische Übersicht

Diese Arbeit beschreibt das entwickelte Lernmodul und beschäftigt sich dabei mit didaktischen Theorien und Konzepten, deren Anwendung im Kontext des webbasierten Trainings und deren Umsetzung im *Lernmodul Normalisierung*. Darüber hinaus wird die Architektur und die Implementierung der Software erläutert sowie weitere Anwendungs- und Einsatzmöglichkeiten der enthaltenen Funktionalität beziehungsweise des Programms beschrieben.
In dem an die Einleitung anschließenden Kapitel 2 werden zunächst allgemeine Lerntheorien und deren Einfluss auf computerbasierte Lernumgebungen beschrieben. Daran anknüpfend werden theoretische Aspekte des didaktischen Designs im Zusammenhang mit Lernsoftware vorgestellt. Anschließend wird die konkrete Umsetzung der vorher behandelten didaktischen Aspekte in der entwickelten Lernumgebung erläutert.
Kapitel 3 wird zunächst ein Überblick über verschiedene Softwarearchitekturmodelle von Web-Anwendungen gegeben. Darauf aufbauend wird die Softwarearchitektur des erstellten Lernmoduls und die darin enthaltenen Komponenten erläutert und eine Beschreibung der Vorteile dieser Architektur geliefert.
Kapitel 4 umfasst die technische Dokumentation der Software. Einzelne Programmkomponenten und Funktionen werden dabei beschrieben und deren Funktionsweise erläutert.
In Kapitel 5 wird die Anwendung der implementierten Algorithmen zur Normalisierung anhand von einigen Beispielen erklärt.
Kapitel 6 schließlich liefert zusammenfassend einen Überblick über Konzeption und Realisierung der Software und beschreibt anschließend weitergehende Einsatz- und Erweiterungsmöglichkeiten des entwickelten Lernmoduls.

Inhalt der beigelegten CD-ROM

Dieser Arbeit ist eine CD-ROM beigelegt, auf der die folgenden Dateien enthalten sind:

- Diese Arbeit als PDF-Datei (Dateiname: ba_lernmodul.pdf) im Verzeichnis *text*.
- Der komplette Programmcode der Lernumgebung im Verzeichnis *lernmodul*.
- Der Webbrowser Mozilla Firefox 2.0.06 im Verzeichnis *software*. Dieser wird zum Starten des Programms benötigt. Downloadadresse: `http://www.mozilla-europe.org/de/products/firefox/`.
  Alternativ zu Firefox kann jeder beliebige anderen Webbrowser benutzt werden.
- XAMPP in der aktuellsten Version 1.6.3 im Verzeichnis *software*. Mit XAMPP ist es möglich, einen Apache-Webserver, einen MySQL-Datenbankserver sowie PHP zu installieren.

Hinweise zur Installation und Konfiguration

Nach der Installation von XAMPP können Sie über das XAMPP Control Panel den Apache Webserver und den MySQL Datenbankserver starten. Kopieren Sie anschließend den kompletten Ordner *lernmodul* von dieser CD in das XAMPP-Verzeichnis *htdocs*. Die benötigte MySQL-Datenbank müssen Sie nicht selbst anlegen, da dies automatisch vom Programm erledigt wird.
Zu dem Lernmodul gelangen Sie anschließend, indem Sie Ihren Webbrowser starten und zur Adresse `http://localhost/lernmodul` navigieren.

Kapitel 2

Didaktische Konzeption

Dieses Kapitel befasst sich mit Grundlagen und Konzepten der Didaktik, die für die Gestaltung webbasierter Lernsoftware von Bedeutung sind, und deren Umsetzung im *Lernmodul Normalisierung*. In diesem Zusammenhang werden die wichtigsten Lerntheorien und einige Ansätze zur Gestaltung von Lernsoftware vorgestellt. Diese theoretischen Grundlagen werden schließlich mit der Beschreibung von didaktisch-methodischen Elementen und der Umsetzung im *Lernmodul Normalisierung* konkretisiert.

2.1 Lerntheorien

Holzinger fasst die verschiedenen Lerntheorien zu den Hauptströmungen *Behaviorismus*, *Kognitivismus* und *Konstruktivismus* zusammen (Holzinger, 2001, Seite 110). In Anlehnung daran werden im Folgenden die jeweiligen wichtigsten Lerntheorien vorgestellt.

2.1.1 Behaviorismus

Beim Behaviorismus steht die Untersuchung beobachtbarer menschlicher Reaktionen auf externe Stimuli im Vordergrund. Ziel dieser Untersuchung ist es, bestimmte Verhaltensweisen von Menschen objektiv zu beschreiben. Dabei wird angenommen, dass das gezeigte Verhalten ausschließlich durch externe Faktoren gesteuert wird und nicht von kognitiven Prozessen abhängt. Die Prozesse, die beim Lernen im Gehirn ablaufen, sind demnach für die Untersuchung nicht relevant.
Zentrale lerntheoretische Aussagen wurden von Thorndike in seinem „Gesetz des Effekts“ (*law of effect*) und später von Skinner in seiner Theorie der *„operanten Konditionierung“* formuliert.

Thorndike’s „law of effect“

Nach Thorndike’s Aussagen haben Reaktionen, die vom Lerner als befriedigend oder unbefriedigend wahrgenommen werden, einen starken Einfluss auf das Lernverhalten. Dabei spielt der Erfolg eine große Rolle: Ein Erfolgserlebnis führt dazu, dass eine Handlungsweise mit höherer Wahrscheinlich-

keit erneut gezeigt wird, als eine, die nicht mit einem Erfolg in Verbindung gebracht werden kann.

Skinner's „operante Konditionierung"

Skinner's Aussagen zum Lernverhalten bauen auf denen von Thorndike auf: Seine Theorie der operanten Konditionierung besagt, dass die Wahrscheinlichkeit, dass ein Verhalten zukünftig wiederholt oder unterlassen wird, durch bestimmte Konsequenzen des Verhaltens beeinflusst wird. Solche Konsequenzen führen entweder zu einer *positiven* oder einer *negativen Verstärkung*: Eine als unangenehme wahrgenommene Konsequenz in Form einer Bestrafung führt zu negativer Verstärkung - die Wahrscheinlichkeit, dass das gezeigte Verhalten erneut auftritt, sinkt. Im Gegensatz dazu führt eine als angenehm wahrgenommene Konsequenz in Form einer Belohnung zu positiver Verstärkung und zu einer Zunahme der Wahrscheinlichkeit, dass das Verhalten erneut gezeigt wird.
In Erweiterung zu Thorndike's Theorie besagt Skinner's Prinzip der operanten Konditionierung, dass Lernende durch gezielte Belohnungen und Bestrafungen in Richtung des Lernziels gelenkt werden sollen (Euler, 1992, Seite 45).
Lernstrategien des Behaviorismus werden besonders in Software zum Erlernen von Faktenwissen angewendet: Lernende erhalten Informationen zu einem bestimmten Sachverhalt und bekommen dazu in einem bestimmten zeitlichen Abstand Fragen präsentiert. Nach Abgabe einer Antwort folgt eine Rückmeldung, mit der das Lernverhalten des Lernenden entweder positiv oder negativ verstärkt werden soll (siehe hierzu: Kerres, 2001, Seite 56).
Um effizientes Lernen zu garantieren, müssen Lernprogramme geeignete Anreize zur positiven Verstärkung des Lernverhaltens liefern.
Ein gutes Beispiel hierfür sind Vokabellernprogramme, bei denen der Nutzer nach der korrekten Angabe einer Übersetzung eines abgefragten Wortes durch eine positive Rückmeldung (Lob, Erfolgserlebnis) zum weiteren Vokabellernen motiviert wird.

2.1.2 Kognitivismus

Im Gegensatz zum Behaviorismus, bei dem kognitive Prozesse gänzlich ausgeblendet werden, stehen beim Kognitivismus eben diese kognitiven Denk- und Verarbeitungsprozesse im Mittelpunkt der Betrachtung.
Abbildung 2.1 zeigt in Anlehnung an Holzinger (Holzinger, 2001, Seite 133) schematisch die kognitivistische Sichtweise des Lernens als Informationsverarbeitungsprozess.

Abbildung 2.1: Kognitivistische Sichtweise des Lernens

Im Lernprozess werden den Lernenden als Eingabe Informationen geliefert. Diese werden vom Lerner aufgenommen, verarbeitet und in Verbindung mit bereits erlangtem Wissen gesetzt und daraus eine Ausgabe generiert. Aus dieser Abhängigkeit von Vorwissen und individueller Verarbeitung ergibt sich, dass sich die Ausgaben von verschiedenen Lernenden unterscheiden können, obwohl auf eine gemeinsame Informationssammlung zurückgegriffen wurde.
Daran wird auch der zentrale Unterschied zum Behaviorismus deutlich: Lernende werden als Individuen angesehen, die externe Reize aktiv verarbeiten und nicht wie in behavioristischen Theorien durch diese gesteuert werden und auf sie reagieren.

Bruner's „Theorie des entdeckenden Lernens"

Ein zentrale Lerntheorie des Kognitivismus ist Bruner's „Theorie des entdeckenden Lernens": Entdeckendes Lernen bedeutet, dass der Lernprozess aktiv und individuell vom Lernenden gesteuert werden kann. Informationen müssen entdeckt und geordnet werden, um anschließend verarbeitet werden zu können. Daraus wird letztendlich Wissen über Problemlösungstrategien abgeleitet.
Wichtige Voraussetzung für entdeckendes Lernen ist die intrinsische Motivation der Lernenden, die das nötige Interesse und die Neugier am selbstständigen Erarbeiten von Lösungsansätzen aufweisen müssen.
Lernsoftware, die nach kognitivistischen Prinzipien konzipiert wird, begleitet den Lernenden bei der aktiven Erarbeitung von Wissen und der Entdeckung von Problemlösungsstrategien. Das Lernprogramm ist dabei ein „helfender Tutor", der die benötigten Informationen bereitstellt und im Bedarfsfall Hilfestellungen gibt.

2.1.3 Konstruktivismus

Lernen ist nach konstruktivistischer Auffassung ein Prozess der aktiven Konstruktion von Wissen. Neues Wissen wird in Verbindung mit Vorwissen und Erfahrungen gesetzt, interpretiert und daraus ein subjektives Abbild der Umwelt konstruiert. Das Lernen vollzieht sich dabei in einem sozialen Kontext: Der Bezug zur Umwelt und die Interaktion mit anderen Lernenden sind wichtige Bestandteile des Lernprozesses. Die Zielsetzung von Lernen ist, Kompetenz zum Lösen von konkreten Problemstellungen zu erlangen.
Der Unterschied zum Kognitivismus besteht somit darin, dass Wissen nicht situationsunabhängig erarbeitet wird, sondern in jeder neuen Situationen unter Rückgriff auf vorhandenes Wissen neu konstruiert wird.
Hinter dieser Sichtweise der Situationsabhängigkeit steckt das konstruktivistische Prinzip des „situierten Lernens".

Das Prinzip des situierten Lernens

Beim situierten Lernen erfolgt der Wissenserwerb im Kontext einer konkreten Lernsituation um dem

Phänomen des sogenannten „trägen Wissens“ entgegenzuwirken. Träges Wissen bedeutet, dass das erlernte Wissen in einer Situation nicht angewendet werden kann, weil es nicht in einem Anwendungskontext oder einem zu abstrakten, realitätsfernen Kontext erlernt wurde.
Damit der Lernende die Kompetenz erlangt, das Wissen in konkreten Anwendungssituationen abrufen und anwenden zu können, muss der Lehrstoff in authentischen Situationen und aus unterschiedlichen Perspektiven vermittelt werden (siehe: Bruns und Gajewski, 2002, Seite 15).
Nach Mandl u. a. (2002) muss der Lernprozess so gestaltet sein, dass ein späterer Wissenstransfer ermöglicht wird. Demnach muss Lernen in Gruppen, unter Verwendung von Hilfsmitteln und der Berücksichtigung der Anwendungsbedingungen erfolgen. Für die Gesaltung von Lernumgebungen nach situativem Ansatz stellen die Autoren folgende Anforderungen zusammen.

- Lernende sollen zur aktiven Auseinandersetzung mit dem Lernstoff motiviert werden. Dazu werden zu Beginn des Lernprozesses interessante und herausfordernde Problemstellungen geliefert, die den Lerner antreiben sollen, ein Problem lösen zu wollen.
- Das zu erwerbende Wissen soll im Anwendungskontext in authentischen Situation und in Verbindung mit realistischen Problemstellungen bereitgestellt werden.
- Das Wissen soll aus unterschiedlichen Perspektiven angeboten und dargestellt werden, damit Lernende sich von verschiedenen Standpunkten mit den Problemstellungen auseinandersetzen um so die Fähigkeit zu erlernen, das gelernte Wissen auch auf einen anderen Kontext anwenden zu können.
- Die Kommunikation und Kooperation mit anderen Lernenden soll ermöglicht werden, um den Wissens- und Erfahrungsaustausch und das gemeinsame Problemlösen im Team zu ermöglichen.

2.2 Konzepte des Computer Based Training

Im vorangehenden Abschnitt wurden die wichtigsten Lerntheorien vorgestellt, aus denen Konzepte abgeleitet werden können, die bei der Entwicklung computer- und webbasierter Lernumgebungen eine wichtige Rolle spielen. In diesem Abschnitt werden drei wesentliche Konzepte mit den jeweiligen Eigenschaften vorgestellt und beschrieben.

2.2.1 Programmierte Instruktion

Mit dem Konzept der programmierten Instruktion werden behavioristische Lerntheorien auf die Gestaltung computerbasierter Lernprogramme angewendet (Kerres, 2001, Seite 58): Die von Skinner getroffenen Aussagen zur operanten Konditionierung werden dabei so umgesetzt, dass der zu vermittelnde Lerhstoff zunächst in sogenannte „Lehrstoffatome“ unterteilt und dem Lernenden präsentiert wird. Nach der Präsentation einer Informationseinheit wird dem Lerner eine Frage gestellt. Nach der

Beantwortung erhält er anschließend ein Feedback und wird bei einer korrekten Antwort zur nächsten Einheit weitergeleitet. Das Feedback wird in diesem Fall zur positiven Verstärkung des gezeigten Verhaltens genutzt. Bei einer fehlerhaften Antwort wird möglichst kein Feedback gegeben, sondern lediglich zur Präsentation des Stoffes zurückgesprungen. Unter Verwendung dieser „Frage-Antwort-Muster“ wird sequentiell der gesamte Lehrstoff abgearbeitet, bis das angestrebte Ziel erreicht ist.
Kerres beschreibt im Weiteren eine durch Norman Crowder vorgenommen Weiterentwicklung dieses linearen Vorgehens hin zu einem verzweigten System: Unter Verwendung von „multiple choice Fragen“ kann in Abhängigkeit der abgegebenen Antwort auch zu anderen Informationseinheiten gesprungen werden.
Lernsoftware, die dem Konzept der programmierten Instruktion folgt, kann nach Euler in Form von Übungsprogrammen, sogenannten „drill-and-practice“-Programmen, zur Überprüfung von Faktenwissen eingesetzt werden (Euler, 1992, Seite 21 f.).

„Drill-and-practice“-Übungsprogramme

Solche Übungsprogramme bedienen sich einer Ansammlung von Aufgaben- und Fragestellungen, die nach speziellen Kriterien ausgewählt und dem Lerner gestellt werden. In Anlehnung an Euler veranschaulicht Abbildung 2.2 den grundsätzlichen Ablauf in einem Übungsprogramm.

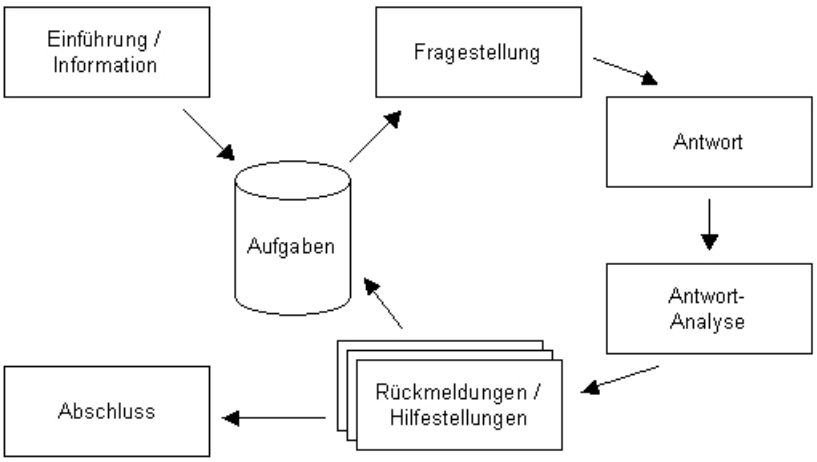

Abbildung 2.2: Ablauf in einem Übungsprogramm

Nach einer kurzen Einführung werden aus der Aufgabendatenbank Aufgabenstellungen und Fragen abgerufen, die dem Anwender präsentiert werden. Nach der Beantwortung durch den Lerner erfolgt die Analyse der abgegebenen Antwort mit anschließender Rückmeldung, die neben Hinweisen zur Korrektheit auch detaillierte Informationen zur Art des Fehlers oder Hilfestellungen enthalten kann. Aus programmtechnischer Sicht gestaltet sich die Generierung von Hinweisen und Hilfestellungen umso einfacher, je geschlossener die Antwortmöglichkeiten des Lerners sind, wie es etwa bei den multiple-choice-Aufgaben der Fall ist. Bei offenen Antwortmöglichkeiten ist zum Teil eine komplizierte Analyse der getätigten Benutzereingaben nötig.

Strittmatter und Niegemann sprechen in diesem Zusammenhang von „Fehleranalysen“, die es ermöglichen, differenzierte Meldungen zu den jeweiligen Fehlern zu geben um so „fachdidaktischen Anforderungen“ gerecht zu werden und um den Nutzen computerbasierten Lernens zu vergrößern (Strittmatter und Niegemann, 2000, Seite 131 f.).
Als Anwendungsmöglichkeiten von Übungssoftware des Typs *drill-and-practice* nennt Euler zum Einen die Prüfungsvorbereitung während der Ausbildung: Dabei bereiten sich die Auszubildenden durch die Beantwortung von Fragen früherer Prüfungen auf ihre eigene vor. Am Ende berechnet das Programm die erreichte Punktzahl und gibt die entsprechende Note aus.
Zum Anderen können Übungsprogramme auch beim Lernen von Sprachen und hierbei speziell beim Lernen von Vokabeln eingesetzt werden.

Adaptive generative Systeme

Eine Weiterentwicklung dieser klassischen Programmansätze der programmierten Instruktion stellen laut Holzinger *adaptive generative Systeme* dar (Holzinger, 2001, Seite 193): Ausgehend von der Annahme, dass die Anpassung der Aufgabenschwierigkeit an die Bedürfnisse der Lernenden ein wichtige Voraussetzung für die Lerneffektivität sei, wurden Systeme entwickelt, die beliebige Aufgaben unterschiedlichen Schwierigkeitsgrades generieren könnnen.
Bei der Entwicklung stellen sich dabei die Probleme, dass mögliche Aufgabentypen beschrieben werden müssen, um unterschiedliche Aufgaben generieren zu können. Außerdem muss für jeden Aufgabentyp ein bestimmtes Vorgehen zur Lösung einer Aufgabe existieren sowie Kriterien zur Feststellung einer erfolgreichen Bearbeitung definiert werden. Ferner muss eine Ablauffolge festgelegt werden, in der die Sequenz der Bearbeitung einzelner Aufgabentypen spezifiziert wird.
Die Anwendung solcher adaptiven generativen Systeme, die Aufgaben selbst generieren können, eignen sich nach Holzinger für Themengebiete, deren Lerhstoff sehr strukturiert ist, wie zum Beispiel Mathematik oder Physik.

2.2.2 Tutorielle und Intelligente Tutorielle Systeme

Tutorielle Systeme sind den Übungsprogrammen der programmierten Instruktion ähnlich. Gemeinsamkeiten bestehen in der Abfrage des Lerhstoffes mit anschließender Antwortanalyse und daraus generierter Rückmeldungen (Riser u. a., 2002, Seite 68).
Der wesentliche Unterschied liegt darin, dass den Lernenden in einem tuoriellen System Freiheiten eingeräumt werden, Wissen aktiv zu erarbeiten. Das bedeutet, dass ein solches Lernprogramm ein „entdeckendes Lernen“ erlaubt (Euler, 1992, Seite 17). Die Lernsoftware übernimmt nach diesem Ansatz die Rolle eines Tutors, der den Lernenden beim Wissenerwerb begleitet und unterstützt, die nötigen Informationen liefert und zum Abschluss einer gewählten Lerneinheit Fragen stellt. Die Antworten werden anschließend analysiert und kommentiert und entsprechende Lernhilfen angeboten oder zu neuen Lerneinheiten verzweigt.

Tutorielle Systeme verknüpfen demnach die Vermittlung von Wissen mit dem Ansatz der programmierten Instruktion, bei dem anhand von problemspezifischen Aufgabenstellungen bereits vorhandenes Wissen abgefragt und kontrolliert wird.
Je nach technischer Implementierung kann der Lehr- und Lernprozess dabei mehr oder weniger offen erfolgen: Lernende können sich Inhalte und Wissen aktiv im Sinne des entdeckenden Lernens oder aber innerhalb vorgegebener Lerhsequenzen erarbeiten.
Die anschließende Abfrage und Kontrolle des Wissensstandes kann dabei in ähnlich offener oder vorgegebener Form erfolgen. Lernende können im Anschluss an die aktive und selbstständige Erarbeitung von Wissen aus einem Aufgabenpool die jeweiligen Aufgabestellungen auswählen und lösen, oder aber sie erhalten direkt im Anschluss an eine Lernsequenz automatisch die entsprechenden Fragestellungen.
Bei der Bewertung der abgegebenen Antworten tritt das System entweder als helfender Tutor auf, der die Lösungen kommentiert und Korrekturvorschläge sowie Lösungshilfen bietet. Oder es tritt als strikter Instruktor auf, der im Sinne des drill-and-practice-Ansatzes lediglich die Korrektheit der Lösungen prüft und zu einer weiteren Fragestellung übergeht.

Intelligente tutorielle Systeme (ITS)

Intelligente turorielle Systeme (ITS) sind Erweiterungen der herkömmlichen tutoriellen Systeme mit Methoden der Künstlichen Intelligenz. ITS analysieren das Verhalten und den Lernfortschritt der Lernenden und leiten davon individuelle Lernpfade ab, mit denen das Lernziel erreicht werden soll. Daneben sind ITS in der Lage, pädagogische Prinzipien und Lerntheorien anzuwenden und individuell, je nach Notwendigkeit im Lernprozess des Benutzers einzusetzen (Steinmetz, 2000, Seite 825).
Abbildung 2.3 (in Anlehnung an Steinmetz, 2000, Seite 826) verdeutlicht den Aufbau intelligenter tutorieller Systeme.
Zu den einzelnen Komponenten eines ITS liefert Schulmeister treffende Beschreibungen (Schulmeister, 2007, Seite 176 ff.). Daran angelehnt werden diese im Folgenden überblicksartig vorgestellt.

Wissens- / Expertenmodell

Das Wissensmodell repäsentiert den gesamten Wissensschatz und ist somit Informationsquelle des Systems. Es enthält deklaratives Wissen über die Begriffe des Anwendungsgebietes (bspw. Kraft und Geschwindigkeit) und deren Beziehungen untereinander, prozedurales Wissen, bestehend aus Regeln zur Lösung von Problemen, sowie heuristisches Wissen, das Erfahrungs- und Problemlösungswissen beinhaltet. Dieses heuristische Wissen ist dann von Bedeutung, wenn Lernende bei der Lösung von Problemen unterstützt und ihnen Tipps gegeben werden sollen.
Das Wissensmodell wird auch als Expertenmodell bezeichnet, weil es das benötigte Expertenwissen enthält, auf dessen Basis Schlussfolgerungen über den Fortschritt des Lernenden gezogen sowie Problem- und Aufgabenstellungen gelöst werden können.

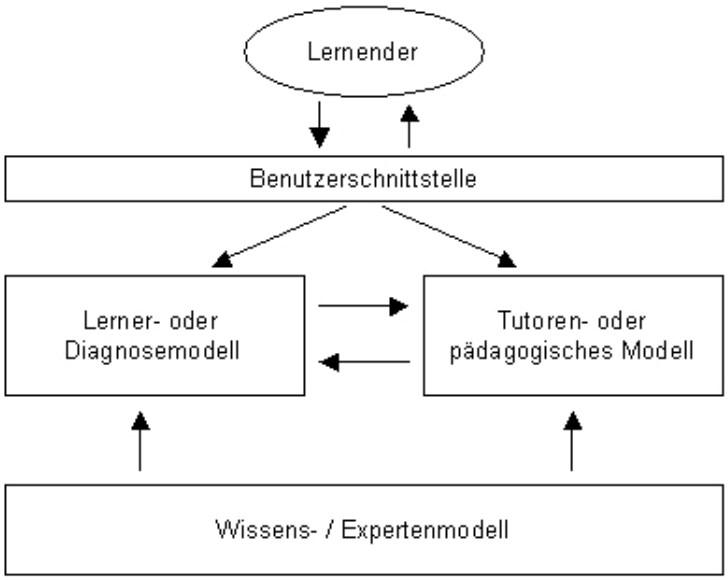

Abbildung 2.3: Aufbau intelligenter tutorieller Systeme

Lerner- oder Diagnosemodell

Im Lernermodell eines ITS wird der aktuelle Wissensstand des Lernenden während des Lernprozesses abgebildet um daraus eine Diagnose über den Lernfortschritt beziehungsweise eventuelle Lerndefizite abzuleiten, daher auch die Bezeichnung Diagnosemodell.

Bei der Diagnose des Wissensstandes der Lernenden existieren zwei Möglichkeiten, um Aussagen über Lernfortschritt oder Lerndefizite zu treffen: Das „subset model“ und das „deviation model“.

Beim subset model wird das Wissen eines Lerners als Teilmenge des Expertenwissens abgebildet. So werden Gemeinsamkeiten zwischen dem Wissensstand des Lerners und dem Gesamtwissen der Domäne, dem Expertenwissen, festgestellt.

Beim deviation model werden Abweichungen zwischen erlerntem Wissen und Expertenwissen ermittelt: Antworten der Lernenden werden analysiert und daraus Schlussfolgerungen über das Verstandene getroffen. Auf Basis des Verstandenen wird anschließend die Abweichung der Fähigkeiten eines Lerners von den Fähigkeiten des Experten ermittelt.

Tutoren- oder pädagogisches Modell

Das Tutorenmodell repräsentiert den Lehrer beziehungsweise dessen Entscheidungsverhalten. Auf Basis der Analyse von Unterschieden zwischen Lerner- und Wissensmodell werden pädagogisch sinnvolle Anweisungen generiert, um bei erkannten Lerndefiziten einzuschreiten und den Lernenden zu dabei zu unterstützen, diese Wissensdifferenzen zu verringern.

Dabei existieren zwei mögliche Arten, wie das System als Lehrer einschreiten kann: In Form des „sokratischen Dialogs“ oder des „Coachings“. Beim sokratischen Dialog wird der Lerner in einem Dialog mit dem System durch den Einsatz gezielter Fragen gelenkt, eigene Fehler kritisch zu be-

trachten und zu analysieren um so die Lösung aus eigener Kraft zu finden. Das Coaching verfolgt das Ziel, Aufgabenstellungen zu präsentieren, die der Lerner lösen muss um Fertigkeiten und Problemlösungsstrategien einzuüben oder auszuprobieren. Währenddessen nimmt das System die Rolle des wartenden *Trainers* ein, der auf Anfragen hin mit Tipps und Hinweisen unterstützend einschreitet.

Benutzerschnittstelle
Über die Benutzerschnittstelle erfolgt die Interaktion zwischen System und Lernenden. Schulmeister unterscheidet dabei vier mögliche Arten, zu denen die vorangehend beschriebenen Formen des sokratischen Dialogs und des Coachings gehören. Darüber hinaus existieren folgende Interaktionsformen:
„Learning by doing“: Hierbei nimmt das System eine aktive Rolle ein, die den Lernenden dazu auffordert, Informationen auszuwählen. Daraus leitet das System anschließend Abweichungen vom Expertenmodell ab.
„Learning while doing“: Diese Form ähnelt dem Coaching. Allerdings nimmt das System hierbei eine aktivere Rolle ein: Hilfestellungen werden nicht nur auf Anfragen hin präsentiert, sondern im Bedarfsfall aktiv während des Lernprozesses ausgegeben.

2.2.3 Explorative Systeme

Explorative Systeme sind Lernsysteme, die offen gestaltet sind. In ihnen kann der Lerner neues Wissen in einem entdeckenden und forschenden Prozess autonom erarbeiten (Kerres, 2001, Seite 217 ff.). Ausgehend von den Schilderungen des Autors, werden nachfolgend die Eigenschaften solcher Lernsysteme beschrieben.
Um ein erfolgreiches Explorieren der Lerninhalte zu ermöglichen, müssen Informationen und Lernangebote trotz der offenen Struktur sinnvoll geordnet werden.
Lernen in explorativen Lernumgebungen bedarf nicht der Vorgabe eines Ziels oder eines bestimmten Lernweges. Das Lernen muss durch solche Systeme gefördert werden und einen Rahmen bieten, sich selbstständig mit Informationen auseinanderzusetzen. Explorative Lernarrangements lassen sich mit folgenden Merkmalen charakterisieren:

- Der Lernprozess wird dadurch angestoßen, dass der Lerner selbst ein Ziel, etwas zu wissen oder zu können, definiert.
- Ausgehend von diesem Lernziel werden Lernaktivitäten gewählt, die für die Zielerreichung geeignet sind.
- Der Lernprozess findet nicht sequentiell sondern „spiralförmig“ statt: Ohne eine fest vorgegebene Sequenz „tastet“ sich der Lernende voran und erkundet die ihm zur Verfügung stehenden Informationsangebote. Dabei können auch Punkte erreicht werden, von denen aus ein weiteres Voranschreiten nicht möglich ist und zu einer früheren Stelle zurückgekehrt werden muss.

- Die Motivation der Lernenden ergibt sich aus dem Interesse, das Lernziel eigenständig zu erreichen. Bereits das Erkunden des Informationsangebots wird als befriedigend empfunden, sodass Eingriffe gemäß behavioristischen Paradigmen nicht nötig sind.

Ausgangspunkt explorativer Systeme ist die menschliche Neugierde, neues Wissen und Problemlösungsstrategien selbstständig zu erarbeiten. Um solche Lernprozesse zu ermöglichen und zu unterstützen, müssen Lernangebote logisch strukturiert sein und Orientierungshilfen gegeben werden. In diesem Zusammenhang spielen hypertextartige Informations- und Bildungssysteme eine entscheidende Rolle.

Das Prinzip des Hypertext

Hypertextsysteme eignen sich sehr gut zur Implementierung explorativer Lernangebote, indem das Lehrangebot in kleine, logisch strukturierte Informationseinheiten aufgeteilt wird. Mit Hilfe von *Hyperlinks* können Teile dieser Einheiten miteinander verknüpft werden.
Je nach Umfang des Lernsystems werden Orientierungs- und Navigationshilfen angeboten, um das Zurechtfinden innerhalb des Systems zu ermöglichen und zu vereinfachen.
Orientierungshilfen dienen der inhaltlichen und räumlichen Orientierung: Inhaltliche Orientierungshilfen liefern Informationen zu den gerade besuchten Lerninhalten. Dazu gehören beispielsweise Gliederungen oder kurze Zusammenfassungen, die es den Lernenden ermöglichen, schnell einen Überblick über die Informationseinheiten zu bekommen.
Räumliche Orientierungshilfen liefern den Lernenden Anhaltspunkte, in welchem Bereich oder an welcher Stelle sie sich innerhalb der Lernumgebung befinden.
Im Zusammenhang mit räumlichen Orientierungshilfen finden Navigationshilfen ihre Anwendung: Mit Hilfe von Lesezeichen können interessante oder wichtige Informationen markiert und wiedergefunden werden. Mittels Pfadangaben kann darüber hinaus ein bereits besuchter Bereich der Lernumgebung identifiziert werden und einfacher zu einer früheren Stelle zurücknavigiert werden, falls beispielsweise ein Punkt erreicht wurde, von dem aus das Lernziel nicht mehr erreicht werden kann.

2.3 Didaktisch-methodisches Design

In diesem Abschnitt folgt nun die Beschreibung des Designs von wichtigen didaktischen und methodischen Elementen, die bei der Konzeption von multimedialen Lernumgebungen berücksichtigt werden müssen.

2.3.1 Zielgruppe und Lernsituation

Bei der Konzeption multimedialer Lernangebote ist es zunächst notwendig, die Zielgruppe und deren Lernsituation zu definieren und zu analysieren. Nach Kerres (2001) ist dieser Schritt sehr wichtig, damit das Lernangebot genau auf diese Zielgruppe hin geplant werden kann. Dazu beschreibt der

Autor wichtige Merkmale, die es zu analysieren gilt.

Neben soziodemografischen Daten über die Zielgruppe wie zum Beispiel Alter, Bildung und technischem Anwendungswissen zu Computern, erfolgt die Charakterisierung der Zielgruppe anhand der im folgenden skizzierten Merkmale.

Vorwissen

Die in der Lernsoftware angebotenen Inhalte und angewendeten Methoden müssen sich am Vorwissen der Anwender orientieren. Kenntnisse über das Vorwissen der Zielgruppe ist besonders dann von Bedeutung, wenn es um die Konzeption der Strukturiertheit eines Lernangebotes geht: Die im vorangehenden Abschnitt beschriebene Methodik der programmierten Instruktion oder der tutoriellen Ansätze eignen sich besonders dann, wenn wenig Vorwissen vorhanden ist. Gerade in tutoriellen Systemen wird dem Lernenden ein unterstützender Tutor zur Seite gestellt, der ihn im Verlauf des Lernprogramms begleitet.
Explorative Ansätzen sind nach Kerres (2001) dann sinnvoll, wenn die Anwender bereits ein gewisses Vorwissen mit sich bringen, da diese sich in einer offenen Lernumgebung besser zurechtfinden können.

Lernmotivation

Je nach Motivation der Lernenden ist eine andere Art der Lernstruktur und -organisation zu wählen. Dabei ist zu unterscheiden, ob Lernende extrinsisch, das heißt durch äußere Faktoren wie beispielsweise Anerkennung von Freunden oder der Erwerb eines Zertifikates, oder aber intrinsisch, das heißt aus eigenem Interesse, motiviert sind.
Der entscheidende Unterschied zwischen intrinsisch und extrinsisch motivierten Lernern liegt darin, wie diese Lernaktivitäten wahrnehmen: Intrinsisch motivierte bringen bereits das Interesse mit, sich mit dem Lehrstoff aktiv auseinanderzusetzen. Für solche Lerner empfiehlt Kerres offen gestaltete, explorative Lernsysteme, in denen sich frei bewegt werden kann und die umfangreiche Informationen zu dem jeweiligen Lernstoff bereitstellen.
Bei extrinsisch motivierten Personen bedarf es demgegenüber der Einteilung des Lernstoffes in kleine Einheiten, die durch Präsentationen vermittelt und in deren Anschluss Tests durchgeführt werden. Auch ist es wichtig, dass Rückmeldungen zum aktuellen Lernfortschritt gegeben werden und.

Über diese Merkmale der Zielgruppe hinaus sind weitere relevante Aspekte bei der Analyse der Zielgruppe zu beachten, wie beispielsweise Lerndauer und Lerngewohnheiten aber auch Einstellungen und Erfahrungen zu beziehungsweise mit Computern, um mögliche Probleme bei der Bedienung der Lernsoftware durch eine entsprechende Gestaltung der Benutzeroberfläche oder Hilfestellungen zur Bedienung zu vermeiden.

2.3.2 Lerngegenstand

Die Grundlage der Konzeption computer- beziehungsweise webbasierter Lernangebote bilden die Lerngegenstände wie beispielsweise Themenkomplexe, Aufgabenstellungen oder Anwendungsfälle. Sie müssen je nach Zielgruppe und Anwendungszweck der Lernsoftware identifiziert und gesammelt werden.
Zur inhaltlichen Strukturierung des Lernangebotes werden zunächst alle relevanten Informationen des Lerngegenstandes gesammelt und gegliedert. Anschließend werden die so identifizierten Lerninhalte nach ihrer Relevanz für das Lernangebot priorisiert und gefiltert (Kerres, 2001, Seite 149 ff.).
Zunächst wird in Zusammenarbeit mit Experten eine umfassende Materialsammlung bestehend aus Informationen, Themen, Aufgabenstellungen, Problembereichen und Anwendungsfällen zusammengetragen. Ist die Materialsammlung abgeschlossen, werden die Lerninhalte strukturiert und gegliedert, um so eine bessere Handhabung zu gewährleisten.
Im anschließenden Schritt der Priorisierung und Reduktion wird eine zielgerichtete Bewertung der gesammelten Materialien vorgenommen: Es wird dabei entschieden, welche Lerninhalte letztlich in der fertigen Software enthalten sein werden. Anhand der im ersten Schritt erstellten Gliederung werden diejenigen Inhalte identifiziert, die für das Erreichen des zugrunde liegenden Lernziels am besten geeignet sind. Weniger gut geeignete Materialien werden ausgeschlossen.

2.3.3 Aufbereitung der Lerninhalte

Nach der Sammlung und Auswahl der geeigneten Lerninhalte erfolgt deren Aufbereitung unter Berücksichtigung medialer und methodischer Prinzipien: Unter Verwendung medialer Elemente werden die Informationen zur Präsentation aufbereitet. Die Art und Weise, wie letztlich auf die Informationen zugegriffen werden kann, wird mit methodischen Elementen verwirklicht.

Mediale Elemente

Bruns und Gajewski (2002) nennen als mediale Elemente Text, Bild und Video, die zur Präsentation der Informationen verwendet werden. Jedes dieser Elemente hat spezifische zu berücksichtigende Eigenschaften und Einsatzmöglichkeiten zur Wissenspräsentation.

Bei der Verwendung von Texten spielt das Design eine wichtige Rolle: Um die Lesbarkeit eines Textes optimal zu gestalten, müssen gestalterische Aspekte bezüglich der verwendeten Schriftarten, Formatierungen und Strukturierungen berücksichtigt werden.
Die Übersichtlichkeit von Texten kann durch Einteilung eines Textes in Abschnitte unter Verwendung von Überschriften, Absätzen und Texteinrückungen erreicht werden. Außerdem können durch Verwendung anderer Schriftgrößen oder -stile wichtige Informationen hervorgehoben werden.
Beim Lesen von Texten wird dem Leser ein hohes Maß an Konzentration abverlangt, was dazu führt, dass das Durchlesen eines Textes besonders am Bildschirm als anstrengend empfunden wird. Nach

Bruns und Gajewski (2002) sollte beim Umgang mit Texten daher eher sparsam umgegangen werden, und wenn möglich, bessere Darstellungsmöglichkeiten von Sachverhalten vorgezogen werden (Seite 86).
Bilder und insbesondere Grafiken eignen sich besonders dann, wenn logische Strukturen und komplexe Zusammenhänge dargestellt werden sollen. Beispielsweise kann durch Verwendung von „mindmaps“ ein umfangreicher Sachverhalt sehr übersichtlich und anschaulich visualisiert werden. Zeitliche Zusammenhänge oder spezielle Arbeitsabläufe lassen sich mit Hilfe von Flussdiagrammen und Mengenrelationen mit Hilfe von Kreis- oder Säulendiagrammen sehr gut darstellen.
Schließlich können durch den Einsatz von Videos und Animationen unterschiedlichste Abläufe sehr anschaulich präsentiert werden. Die Autorinnen nennen in diesem Zusammenhang den Einsatz von Lehrfilmen, um beispielsweise biologische Stoffwechselprozesse oder die Funktionsweise eines Motors in ansprechender und interessanter Form zu erklären.

Methodische Elemente

Methodische Elemente kommen zur Gestaltung der Art und Weise, wie das Wissen innerhalb der Lernumgebung durch die Lernenden erschlossen werden kann, zum Einsatz. In diesem Zusammenhang existiert eine Vielzahl solcher Elemente, von denen im Folgenden in Anlehnung an Bruns und Gajewski (2002) und Kerres (2001) die wichtigsten vorgestellt werden.

Dokumentensammlungen

Dokumentensammlungen werden bei webbasierten Lernangeboten als umfangreiche Wissensdatenbank benutzt. Bei der Entwicklung eines solchen Lernangebotes werden, wie bereits erwähnt, die relevanten Lerninhalte gesammelt. Diese können anschließend den Lernenden in aufbereiteter Form als Lehr- und Lernmaterialen über eine Dokumentensammlung zugänglich gemacht werden. Jeder Kursteilnehmer kann auf diese Sammlung zugreifen und eigenen Dokumente hinzufügen.

Lexikon

In einem Lexikon kann über Schlüsselbegriffe zu den angebotenen Themen und Lerninhalten gezielt nach weiteren Informationen gesucht werden. Durch spezielle Suchmechanismen kann dabei das Auffinden dieser Informationen erleichtert werden.

Interaktive Übungen und Tests

Durch den Einsatz von Übungen und Tests können Lernende Wissen interaktiv erarbeiten beziehungsweise testen.
In behavioristisch geprägten Lernprogrammen kommen solche Übungen zur Überprüfung des Wissenstandes und zur Abfrage von Faktenwissen zum Einsatz. Die Interaktivität in Form von Rückmeldungen wird dabei so gestaltet, dass der Lernende mittels positiver Antworten des Systems in seinem Lernverhalten bekräftigt wird. Eine solche Rückmeldung gibt dem Lernenden somit einen

Hinweis, dass die Frage richtig beantwortet wurde. Zudem hat sie einen motivationalen Effekt, da durch ein Lob zum weiteren Beantworten von Fragen ermuntert wird (Kerres, 2001, Seite 201).

Sequenzielle und logische Strukturierung des Lernangebots

Die Art der Strukturierung des Lernangebotes entscheidet darüber, wie der Ablauf einer Übungssitzung gestaltet ist.

Bei einer sequenziellen Struktur erfolgt der Ablauf einer Sitzung in einer linearen Form und unter Kontrolle des Lernprogramms (siehe hierzu: Kerres, 2001, Seite 192 ff.). Das grundlegende Schema sieht dabei vor, dass zu Beginn einer Sequenz die nötigen Informationen und das zu vermittelnde Wissen präsentiert werden. Der Lernende beschäftigt sich mit den angebotenen Inhalten und absolviert anschließend einen Test. Im Anschluss an den Test erfolgt die Rückmeldung und der Lerner wird zu einer weiteren Sequenz weitergeleitet. Dabei kann die Weiterleitung je nach technischer Realisierung nach einem fest vorgeschriebenen Schema ablaufen oder in Abhängigkeit der gegebenen Antwort erfolgen.

Logisch strukturierte Lernangebote werden dahingegen in offenen, explorativen Lernprogrammen verwendet. Der entscheidende Unterschied zu sequenziell organisierten Lernangeboten liegt darin, dass durch die logische Strukturierung ein exploratives Lernen unterstützt wird, bei dem sich die Lernenden frei durch ein Angebot an Lernmaterialien bewegen können.

Hilfestellungen

Hilfestellungen kommen in webbasierten Lernprogrammen zum Einsatz, weil hier nicht die Möglichkeit besteht, dass bei Schwierigkeiten eine Lehrperson beziehungsweise ein Tutor um Hilfe gebeten werden kann. Um diese Möglichkeit auch in einer Lernsoftware zu geben, müssen Hilfestellungen integriert werden (Bruns und Gajewski, 2002, Seite 48). Das kann beispielsweise durch die Integration einer textuellen Hilfefunktion innerhalb des Programms erfolgen.

Unter Verwendung von Methodiken der künstlichen Intelligenz kann die Lernumgebung darüber hianus selbst die Rolle eines Tutors einnehmen wie es bei intelligenten tutoriellen Systemen der Fall.

2.4 Didaktische Konzeption des Lernmoduls Normalisierung

Im folgenden Abschnitt wird nun die konkrete Umsetzung der in den vorangehenden Abschnitten beschriebenen didaktischen Theorien und Konzepte im *Lernmodul Normalisierung* erläutert.

2.4.1 Anwendungszweck

Die Entwicklung des Lernmoduls zur Normalisierung relationaler Datenbanken hat die Zielsetzung, im Sinne eines *Blended-Learning*-Ansatzes ein unterstützendes Lernangebot zu Vorlesung Datenbanken für Informationsmanager bereitzustellen. Im Speziellen betrifft dies die Anwendung der Algorithmen zur Normalisierung relationaler Datenbankschemata. Studierende sollen damit die Möglichkeit erhalten, neben der Vorlesung und der dazu angebotener Übung die Lerninhalte des Themenkomplexes Normalisierung relationaler Datenbanken üben und vertiefen zu können.

Die Zielgruppe besteht demnach primär ausschließlich aus den Studenten der Universität Koblenz, die im Sommersemesters 2007 die Vorlesung Datenbanken für Informationsmanager besuchten.
Da das Lernmodul über das Internet frei zugänglich ist, vergrößert sich die Zielgruppe sekundär um all diejenigen, die sich aus Interesse oder schulischen, universitären oder sonstigen Ausbildungszwecken mit der Anwendung der Algorithmen beschäftigen.

2.4.2 Lerninhalte

Die angebotenen Lerninhalte richten sich nach den Inhalten der Vorlesungseinheiten bezüglich der Anwendung der Algorithmen zur Normalisierung relationaler Datenbankschemata.
Im Einzelnen betrifft dies folgende Inhalte:

- **Anwendung der RAP-Regeln**: Die RAP-Regeln Reflexivität, Akkumulation und Projektion werden dazu verwendet, aus einer gegebenen funktionalen Abhängigkeitsmenge weitere funktionale Abhängigkeiten abzuleiten.
  Außerdem können diese Regeln verwendet werden, um für eine funktionale Abhängigkeit zu überprüfen, ob diese in einer funktionalen Abhängigkeitsmenge gilt.

- **Schlüsselberechnung**: Der Schlüssel-Algorithmus wird dazu verwendet, zu einem gegebenen Relationsschema R(V,F) mit einer Attributmenge V und einer Menge funktionaler Abhängigkeiten F einen Schlüssel zu berechnen. Durch einmaliges Anwenden findet man genau einen Schlüssel, abhängig davon, wie die Anordnung der Attribute in V ist. Um alle Schlüssel zu finden, muss der Algorithmus auf jede mögliche Permutation von V angewendet werden.

- **Basisberechnung**: Mit zugrunde liegenden Algorithmus kann eine minimale Basis einer funktionalen Abhängigkeitsmenge berechnet werden.
  Mit der Anwendung des Algorithmus müssen folgenden Minimalitätseigenschaften hergestellt werden:
  (1) Jede Abhängigkeit in F hat auf der rechten Seite nur ein einziges Attribut. (Rechts-Minimalität).
  (2) Man kann keine Abhängigkeit XA durch eine Abhängigkeit YA mit YX ersetzen und weiterhin eine zu F äquivalente Abhängigkeitsmenge haben. (Links-Minimalität): Es gibt keine redundanten Attribute auf der linken Seite einer funktionalen Abhängigkeit.
  (3) Man kann keine funktionale Abhängigkeit in F entfernen und weiterhin eine zu F äquivalente Abhängigkeitsmenge haben. (Redundanzfreiheit).

- **Testen der Lossless Join Property**: Mit dem Algorithmus zum Testen der Lossless Join Property kann geprüft werden, ob eine gegebene Zerlegung eines Relationsschemas R in Teilschemata $R_1 ... R_n$ verlustlos ist. Verslustlos bezieht sich hierbei auf den Verlust von Informationen, der bei einer Zerlegung entstehen kann. Wenn beispielsweise nach einer Zerlegung

eines Relationsschemas R dieses nicht mehr aus den Teilschemata $R_1 ... R_n$ reproduzierbar ist, dann ist diese Zerlegung verlustlos.

- **Anwenden des Synthese-Algorithmus**: Mit dem Synthese-Algorithmus kann ein gegebenes Schema R mit der Attributmenge $U = A_1, ..., A_n$ und einer Menge funktionaler Abhängigkeiten F zerlegt werden. Die resultierende Zerlegung ist in 3. Normalform, erfüllt die Lossless Join Property, und die ursprünglichen Abhängigkeiten bleiben erhalten.

2.4.3 Gestaltung der Benutzeroberfläche

Bei der Gestaltung wurde darauf geachtet, eine schlichte, aber dennoch ansprechend gestaltete und logisch strukturierte Benutzeroberfläche zu erstellen. Abbildung 2.4 veranschaulicht schematisch die Struktur der Benutzeroberfläche.

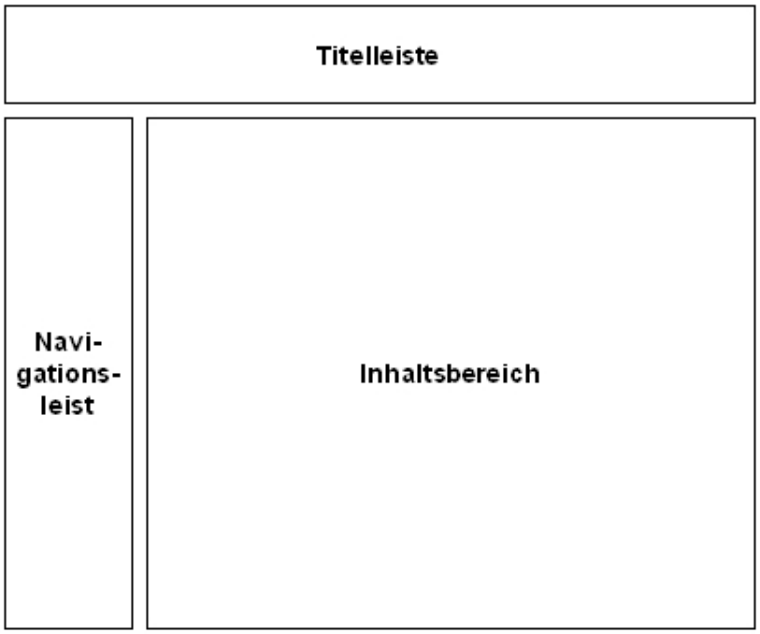

Abbildung 2.4: Schematische Darstellung der Struktur der Benutzeroberfläche

Über die Navigationsleiste im linken Bereich der Oberfläche gelangt man zu den Hauptbereichen des Lernmoduls. Im Inhaltsbereich sind unter anderem die Aufgabenbereiche eingebettet, in denen der Benutzer die Möglichkeit hat, Aufgaben zu lösen und anschließend bewerten zu lassen oder theoretische Informationen zu den Algorithmen nachzulesen. Außerdem kann er sich zu jedem Themenbereich Video-Tutorials anschauen, in denen sowohl die Anwendung der Algorithmen als auch die Bedienung der Benutzeroberfläche anhand einer Beispielaufgabe erklärt wird. Über eine Schaltfläche zum Versenden einer Feedbackmail kann der Benutzer direkt in Kontakt mit dem Betreuer des Lernmoduls treten um Fragen zu stellen und selbst eine Rückmeldung zum Programm zu geben. Abbildung 2.5 stellt den schematischen Aufbau eines Aufgabenbereiches dar.
Die klar und logisch strukturierte Oberfläche verfolgt den Zweck einer einfachen Navigation zwi-

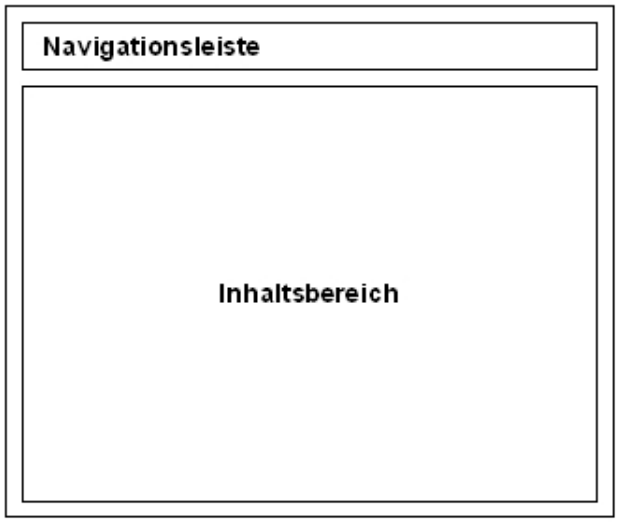

Abbildung 2.5: Schematische Darstellung der Struktur der Aufgabenbereiche

schen den einzelnen Bereichen des Lernmoduls. Durch Orientierungshilfen kann der Benutzer jederzeit erkennen, in welchem Teil des Programms er sich befindet. In jedem Aufgabenbereich wird dazu der Bereichsname als Überschrift des Inhaltsbereich angezeigt, die einzelnen Bereiche eines Aufgabenteils werden dadurch gekennzeichnet, dass die jeweilige Navigationsschaltfläche durch eine fett-gedruckte Beschriftung von den Übrigen abgehoben ist.

2.4.4 Vermittlung der Lerninhalte

Der Großteil der Informationen zu den einzelnen Lerninhalten wird in Textform geliefert. Dabei werden bei längeren Texten Strukturierungshilfen verwendet: Durch farblich hinterlegte Zwischenüberschriften werden einzelne Textblöcke visuell voneinander abgetrennt. Durch die Verwendung von vergrößerten Zeilenabständen wird außerdem eine einfache Lesbarkeit erreicht. Darüber hinaus werden wichtige Begriffe fett gedruckt, um sie vom restlichen Text abzuheben.
Zur Vermittlung der nötigen Hintergrundinformationen und der Lerninhalte wurden sowohl Texte als auch kommentierte Video-Tutorials verwendet.
Um die nötigen theoretischen Hintergrundinformationen zu den Algorithmen zu geben, ist die Textform ausreichend, da die zu vermittelnden Sachverhalte in kurzen Texten ausgedrückt werden können.
Zu jedem Aufgabenbereich werden die Algorithmen erklärt. Dabei erweist sich die Textform als sehr geeignete Präsentationsform. Die einzelnen logischen Passagen der Algorithmen, beispielsweise *if-then-else*-Blöcke, können durch Texteinrückungen voneinander unterschieden werden. Außerdem bietet die textuelle Darstellung der Algorithmen die Möglichkeit, zu jeder Zeile einen Kommentar zu ergänzen, um die Funktionsweise zu erklären.

Die eingesetzten Video-Tutorials bieten daneben eine sehr gute Form, um eine anschaulich und

ansprechend gestaltete Erklärung zur Anwendung der Algorithmen und gleichzeitig zur Bedienung der Benutzeroberfläche zu liefern.

2.4.5 Lehrmethode und Interaktivität

Die verwendete Lehrmethode ist eine Verbindung aus programmierter Instruktion und tutoriellem Ansatz: Die Lösung der Aufgabenstellungen erfolgt interaktiv über Eingabefelder und Schaltflächen. Da wo es möglich ist, werden dem Benutzer bei der Eingabe der Daten und der Auswahl der aktuellen Lösungsschritte Freiheiten gelassen. Das erfolgt mit der Zielsetzung, den Lösungsweg nicht durch das Programm vorzugeben um so eine größere Lerneffizienz zu erzielen.
Durch die Bewertung der Lösungen werden dem Benutzer Rückmeldungen zu richtigen und falschen Lösungsschritten geliefert. Positive Rückmeldungen in Form eines Lobes bei korrekter Lösung sollen der Motivation des Lernenden dienen. Rückmeldungen zu gemachten Fehlern sollen Hilfestellungen und Hinweise liefern, um die Ursache eines Fehlers zu erkennen.
Durch die offene Gestaltung der Benutzeroberfläche wird außerdem die freie Auswahl der Lerneinheiten ermöglicht. Dadurch kann der Benutzer die Themenbereiche je nach Lernbedarf auswählen. Auch in der Abfolge einer Übungseinheit werden keine programmtechnischen Vorgaben gesetzt: Es steht dem Lerner frei, in welcher Reihenfolge er die einzelnen Bereiche eines Aufgabenteils besucht.

Kapitel 3

Softwarearchitektur

3.1 Softwarearchitekturen und Entwurfsmuster

Im Prozess der Softwareentwicklung ist eine Softwarearchitektur das Ergebnis der Designphase, in der die in vorangehenden Phasen spezifizierten Anforderungen an ein Softwareprodukt umgesetzt werden. Softwarearchitekturen liefern technische Spezifikationen für den Aufbau einer Software und deren Einteilung in Komponenten sowie deren funktionale Beziehungen (Balzert, 1996, Seite 632 ff.). Die Struktur der entwickelten Softwarearchitektur wird dabei durch verschiedene Faktoren beeinflusst.
Nicht-funktionale und qualitative Anforderungen an das zu entwickelnde Produkt wie Wiederverwendbarkeit, Änderbarkeit und Effizienz aber auch funktionale und technische Anforderungen an die Gestaltung der Benutzerschnittstelle oder der Einsatz von Datenbanken spielen im Designprozess eine wichtige Rolle. Ebenso beeinflusst die Anzahl der Benutzer und die räumliche Verteilung der Standorte, an denen ein Softwaresystem eingesetzt werden soll, den Grad der Zentralisierung beziehungsweise Dezentralisierung des Softwaresystems und der damit verbundenen Unterteilung in Systemkomponenten.

Je nach technischen und funktionalen Anforderungen an Software und deren Zentralisierung können dabei unterschiedliche Architektur-Strukturen entstehen (siehe Vogel u. a., 2005, Seite 208 ff.). In einem dezentralisierten System werden einzelne Systembelange auf mehrere Bausteine verteilt. Vorteile eines solch verteilten Systems sind beispielsweise niedrigere Hardwarekosten, da auf den Einsatz von teuren und leistungsfähigen Großrechnern in solchen Systemen verzichtet werden kann.

Client-Server-Modell und N-Tier-Architekturen

Eine klassische Architektur verteilter Systeme ist nach Vogel u. a. (2005) die *n-Tier-Architektur*. Das *Client-Server-Modell*, nach dem beispielsweise Rechnernetze von Unternehmen mit verschiedenen Niederlassungen aufgebaut sind, basiert auf einer solchen Architektur.

In der einfachsten Ausprägung, der 2-Tier-Architektur, kommuniziert dabei ein Client mit einem Server in Form von Anfrage-Antwort-Mustern: Der Client sendet eine Anfrage zur Nutzung der gewünschten Dienste an den Server, der als Antwort die geforderten Daten zurücksendet.
Vergrößert sich die Anzahl der Clients, sinkt die Leistungsfähigkeit eines solchen Netzwerkes. Dieses Problem wird durch 3-Tier-Architekturen gelöst, bei denen zwischen Datenbankserver und den einzelnen Clients ein weitere Server zwischengeschaltet wird, der zentrale Aufgaben wie das „Queuing“ und das „scheduling“ einzelner Anfragen übernimmt, und so die Leistungsfähigkeit steigert.
N-Tier-Architekturen mit höheren Ordnungszahlen existieren beispielsweise dann, wenn mehrere 3-Tier-Architekturen miteinander verbunden sind.

3-Schichten-Architektur

Auf funktionaler Ebene werden Standard-Softwaresysteme häufig in drei Schichten unterteilt: Präsentationsschicht, Anwendungsschicht und Persistenzschicht (Dunkel und Holitschke, 2003, Seite 17).
Die Benutzeroberfläche (*graphical user interface - GUI*) stellt die Schnittstelle zwischen Benutzer und System dar: Über sie erfolgt die Interaktion des Anwenders mit der Software. Dabei können mit Hilfe von Bedienelementen wie beispielsweise Textfeldern, Schaltflächen oder Menüs Daten eingegeben sowie Ereignisse ausgelöst und Funktionen der Software genutzt werden. Die Daten, die als Ergebnis der vom Benutzer getätigten Aktionen werden im Gegenzug auf der Benutzeroberfläche in geeigneter Form dargestellt.
Die Benutzeroberfläche übernimmt dabei die Kontrolle der Interaktionen zwischen Anwender und System. Sie übermittelt die eingegebenen Daten und Funktionsaufrufe an die Anwendungsschicht und bereitet die zurückgelieferten Daten auf und präsentiert sie dem Benutzer.
Die Anwendungsschicht oder Applikationsschicht beinhaltet die gesamte Funktionalität der Software. Daneben besitzt sie Kenntnisse über alle benötigten Objekte und Prozesse, die in dem System ablaufen.
Diese Schicht fungiert als Vermittler zwischen Präsentations- und Persistenzschicht, da sie Funktionsanfragen sowie Dateneingaben des Benutzers entgegennimmt, diese verarbeitet und gegebenenfalls an die Persistenzschicht weiterleitet. Im Gegenzug ruft sie Daten aus der Persistenzschicht ab und leitet sie nach der Verarbeitung an die Präsentationsschicht weiter.
Die Persistenzschicht übernimmt die Speicherung der in der Software anfallenden Daten. Darüber hinaus sorgt sie dafür, dass gespeicherte Daten geladen werden und auf Anfrage an die Anwendungsschicht geleitet werden.

Entwurfsmuster

In den vorangehenden Beschreibungen wurden einige mögliche Softwarearchitekturen vorgestellt. Für die Erstellungen von objekt-orientierten Softwarearchitekturen existiert ein umfangreiches An-

gebot an sogenannten *Entwurfsmustern*. Solche Muster liefern geeignete Strukturen von Objektklassen, um Probleme bei der Erstellung objektorientierter Software zu lösen. Gamma u. a. (1995) liefern eine ausführliche Dokumentation von verschiedenen Entwurfsmustern und nennen mögliche Kombinationsmöglichkeiten, um unter Verwendung mehrerer geeigneter Muster spezifische Entwurfsprobleme zu lösen.
Die Zusammenstellung der Klassen *Model/View/Controller* (MVC) liefert eine Möglichkeit, wiederverwendbare und veränderbare Benutzerschnittstellen zwischen Software und Anwender zu erstellen (Gamma u. a., 1995, Seite 4 ff.).
Dabei erfolgt eine Einteilung der Anwendung in drei Objekte: Das Objekt *Model* enthält die Kernfunktionalität der Anwendung sowie die darin anfallenden Daten, mittels des Objekts *View* wird die Präsentation von Informationen auf dem Bildschirm realisiert und der *Controller* definiert, wie die Benutzeroberfläche auf Eingaben reagiert.
Durch diese Trennung der Objekte nach ihrer Funktionalität wird die Flexibilität und Wiederverwendbarkeit erhöht.
MVC nutzt vorhandene Entwurfsmuster, wie zum Beispiel *Decorator*, mit dem ein Entwurfsmuster zur Erstellung von Scrollleisten in einer Ansicht (*view*) realisiert werden kann.

Eine anderes Muster, das bei Systemen angewendet werden kann, deren Größe eine Zerlegung in Aufgabenaspekte erforderlich macht, ist nach Buschmann u. a. (1998) (Seite 32 ff.) das *Layers*-Muster. Mit diesem Muster können komplexe Anwendungen strukturiert und in Gruppen von kooperierenden Teilaufgaben zerlegt werden.

3.2 Die Architektur des Lernmoduls

Beim Entwurf der Software wurde eine funktionale Unterteilung in die Module Benutzeroberfläche, Funktionalität und Datenspeicher vorgenommen. Dadurch wurden die Belange der einzelnen Anwendungsaspekte strikt voneinander abgetrennt und durch entsprechende Schnittstellenfunktionen der Datenaustausch zwischen den Modulen realisiert.
Der modulare Aufbau ermöglicht eine einfache Wartung. Durch die lose Kopplung der Komponenten und die strikte Trennung der Belange ist außerdem ein flexibles Anpassen an veränderte Bedingungen sowie das Erweitern um neue Funktionalitäten möglich.

Im Folgenden wird der Aufbau der Software sowie die einzelnen Komponenten, deren Aufgaben und Interaktionen untereinander überblicksartig dargestellt.
Zunächst wird anhand einer Grafik ein Überblick über die Architektur der Lernumgebung gegeben. Anschließend werden die Komponenten, die die Funktionalität des Programms beinhalten, und deren Interaktionen erklärt.

Abbildung 3.1 veranschaulicht die Architektur der Software und deren modularen Aufbau.

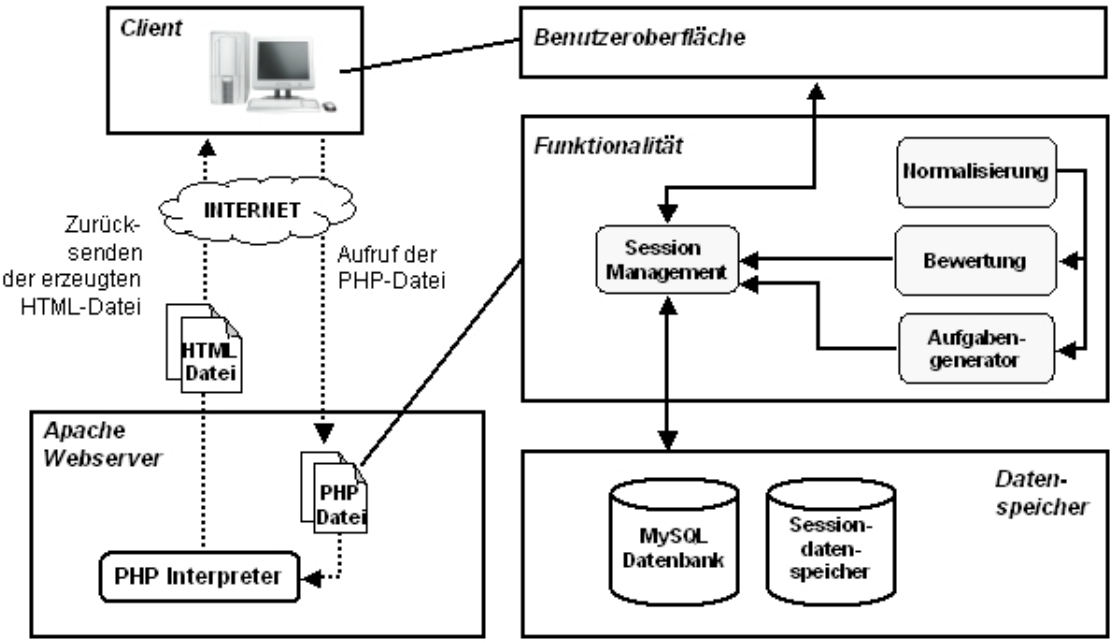

Abbildung 3.1: Softwarearchitektur

Der Zugriff auf das Lernmodul erfolgt über das Internet. Dazu wird über einen Internetbrowser des Client-Computers die Startseite *index.php* aufgerufen, wodurch der Anwender zur grafischen Benutzeroberfläche gelangt. Die gesamte Funktionalität der Software ist in PHP-Dateien enthalten, die auf dem Apache Webserver abgelegt sind. Abhängig von der Aktion, die der Anwender über die Benutzeroberfläche ausführt, wird auf Funktionen und Methoden zugegriffen und somit die jeweiligen PHP-Dateien aufgerufen.
Der PHP-Interpreter verarbeitet die PHP-Skripte und erzeugt daraus HTML-Dateien, die an den Client zurückgesendet und dort im Internetbrowser dargestellt werden.

BENUTZEROBERFLÄCHE

Die *Benutzeroberfläche* besteht aus statischen und interaktiven Elementen: Textuelle Informationen zum Programm und den Lerninhalten bilden die statischen Elemente. Interaktive Elemente sind Eingabefelder und Schaltflächen, mit denen der Anwender während der Benutzung des Lernmoduls Daten eingeben und Aktionen ausführen kann.
Diese Benutzereingaben werden von der Komponente *Session Management* sowohl im *Sessiondaten-Speicher* abgelegt als auch zur Präsentation aus diesem abgerufen.

FUNKTIONALITÄT

Unter *Funktionalität* sind diejenigen Komponenten zusammengefasst, die die Kernfunktionen der Software bereitstellen.

Die Komponenten *Session Management*, *Normalisierung*, *Bewertung* und *Aufgabengenerator* beinhalten Klassen und Funktionsbibliotheken, die alle benötigten Objekte und Methoden zur Verfügung stellen.

Session Management Die Komponente *Session Management* übernimmt das Speichern und Abrufen von Laufzeitdaten.

Im *Sessiondaten-Speicher* werden die Daten der Komponenten *Bewertung* und *Aufgabengenerator* sowie Benutzereingaben gespeichert. Außerdem werden Zugriffe auf das Programm zu Beginn einer neuen Sitzung in der MySQL-Datenbank protokolliert.

Abgespeicherte Daten werden an die Bewertungskomponente und Benutzeroberfläche geleitet. Somit übernimmt diese Komponente die Verknüpfung der Daten-, Anwendungs- und Präsentationsschicht.

Normalisierung Die Komponente *Normalisierung* beinhaltet Klassen, die die nötigen Objekte und Funktionen zur Normalisierung relationaler Datenbankschemata bereitstellen.

In diesen Klassen sind unter anderem die Algorithmen *Anwendung der RAP-Regeln*, *Schlüssel*, *Basis*, *Lossless Join Property* und *Synthese* implementiert.

Diese Klassen werden dem *Aufgabengenerator* und der *Bewertungskomponente* zur Verfügung gestellt.

Bewertung In der *Bewertungskomponente* ist die Prüfung der Benutzereingaben zu den jeweiligen Aufgabenstellungen realisiert. Diese Eingabedaten werden dabei vom *Session Management* im *Sessiondaten-Speicher* gespeichert. Die *Bewertungskomponente* verarbeitet diese Daten und gibt eine Liste von positiven (bei korrekten Eingaben) oder negativen (bei fehlerhaften Eingaben) Rückmeldungen zur Speicherung an das *Session Management* zurück.

Aufgabengenerator Der *Aufgabengenerator* erzeugt auf Grundlage von vordefinierten Standardaufgaben neue Aufgabenstellungen für die aktuelle Sitzung und gibt diese zur Speicherung an das *Session Management* weiter.

DATENSPEICHER

Der *Datenspeicher* beinhaltet diejenigen Komponenten, die Daten während der Laufzeit aufbewahren.

In der *Datenbank* werden im Unterschied zum *Sessiondaten-Speicher*, in dem die Daten nur während der Laufzeit des Programms gespeichert werden, Daten dauerhaft abgelegt.

MySQL Datenbank Die *MySQL Datenbank* besteht aus der Tabelle *sessionprotocol*. In ihr wird bei

jedem Sitzungsstart, gesteuert von der Komponente *Session Management*, jeder Programmaufruf protokolliert.

Sessiondaten-Speicher Im Sessiondaten-Speicher werden alle Daten abgelegt, die während der Laufzeit des Programms anfallen. Die Komponente *Session Management* übernimmt auch hier das Ablegen und darüber hinaus die Bereitstellung von Daten für andere Komponenten.

Kapitel 4

Softwaredokumentation

Dieses Kapitel beinhaltet die Beschreibung und Erklärung der Komponenten und Klassen des Lernmoduls, die alle nötigen Funktionalitäten zur Verfügung stellen.

4.1 Datenspeicherung

Die in Kapitel 3 vorgestellte Komponente *Datenspeicher* wird in diesem Kapitel beschrieben. Es wird eine genaue Erklärung der verwendeten Datenstrukturen sowohl für die MySQL-Datenbank als auch für den Session-Datenspeicher geliefert.
Die Speicherung und das Abrufen von Laufzeit-Daten sowie die Zugriffs-Protokollierung erledigt die Komponente *Session Management*. Deshalb wird die genaue Vorgehensweise der Speicherung und des Abrufens in Abschnitt 4.5, Seite 43 ff. erklärt.

4.1.1 Die MySQL-Datenbank

In der Datenbank werden die Zugriffe auf das Lernmodul protokolliert. Diese enthält eine Tabelle, in der die *Session-ID*, das *Datum* und die *Uhrzeit* des Aufrufs der Startseite gespeichert.

Die Namen für Datenbank und Tabelle sind:

- **lernmodul:** Name der Datenbank.
- **sessionprotocol:** Name der Tabelle zur Protokollierung.

Die Tabelle enthält folgende Spalten, in denen die angegebenen Daten abgespeichert werden:

- **id:** Ist der Primärschlüssel der Tabelle und eine fortlaufende Nummer, die bei jedem neuen Eintrag automatisch um 1 erhöht wird.
- **sid:** Die PHP-SessionID, die beim erstmaligen Aufrufen der Startseite des Lernmoduls vergeben wird.
- **date:** Das aktuelle Datum.
- **time:** Die aktuelle Uhrzeit.

Listing 4.1 zeigt den MySQL-Quelltext zum Anlegen der Datenbank:

```
1  CREATE DATABASE IF NOT EXISTS 'lernmodul';
2
3  CREATE TABLE IF NOT EXISTS 'sessionprotocol' (
4    'id' int(11) NOT NULL auto_increment,
5    'sid' tinytext default NULL,
6    'date' date default NULL,
7    'time' time default NULL,
8
9    PRIMARY KEY  ('id')
10 );
```

Listing 4.1: Quelltext zum Anlegen der Datenbank

4.1.2 Der Sessiondaten-Speicher

Daten, die während der Laufzeit anfallen, werden im Sessiondaten-Speicher abgelegt.
Die dafür verwendete Datenstruktur ist ein zwei-dimensionaler Array, der auf dem von PHP zur Verfügung gestellten Array `$_SESSION` aufbaut.

Der Array `$_SESSION` beinhaltet als Elemente *sections*, die ebenfalls vom Typ Array sind.
Die einzelnen Elemente einer *section* sind *keys*, die auf die ihnen zugewiesenen *values* verweisen, in denen letztlich die konkreten Daten stecken.

Ein Überblick über die *sections*, die in diesem Array enthalten sind, wird in folgender Auflistung gegeben:

general In dieser *section* werden allgemeine Daten abgelegt, die für alle Komponenten der Software von Bedeutung sind.

rap, keys, minimal, ljp, synthesis Diese *sections* enthalten nur die Daten, die für die jeweiligen Aufgabenteile von Bedeutung sind. Dazu zählen unter anderem die Benutzereingaben zur Aufgabenlösung sowie die während der Bewertung erzeugten Fehlermeldungen.

4.2 Normalisierungsklassen

Mit den Normalisierungsklassen sind die Konzepte und Algorithmen zur Normalisierung relationaler Datenbanken implementiert. In diesem Abschnitt werden diese Klassen beschrieben und deren Funktionen detailliert erklärt.

Im Verzeichnis *\program \FDClasses* befinden sich die entsprechenden PHP-Dateien:

AL.class.php5 Beinhaltet Variablen und Funktionen für Attributlisten. Instanzen dieser Klasse sind Objekte vom Typ *AL*.

FD.class.php5 Beinhaltet Variablen und Funktionen für funktionale Abhängigkeiten. Instanzen dieser Klasse sind Objekte vom Typ *FD*.

FDSet.class.php5 Beinhaltet Variablen und Funktionen für funktionale Abhängigkeitsmengen. Instanzen dieser Klasse sind Objekte vom Typ *FDSet*.

DB.class.php5 Ist eine statische Klasse, die Funktionen zur Berechnung des *Synthese*-Algorithmus bereitstellt.

Das UML-Klassendiagramm in Abbildung 4.1 gibt einen Überblick über diese vier Klassen und ihre Abhängigkeiten.

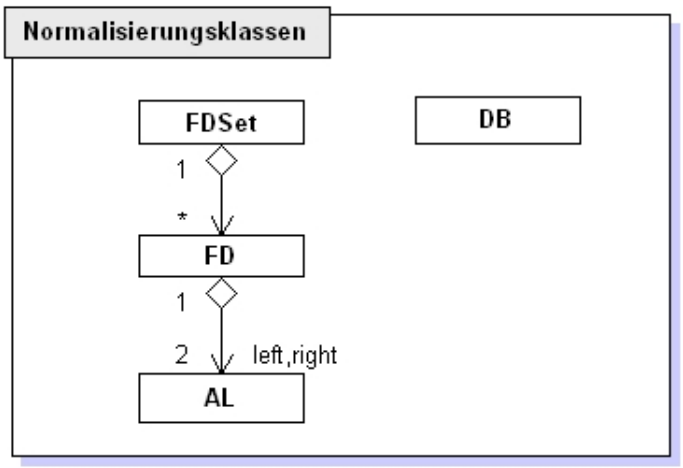

Abbildung 4.1: Klassendiagramm der Normalisierungsklassen

Die Klassen *AL*, *FD* und *FDSet* sind aufgrund ihrer Bedeutung und ihrer Abhängigkeiten untereinander in drei Ebenen angeordnet. Instanzen der Klassen der jeweiligen nächsthöheren Ebene können ohne die Klassen der darunter liegenden Ebenen nicht existieren. In dem Klassendiagramm ist diese Eigenschaft durch Aggregationen dargestellt. Die Kardinalitäten geben an, wie viele Instanzen einer Klasse für eine Instanz der nächst höheren Ebene benötigt werden.
Die Basis bildet die Klasse *AL*, mit der Attributlisten erzeugt werden. Die darin enthaltenen Attribute sind vom PHP-Standardtyp *String*.
Zwei Objekte vom Typ *AL* werden für ein Objekt vom Typ *FD* benötigt, eine linke Seite *left* und eine rechte Seite *right*.
Mehrere Objekte vom Typ *FD* sind wiederum in einem Objekt vom Typ *FDSet* enthalten und bilden eine funktionale Abhängigkeitsmenge.
Die statische Klasse *DB* existiert eigenständig, da sie lediglich die Funktionalität zur Berechnung des *Synthese*-Algorithmus liefert.

In den folgenden Abschnitten werden alle Attribute und Funktionen der einzelnen Klassen vollständig aufgelistet. Die wichtigsten der jeweiligen Funktionen werden anschließend detailliert erklärt.

4.2.1 Die Klasse AL

Die Klasse *AL* repräsentiert Attributlisten. Mit ihr können Instanzen vom Typ *AL* erzeugt, die als Basis-Objekttypen von den Klasen *FD* und *FDSet* benötigt werden.
Einen Überblick über die Attribute und Funkionen der Klasse *AL* liefert das UML-Klassendiagramm in Abbildung 4.2.

VARIABLEN DER KLASSE

private $arrAttributes: Array Array bestehend aus den Attributen der *AL*. Die einzelnen Array-Elemente sind vom Typ *String*. Standardmäßig ist dieser Array leer.

private $arrPermutations: Array Array zur Speicherung der Permutationen. Standardmäßig ist dieser Array leer.

private $permutationsCalculated: Boolean *true*, wenn die Permutationen berechnet wurden, sonst *false*. Standardwert ist *false*.

private $typeName: String Typname 'al' zur internen Typerkennung.

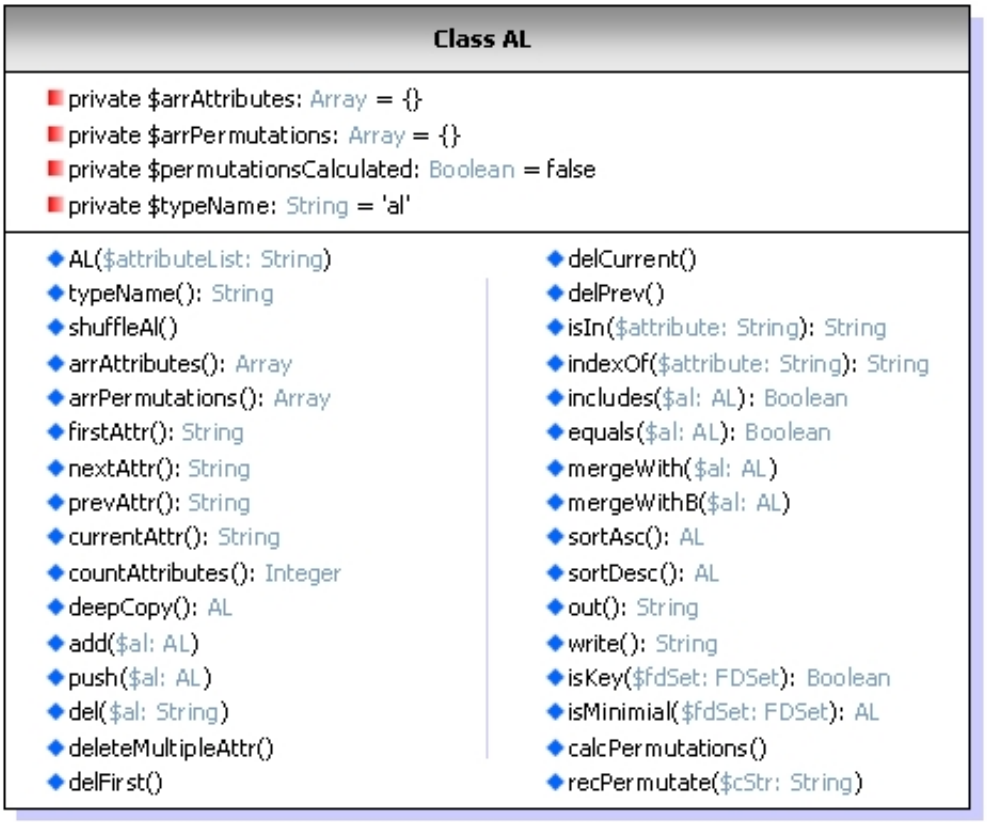

Abbildung 4.2: Klassendiagramm der Klasse AL

FUNKTIONEN DER KLASSE

function AL($strAttr: String) Konstruktor der Klasse *AL*. Benutzt wird diese Funktion, um eine neue Instanz der Klasse *AL* anzulegen, die eine Attributmenge repräsentiert. Mit dem Parameter `$strAttr: String` werden die Attribute der *AL* übergeben, die in der Variablen `$arrAttributes` abgespeichert werden.

function shuffleAl() Ordnet die Attribute der *AL* in zufälliger Reihenfolge neu an.

function arrAttributes(): Array Liefert den Array `$arrAttributes` zurück.

Da es sich bei `$arrAttributes` um eine statische Variable der Klasse *AL* handelt, wird diese Funktion benötigt, um aus anderen Klassen heraus auf `$arrAttributes` zugreifen zu können.

function arrPermutations(): Array Funktioniert wie `arrAttributes`. Diese Funktion gibt den Array `$arrPermutations` zurück.

function add($al: AL) Fügt der Attributliste *AL* neue Elemente hinzu und sortiert diese anschließend in aufsteigender Reihenfolge. Bereits enthaltene Elemente werden nicht hinzugefügt. Die als Parameter übergebenen Attributliste `$al` enthält die hinzuzufügenden Attribute.

function del($al: AL) Entfernt die Elemente in `$al` aus der *AL*.

function includes($al: AL): Boolean Prüft, ob die *AL* die mit dem Parameter `$al` übergebenen Attributliste enthält.

function equals($al: AL): Boolean Prüft, ob die *AL* mit der Attributliste `$al` identisch ist.

function sortAsc() Sortiert die Attribute der Attributliste *AL* in aufsteigender Reihenfolge.

function out(): String Wandelt die Attributliste in einen *String* um. Einzelne Attribute werden dabei durch Kommas getrennt.

function isKey($fdSet: FDSet): Boolean Prüft, ob die Attributliste ein Schlüssel der funktionalen Abhängigkeitsmenge `$fdSet` ist.

function isMinimial($fdSet: FDSet): AL Prüft, ob die Attributliste ein minimaler Schlüssel der funktionalen Abhängigkeitsmenge `$fdSet` ist.

function calcPermutations() Ruft die Funktion `recPermutate` auf, falls die Permutationen noch nicht berechnet wurden.

private function recPermutate($cStr: String) Berechnet rekursiv alle möglichen Permutationen der Attributliste und speichert sie im Array `$arrPermutations`.

4.2.2 Die Klasse FD

Die Klasse *FD* repräsentiert funktionale Abhängigkeiten. In ihr sind alle nötigen Variablen und Funktionen zum Umgang mit funktionalen Abhängigkeiten enthalten.
Eine funktionale Abhängigkeit setzt sich aus zwei Attributlisten für die linke und rechte Seite zusammen.
Einen Überblick über die Attribute und Funkionen der Klasse *FD* liefert das UML-Klassendiagramm in Abbildung 4.3.

VARIABLEN DER KLASSE

protected $left: AL Attributliste vom Typ *AL*, die die linke Seite einer funktionalen Abhängigkeit repräsentiert.

protected $right: AL Attributliste vom Typ *AL*, die die rechte Seite einer funktionalen Abhängigkeit repräsentiert.

protected $typeName: String Typname 'fd' zur internen Typerkennung.

Abbildung 4.3: Klassendiagramm der Klasse FD

FUNKTIONEN DER KLASSE

function FD($left, $right: String) Konstruktor der Klasse *FD*. Benutzt wird diese Funktion, um eine neue Instanz der Klasse *FD* anzulegen. Die mit den Parametern übergeben Attributlisten für linke und rechte Seite der *FD* werden in den Variablen `$left` und `$right` gespeichert. Die in diesen Attributlisten enthaltenen Attribute werden an `$arrAttributes` übergeben.

function getAttributeList(): AL Gibt die in der *FD* verwendeten Attribute als Attributliste zurück.

function sortAsc() Sortiert die Attributlisten `$left` und `$right` in aufsteigender Reihenfolge.

function left(): AL Gibt die linke Seite `$left` der funktionalen Abhängigkeit zurück.

function right(): AL Gibt die rechte Seite `$right` der funktionalen Abhängigkeit zurück.

function addLeft($al) Fügt der linken Seite der *FD* die Attribute des Parameters `$al` hinzu. Bereits vorhandene Attribute werden nicht hinzugefügt.

function addRight($al) Funktioniert wie `addLeft` für die rechte Seite der *FD*.

function delLeft($al) Entfernt die Attribute in `$al` aus der linken Seite der *FD*.

function delRight($al) Entfernt die Attribute in `$al` aus der rechten Seite der *FD*.

function equals($fd): Boolean Vergleicht die *FD* mit der im Parameter `$fd` übergebenen funktionalen Abhängigkeit.

function out(): String Wandelt die funktionale Abhängigkeit in einen *String* um. Linke und rechte Seite werden durch einen Pfeil, die einzelnen Attribute durch Kommas getrennt.

4.2.3 Die Klasse FDSet

Die Klasse *FDSet* repräsentiert funktionale Abhängigkeitsmengen. Instanzen dieser Klasse bestehen aus mehreren funktionalen Abhängigkeiten. Die bereitgestellte Funktionalität umfasst unter anderem die Implementation der Algorithmen zur Normalisierung.
Einen Überblick über die Attribute und Funkionen der Klasse *FDSet* liefert das UML-Klassendiagramm in Abbildung 4.4.

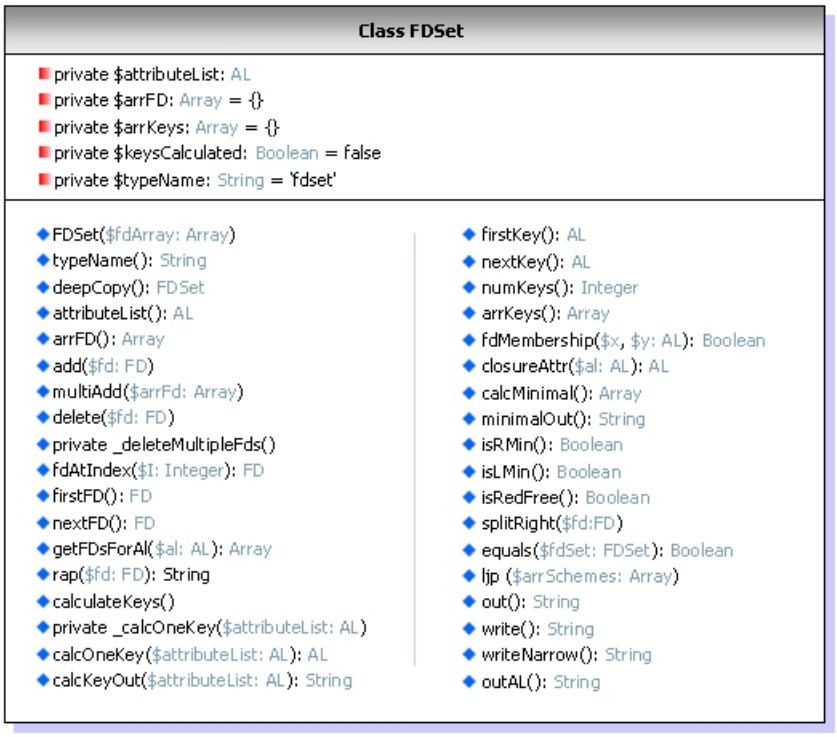

Abbildung 4.4: Klassendiagramm der Klasse FDSet

VARIABLEN DER KLASSE

private $attributeList: AL Attributliste vom Typ *AL*, die alle in der funktionalen Abhängigkeitsmenge *FDSet* benutzten Attribute enthält.

private $arrFD: Array Array, dessen Elemente vom Typ *FD* sind. Enthält alle *FDs* der funktionalen Abhängigkeitsmenge.

private $arrKeys: Array Array, dessen Elemente vom Typ *AL* sind. Enthält alle Schlüssel des *FDSets*.

private $keysCalculated: Boolean Vom Typ Boolean. *True*, falls die Schlüssel berechnet wurden, sonst *false*. Standardwert ist *false*.

private $typeName: String Typname 'fdset' zur internen Typerkennung.

FUNKTIONEN DER KLASSE

function FDSet($fdArray: Array) Konstruktor der Klasse *FDSet*. Mit dieser Funktion werden neue Instanzen der Klasse *FDSet* angelegt. Die einzelnen funktionalen Abhängigkeiten werden mit dem Parameter `$fdArray` übergeben. Dieser Array wird an die Objekt-Variable `$arrFD` übergeben.
Aus den einzelnen funktionalen Abhängigkeiten werden die Attribute extrahiert und im Array `$attributeList` gespeichert.

function attributeList(): AL Gibt die Attributliste des *FDSet* zurück.

function arrFD(): Array Gibt den Array `$arrFD` der funktionalen Abhängigkeiten zurück.

function add($fd: FD) Fügt dem *FDSet* eine neue funktionale Abhängigkeit hinzu.

function delete($fd: FD) Entfernt die im Parameter `$fd` übergebene funktionale Abhängigkeit aus der funktionalen Abhängigkeitsmenge.

function equals($fdSet: FDSet): Boolean Prüft, ob die funktionale Abhängigeitsmenge identisch zu der im Parameter `$fdSet` übergebenen ist und gibt die entsprechenden Werte *true* oder *false* als Ergebnis zurück.

function out(): String Wandelt die funktionale Abhängigkeitsmenge in eine Zeichenkette um. Die einzelnen *FDs* werden durch Semikola getrennt.

function rap($fd: FD): String Gibt die Musterlösung zur Anwendung der *RAP-Regeln* zurück.

function calculateKeys() Berechnet die Schlüssel des FDSets und speichert sie im Array `$keyAttributes`. Dazu werden zunächst die Permutationen der Attributliste des *FDSet* berechnet und für jede dieser Permutationen der Schlüssel berechnet.

function calcOneKey($attributeList: AL): AL Berechnet einen Schlüssel des FDSets und gibt diesen zurück. Der Schlüssel ist dabei abhängig von der in `$attributeList` übergebenen Attributliste.

function fdMembership($x, $y: AL): Boolean Prüft, ob eine funktionale Abhängigkeit mit der linken Seite `$x` und der rechten Seite `$y` in dem *FDSet* gilt.

function closureAttr($al: AL): AL Berechnet die Hülle der im Parameter `$al` übergebenen Attributliste und gibt diese als Ergebnis zurück.

function calcMinimal(): Array Berechnet die minimale Basis der funktionalen Abhängigkeitsmenge. Im zurückgegebenen Array befinden sich die Ergebnisse der Zwischenschritte *Rechtsminimalität herstellen* und *Linksminimalität herstellen* sowie das Endergebnis *minimale Hülle* jeweils Objekte vom Typ *FDSet*.

function ljp ($arrSchemes: Array): Array Wendet den LJP-Algorithmus für die funktionale Abhängigkeitsmenge und die im Parameter `$arrSchemes` übergebenen Zerlegungen an. Als Ergebnis wird ein Array zurückgeliefert, in dem die Zwischenschritte der Berechnung protokolliert sind.

4.2.4 Die Klasse DB

Die statische Klasse *DB* enthält die Implementation des auf Seite erklärten Algorithmus *Synthese*. Mit dieser Klasse werden im Unterschied zu den anderen Normalisierungsklassen keine Objekte erstellt. Sie hat lediglich die Aufgabe, die Funktionen zur Berechnung des *Synthesealgorithmus* zur Verfügung zu stellen.
Das UML-Klassendiagramm in Abbildung 4.5 liefert einen Überblick über die Klasse *DB*.

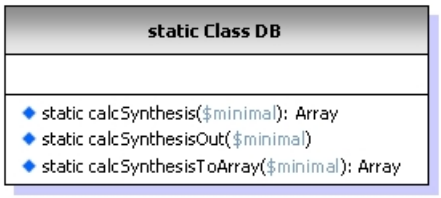

Abbildung 4.5: Klassendiagramm der Klasse DB

FUNKTIONEN DER KLASSE

static function calcSynthesis($minimal: FDSet): Array Berechnet den Synthese-Algorithmus für die funktionale Abhängigkeitsmenge `$minimal` und gibt einen Array bestehend aus den erzeugten Schemata zurück.

static function calcSynthesisToArray($minimal: FDSet): Array Berechnet den Synthese -Algorithmus für die funktionale Abhängigkeitsmenge `$minimal` und gibt in einem Array sowohl die erzeugten Schemata als auch Meldungen, in welchen Schemata Schlüssel enthalten sind, zurück.

4.3 Aufgabengenerator

Die Generierung von Aufgabenstellungen ist implementiert in der Datei *lessons.php5*, die im Verzeichnis *\program \Exercises* abgelegt ist.

Die zugrundeliegende Idee des Aufgabengenerators ist, dass eine funktionale Abhängigkeitsmenge erstellt wird, die ein Relationsschema R repräsentiert, das bei jeder Aufgabenstellung vorgegeben ist. Ergänzend dazu werden für die Aufgabenstellung *RAP-Regeln* mehrere funktionale Abhängigkeiten angegeben, deren Gültigkeit in R geprüft werden soll. Für die *LJP*-Aufgabenstellung werden mehrere Arrays bestehend aus Attributlisten benötigt, die Zerlegungen von R repräsentieren und auf Verlustlosigkeit geprüft werden sollen.

Folgende Objekte werden für die verschiedenen Aufgabentypen benötigt und vom Aufgabengenerator erzeugt:

$fdSet: FDSet Vom Typ *FDSet*. Die funktionale Abhängigkeitsmenge, die das Relationsschema R repräsentiert.

$rapExercises: Array Ein Array bestehend aus funktionalen Abhängigkeiten für die *RAP*-Aufgaben.

$ljpExercises: Array Ein Array, der wiederum aus Arrays mit mehreren Attributlisten besteht, die die Zerlegungsschemata von R repräsentieren.

Generierung der funktionalen Abhängigkeitsmenge

Abbildung 4.6 veranschaulicht die Funktionsweise der Erstellung einer funktionalen Abhängigkeitsmenge.

Ausgangspunkt ist der Attributarray $A = \{a, b, c, d, e, g\}$, dessen Elemente zunächst zufällig angeordnet werden.
Aus dem Attributarray werden unter Angabe eines fest vorgegebenen Indexes *i* einzelne Attribute abgerufen und daraus funktionale Abhängigkeiten erzeugt. Diese werden in einem Array gespeichert und zur Erstellung einer funktionalen Abhängigkeitsmenge verwendet.

Dieses Prinzip wird auch bei der Generierung der zusätzlichen Arrays für *RAP*- und *LJP*-Aufgaben angewendet.

Durch das zufällige Anordnen der Attribute einer vordefinierten Attributmenge können bei der Generierung stets unterschiedliche funktionale Abhängigkeitsmengen erzeugt werden.
Mit der Vorgabe eines Musters wird sichergestellt, dass die funktionalen Abhängigkeitsmengen zu

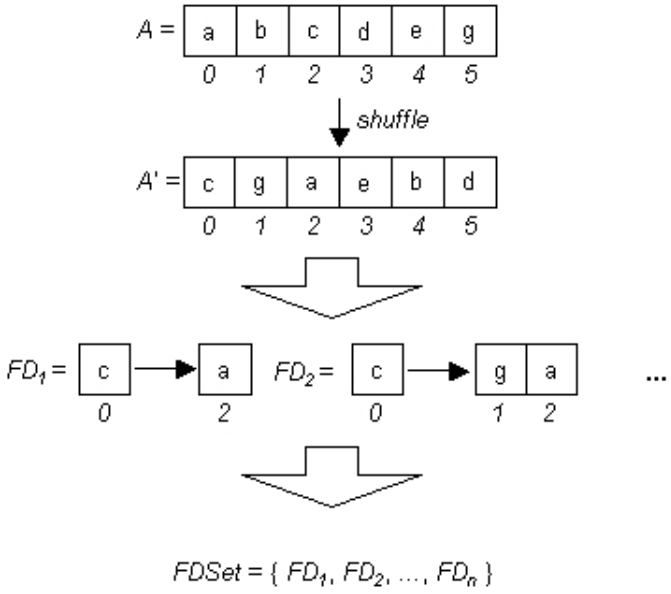

Abbildung 4.6: Generierung einer funktionalen Abhängigkeitsmenge

sinnvollen Aufgabenstellungen führen.

4.4 Bewertungs- und Zuatzfunktionen

Zu jedem Aufgabentyp existiert eine Datei *functions.php5*, in der sowohl die Bewertungsfunktionen als auch Aufgaben-spezifische Zusatzfunktionen implementiert sind.
In den folgenden Abschnitten werden die jeweils wichtigsten Funktionen beschrieben, die zur Bewertung der Lösungen benötigt werden.

4.4.1 Funktionen des Aufgabentyps RAP

Abgelegt ist diese Datei in dem Ordner *program\Exercises\RAP*.

Sie enthält die folgenden Funktionen, die für den Umgang mit den Benutzereingaben und deren Bewertung benötigt werden.

function checkMistakes() Überprüft die Lösungseingaben des Benutzers auf Fehler. Erzeugte Fehlermeldungen werden zum Schluss im Sessiondatenspeicher abgelegt.

function checkValidFd($prevLeft,$prevRight,$currentFd,$currentRight: String): String Überprüft, ob der Benutzer eine FD zur Akkumulation in einem Lösungsschritt richtig verwendet

hat und gibt im negativen Fall Fehlermeldungen zurück.
$prevLeft, $prevRight sind linke und rechte Seite der zu entwickelnden vor der Bearbeitung. *$currentFd* ist die benutzte FD und *$currentRight* die rechte Seite der zu entwickelnden FD nach der Bearbeitung.

function multipleAttributes($al: AL): Boolean Prüft, ob in einer Attributliste *$al* einzelne Attribute mehrfach vorkommen und gibt den entsprechenden booleschen Werten als Ergebnis zurück.

4.4.2 Funktionen des Aufgabentyps Schlüssel

Abgelegt ist diese Datei in dem Ordner *program\Exercises\Keys*. Sie enthält die folgenden Funktionen:

function checkResults() Überprüft die Benutzereingaben auf Fehler. Dazu wird für jeden gespeicherten Lösungsschritt die Funktion `checkMistakes` aufgerufen.

function checkMistakes($oldK,$deletedAttr,$newK: String) Überprüft einen einzelnen Lösungsschritt auf Fehler. Erzeugte Fehlermeldungen werden zum Schluss im Sessiondaten-Speicher abgelegt. *$oldK* und *$newK* sind die Schlüssel vor und nach der Bearbeitung, *$deletedAttr* die gelöschten Attribute.

4.4.3 Funktionen des Aufgabentyps Basis

Abgelegt ist diese Datei in dem Ordner *program\Exercises\Minimal*. Sie enthält die folgenden Funktionen:

function printStep($step,$err: String) Erzeugt HTML-Code für einen Lösungsschritt *$step* und gibt diesen aus. Der Lösungsschritt selbst und die Fehlermeldung *$err* (falls vorhanden) werden dabei in den HTML-Code eingebettet.

function checkMistakes() Überprüft die Lösungsschritte des Benutzers auf Fehler. Erzeugte Fehlermeldungen werden im Sessiondaten-Speicher abgelegt.

4.4.4 Funktionen des Aufgabentyps LJP

Abgelegt ist diese Datei in dem Ordner *program\Exercises\LJP*.

Stellt die nötigen Zusatzfunktionen zur Verarbeitung der Benutzereingaben im Übungsteil, zur Bewertung der vom Benutzer getätigten Lösung, zum Umgang mit den anfallenden Daten und zur Erzeugung von HTML-Code aus den Daten bereit.

Für die Schemamatrix wird ein 2-Dimensionaler Array `$matrix` verwendet.
Jedes Element des Arrays $matrix ist ein Array für eine Zeile der Matrix, jeder dieser Zeilenarrays setzt sich aus den Zellenwerten zusammen.

Die Datei enthält die folgenden Funktionen:

function evalConstrMatrix($matrix): String Überprüft eine erstellte Matrix *$matrix* auf Fehler und gibt diese in einem String zurück. Wird in `printSteps1` verwendet, um die vom Benutzer erstellte Matrix auf Fehler zu überprüfen.

function evalEditMatrix($usedFd,$oldMatrix,$newMatrix,$row1,$row2): String Überprüft, ob eine unter Verwendung einer FD *$usedFd* editierte Matrix *$oldMatrix* richtig bearbeitet wurde und gibt, wenn nötig, die entsprechenden Fehler-Meldungen in einem String zurück. *$newMatrix* ist dabei die Matrix nach der Bearbeitung, *$row1* und *$row2* die Indizes der veränderten Zeilen.

4.4.5 Funktionen des Aufgabentyps Synthese

Abgelegt ist diese Datei in dem Ordner *program\Exercises\Synthesis*. Sie enthält die folgenden Funktionen zur Bewertung und Ausgabe der Benutzereingaben:

function evalScheme($str,$arrPrimeSchemes,$arrPrimeKeyMsgs): String Überprüft, ob ein Schema korrekt erstellt wurde. Dazu wird das in *$str* kodierte Schema mit der Musterlösung in *$arrPrimeSchemes* und *$arrPrimeKeyMsgs* verglichen.

function evalEnd($arrPrimeSchemes,$arrPrimeKeyMsgs): String Überprüft das Ergebnis der Lösung. Vergleicht dazu die Lösung des Benutzers mit einer Musterlösung. Die Lösung des Benutzers wird aus dem Sessiondaten-Speicher abgerufen und mit der Musterlösung in *$arrPrimeSchemes* und *$arrPrimeKeyMsgs* verglichen. Im Falle eines Fehlers wird die entsprechende Meldung als String zurückgegeben.

4.5 Das Session Management

In Kapitel 4.1 wurde bereits der Aufbau der Komponente *Datenspeicherung* sowie die verwendeten Datenstrukturen zu Speicherung der Daten erläutert. In diesem Abschnitt wird nun die Funktionsweise beschrieben, wie die Daten abgelegt und wieder abgerufen werden.

Die statische Klasse *Session* stellt die nötigen Funktionen zur Protokollierung der Programmzugriffe und der Speicherung von Laufzeitdaten zur Verfügung, auf die andere Komponenten zugreifen können. Somit wird durch diese Klasse die Verbindung von Darstellungs- und Datenschicht realisiert.

Folgende Funktionen sind in der Klasse enthalten:

function saveSID() Stellt eine Verbindung zur MySQL-Datenbank her und speichert in der Tabelle *sessionprotocol* bei jedem neuen Programmaufruf die aktuelle SessionID, Datum und Uhrzeit.

function put($section,$key,$value) Legt den Wert *$value* unter der Bezeichnung *$key* in der *$section* des Sessiondaten-Speicher ab.

function addToValue($section,$key,$string) Hängt die Daten in *$string* an bestehende Daten in *$section* und *$key* des Sessiondaten-Speicher an.

function get($section,$key): String Ruft Daten in *$section* und *$key* aus dem Sessiondaten-Speicher ab und gibt diese als String zurück.

function resetSection($section) Setzt die Daten der *section* zurück.

function resetKey($section,$key) Setzt die Daten des *key* zurück.

In Tabelle 4.1 sind die im Sessiondaten-Speicher angelegten *keys* mit einer Beschreibung der *values* aufgelistet. Die *values*, die über die *keys* adressiert werden, sind alle vom Typ String.

Key	**Value**
Section: general	
lessonGenerateDateTime	Datum und Uhrzeit der Aufgabengenerierung
lessonGenerated	1, wenn Aufgabe generiert wurde, sonst 0
lessonAlArray	Die Attributliste, die zur Aufgabengenerierung verwendet wurde
fDSet	Das generierte FDSet
ljpExercises	Die generierten LJP-Aufgabenstellungen
rapExercises	Die generierten RAP-Aufgabenstellungen
minimal	Die berechnete minimale Basis, falls berechnet

minimalExists	1, wenn minimale Basis berechnet wurde, sonst 0
minimalCorrect	1, wenn die Basis korrekt berechnet wurde, sonst 0
Section: rap	
exerciseChoosen	1, wenn eine Aufgabe ausgewählt wurde, sonst 0
exerciseNo	Index der gewählten RAP-Aufgabenstellung
steps	Die Lösungsschritte des Benutzers
result	Das Ergebnis der RAP-Aufgabenstellung
mistakes	Die Fehlermeldungen der Bewertung. Symmetrisch zu result
resultMistake	1, wenn das Ergebnis der Aufgabenberechnung falsch ist, sonst 0
Section: keys	
findKey	1, wenn Benutzer die Schlüsselsuche begonnen hat, sonst 0
steps	Die Lösungsschritte des Benutzers
mistakes	Die Fehlermeldungen der Bewertung
finished	1, wenn Benutzer Berechnung abgeschlossen hat, sonst 0
Section: minimal	
calcMin	1, wenn Benutzer die Bearbeitung begonnen hat, sonst 0
currentStep	Rechtsminimalität, Linksminimalität oder Redundanzfreiheit
currentMode	Zerlegen von FDs, Entfernen von Attributen oder Entfernen von FDs'
currentFDSet	Die momantan berechnete Hülle
finished	1, wenn Benutzer die Bearbeitung abgeschlossen hat, sonst 0
steps	Die Lösungsschritte des Benutzers
mistakes	Die Fehlermeldungen der Bewertung
Section: ljp	
exerciseChoosen	1, wenn eine Aufgabenstellung gewählt wurde, sonst 0
exerciseNo	Index der gewählten Aufgabenstellung
mConstructed	1, wenn Schemamatrix erstellt wurde, sonst 0
ljpCalculated	1, wenn Aufgabenstellung berechnet wurde, sonst 0
editMatrix	1, wenn mit Erstellung der Schemamatrix begonnen wurde, sonst 0
rows	Anzahl der Zeilen der zu erstellenden Schemamatrix
cols	Anzahl der Spalten der zu erstellenden Schemamatrix

steps1	Lösungsschritte der Erstellung der Schemamatrix
constrMatrix	Die vom Benutzer erstellte Schemamatrix
steps2	Die Lösungsschritte der Berechnung der LJP
matrix	Die momentane Schemamatrix während der Berechnung
checkFd	1, wenn eine FD zum Angleichen der Schemamatrix gewählt wurde, sonst 0
chosenFd	Die vom Benutzer ausgewählte FD
ljpCalculated	1, wenn Berechnung der LJP abgeschlossen ist, sonst 0
row1	Index der ersten Zeile, die vom Benutzer bearbeitet werden soll
row2	Index der zweiten Zeile, die vom Benutzer bearbeitet werden soll
Section: synthesis	
mode	1, wenn Benutzer ein neues Schema erstellen möchte. 2, wenn Benutzer die erstellten Schemata auf Schlüsselexistenz überprüfen möchte
nrOfSchemes	Anzahl der erstellten Schemata
nrOfFds	Anzahl der FDs während der Schemaerstellung
fds	Die FDs während der Schemaerstellung
schemes	Die erstellten Schemata
steps	Die Lösungsschritte der Berechnung
finished	1, wenn Benutzer die Berechnung abgeschlossen hat, sonst 0

Tabelle 4.1: Sections und Keys des Sessiondaten-Speichers

4.6 Die grafische Benutzeroberfläche

In diesem Abschnitt wird der Aufbau der Benutzeroberfläche erläutert. Dazu wird beschrieben, wie in den PHP-Skripten HTML-Code erzeugt wird, der in einem Internetbrowser dargestellt wird. Schließlich wird darauf eingegangen, wie die einzelnen PHP-Dateien des Programms zusammengefügt werden, um die grafische Benutzeroberfläche zu erzeugen.

4.6.1 Funktionen zur Generierung des HTML-Codes

Die nötigen Funktionen, mit denen der HTML-Code erzeugt werden kann, sind in der Datei *header.php5* im Verzeichnis *program\GUI* abgelegt.

Die in dieser Datei enthaltenen Funktionen sind in zwei Gruppen unterteilt: Funktionen der ersten Gruppe liefern als Rückgabewert den HTML-Code zurück. Mit den Funktionen der zweiten Gruppe wird der generierte HTML-Code über den PHP-Befehl *echo* direkt in das zu erzeugende HTML-Skript der Webseite eingebunden.
Im Folgenden sind die Funktionen zur Generierung der wichtigsten HTML-Elemente beschrieben. Die meisten der Funktionen haben mindestens einen Parameter namens *$attr* vom Typ String, mit dem HTML-Attribute übergeben werden, um mit Hilfe von *Cascading Style Sheets* spezielle Formatierungen zu verwenden.

function html($t) Dient zur Einbettung von Text in das HTML-Skript. In diesem Text werden Umlaute in die entsprechenden HTML-Zeichenkodierungen umgewandelt.

function beginHtml() Bindet die Dokumenttypdefinition und den HTML-Tag zum Beginn eines HTML-Skripts in die HTML-Datei ein. Diese Funktion muss am Anfang einer PHP-Datei aufgerufen werden.

function endHtml() Schließt die HTML-Datei mit dem HTML-Tag zum Beenden des Skripts. Diese Funktion muss am Ende eines PHP-Skriptes aufgerufen werden.

function HeadTitle($title: String) Bindet den *head*- und den *title*-Block in das HTML-Skript ein. Innerhalb der *title-tags* werden die *Cascading Style Sheets* eingebunden. Der als Parameter übergebene String wird als Titel der Webseite eingebunden.

function beginBody($attr: String) und function endBody() Werden zum Beginnen und zum Beenden eines *body*-Block des HTML-Skriptes verwendet.

function beginDiv($attr: String) und function endDiv() Definieren einen Block, in dem weitere Elemente und Blöcke definiert werden können.

function beginForm($attr: String) und function endForm() Werden zum Beginnen und zum Beenden eines Formulars verwendet.

function beginSelect($attr: String) und function endSelect() Werden zum Beginnen und zum Abschließen eines Blockes verwendet, mit dem eine Dropdown-Auswahlliste definiert werden kann.

function beginTable($attr: String) und function endTable() Definieren Anfang und Ende eines Tabellenblockes.

function beginTr($attr: String) und function endTr() Definieren Anfang und Ende einer Tabellenzeile. Werden innerhalb des Tabellenblockes verwendet.

function td($attr, $str: String) Definiert eine Spalte innerhalb einer Zeile einer Tabelle. Werden innerhalb des Tabellenblockes verwendet.

4.6.2 Einbindung der verschiedenen Programmdateien

Über das PHP-Skript der Startseite werden die einzelnen Dateien aufgerufen und in die Webseite integriert.
Zu Identifikation der Dateien wird in der URL der Name der *section* und des *tab*, die der Benutzer über eine Navigationsschaltfläche aufgerufen hat, mittels eines *query-strings* übergeben. Die URL mit dem *query-string* sieht in allgemeiner Form wie folgt aus:
http://../index.php5?section=<sectionname>&tab=<tabname>
Der *sectionaname* und der *tabname* werden über den Array $_GET, der die im *query-string* übergebenen Werte enthält, abgerufen.

Die möglichen *sectionnames* und die ihnen zugeordnete Bereiche der Benutzeroberfläche sind in Tabelle 4.2 aufgelistet:

Sectionname	**Zugeordneter Bereich**
exercises	Der Bereich des Aufgabengenerators, in dem eine neue Aufgabe generiert werden kann.
rap, keys, minimal, ljp, synthesis	Die einzelnen Aufgabenbereiche.
impressum	Das Impressum zur Webseite.

Tabelle 4.2: Mögliche Namen für Sections

Innerhalb der Aufgabenbereiche geben die *tabnames* an, welcher Abschnitt des Aufgabenbereichs angezeigt wird. Einem *tabname* ist dabei der zugehörige Name der *section* <*sn*> vorangestellt. Tabelle 4.3 zeigt die möglichen *tabnames* und die entsprechenden Aufgabenbereiche.

Tabname	**Zugeordneter Aufgabenbereich**
<sn>index	Die Hauptseite *index.php5* eines Aufgabenbereichs und zugleich der Übungsteil, in dem der Benutzer eine Aufgabe lösen kann.
<sn>eval	Der Bewertungsteil, in dem die Lösung bewertet und zusammen mit Fehlermeldungen auf der Webseite angezeigt wird.
<sn>def	Der Definitionsteil, in dem theoretisches Hintergrundwissen zu den jeweiligen Aufgabenstellungen und anzuwendenden Algorithmen vermittelt wird.
<sn>demo	Der Demoaufgabenteil, in dem mittels eines eingebundenen Videos eine Aufgabe exemplarisch gelöst wird.

Tabelle 4.3: Mögliche Namen für Tabs

Ausgangspunkt ist die Datei *index.php5*. In diese werden zunächst die jeweiligen Dateien eingebunden, die dem in der URL übergebenen *sectionname* zugeordnet sind. In den Aufgabenbereichen

werden dann diejenigen Dateien eingebunden, die dem in der URL übergebenen *tabname* zugeordnet sind.
Als Resultat wird aus dem so zusammengesetzten PHP-Skript eine HTML-Datei erzeugt, die im Webbrowser als grafische Benutzeroberfläche angezeigt wird.

Kapitel 5

Anwendung der Klassen zur Normalisierung

Neben der Verwendung zur Bewertung von Benutzereingaben in der Lernumgebung eignen sich die in Kapitel 4, Abschnitt 4.2 beschriebenen Normalisierungsklassen zum Einsatz im Kontext der Normalisierung relationaler Datenbankschemata.
Die benötigten Algorithmen wurden in den jeweiligen Funktionen der Klassen *AL*, *FD*, *FDSet* und *DB* implementiert.

Für die Anwendung der Funktionen zur Normalisierung wird die in die Benutzeroberfläche integrierte Testumgebung benutzt.
(`http://localhost/LernmodulNormalisierung/index.php5?section=test`)
Sämtliche Daten, die in der Testumgebung verwendet und dargestellt werden, können in der Datei *test.php5* im Verzeichnis *program* editiert werden.

Zunächst ist es erforderlich, dass in dieser Datei die nötigen Objekte erzeugt werden. Benötigt werden folgende Objekte:

- Mehrere Attributlisten vom Typ *AL*, die zu einem Array zusammengefasst die Zerlegungschemata bei der Anwendung des LJP-Algorithmus bilden.
- Mehrere funktionale Abhängigkeiten vom Typ *FD*. Diese werden zur Erstellung einer funktionalen Abhängigkeitsmenge und bei der Anwendung der RAP-Regeln verwendet.
- Eine funktionale Abhängigkeitsmenge vom Typ *FDSet*. Diese Abhängigkeitsmenge repräsentiert das Relationsschema R, auf das die Algorithmen zur Normalisierung angewendet werden sollen.

Listing 5.1 zeigt einen Auszug aus der Datei *test.php5* und veranschaulicht, wie die Konstruktoren zur Erstellung der jeweiligen Objekte aufgerufen werden.

```
1  // Die benötigten ALs:
2  $al1 = new AL('a,b,d');
3  $al2 = new AL('b,d,e,g');
4  $al3 = new AL('a,b,c,d');
5
6  // Die benötigten FDs:
7  $fd1 = new FD('a,d','g');
8  $fd2 = new FD('a,g','c,e');
9  $fd3 = new FD('b,g','d');
10 $fd4 = new FD('c','g');
11 $fd5 = new FD('c,e','g');
12 $fd6 = new FD('g','b,d');
13
14 // Das FDSet, repräsentiert das Relationsschema R:
15 $fdSet = new FDSet(array($fd1,$fd2,$fd3,$fd4,$fd5,$fd6));
16
17 // FD zur Anwendung der RAP-Regeln:
18 $rapFd = new FD('a,d','b,e');
19
20 // Array der Zerlegungen des Relationsschemas für LJP-Berechnung:
21 $ljpSchemes = array($al1,$al2,$al3);
```

Listing 5.1: PHP-Quelltext in test.php5 zur Erstellung der benötigten Objekte

In den folgenden Abschnitten wird erläutert, wie diese Objekte und deren Funktionen zur Normalisierung eingesetzt werden.

5.1 Anwendung der RAP-Regeln

Mit den RAP-Regeln kann geprüft werden, ob eine Attributmenge Y mit einer anderen Attributmenge X in einer funktionalen Abhängigkeitsmenge F hergeleitet werden kann.
Diese Prüfung wird von der Funktion `rap` des Objekts *FDSet* durchgeführt, der als Parameter die oben definierte funktionale Abhängigkeit $ad \rightarrow be$ übergeben wird und die als Ergebnis HTML-Code zur Anzeige der Berechnung zurückliefert. In der Datei *test.php5* müssen dazu die in Listing 5.2 dargestellten Ergänzungen vorgenommen werden.

```
1 // Anzeigen des Relationsschemas R und der FD $rapFd:
2 html('R = ({'.$fdSet->attributeList()->write().'}, {'.$fdSet->write().'})
      ');
3 html('Ist '.$rapFd->write().' herleitbar?');
4 // Funktion rap ausführen und HTML-Code ausgeben:
5 $htmlCode = $fdSet->rap($rapFd);
6 html($htmlCode);
```

Listing 5.2: Aufruf der Funktion rap() in test.php5

Mit dem Aufrufen der Testumgebung wird das PHP-Skript der Datei *test.php5* abgearbeitet und die Funktion `rap` ausgeführt.

Als Ergebnis liegt eine vollständige Anwendung der RAP-Regeln zur Prüfung der Gültigkeit der funktionalen Abhängigkeit vor. Der Screenshot in Abbildung 5.1 zeigt die Ausgabe, die bei der Verarbeitung des PHP-Skriptes erzeugt wurde.

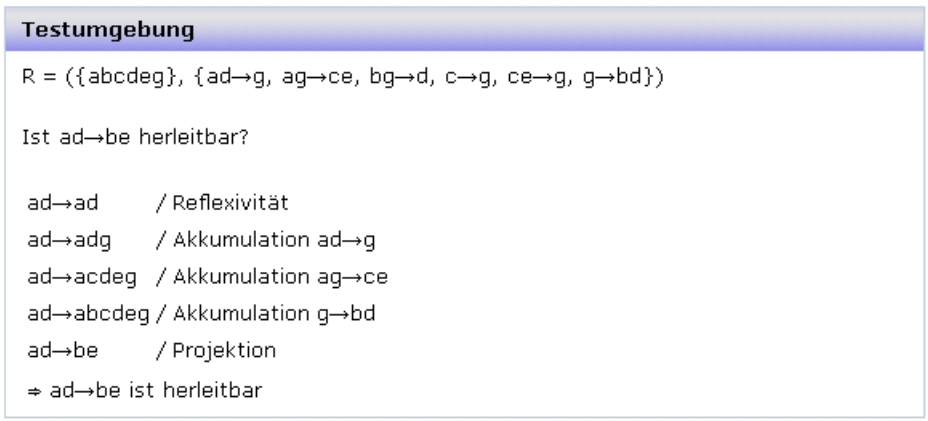

Abbildung 5.1: Ergebnis der Ausführung der Funktion rap()

5.2 Berechnung eines Schlüssels eines Relationsschemas

Zur Berechnung eines Schlüssel wird die Funktion `calcKeyOut` des Objekts *FDSet* verwendet. Dieser Funktion wird als Parameter eine Attributliste übergeben, von der aus der Schlüssel des Relationsschemas R berechnet wird.

In der Datei *test.php5* müssen die in Listing 5.3 dargestellten Programmzeilen ergänzt werden.

```
1 $htmlCode = $fdSet->calcKeyOut($fdSet->attributeList());
2 html($htmlCode);
```

Listing 5.3: Aufruf der Funktion calcKeyOut() in test.php5

Das Aufrufen der Testumgebung führt zu dem in Abbildung 5.2 dargestellten Ergebnis.

Testumgebung

R = ({abcdeg}, {ad→g, ag→ce, bg→d, c→g, ce→g, g→bd})

Berechnung eines Schlüssels:

Beginnen mit K = {abcdeg}

a weggelassen?: $(bcdeg)^+$ = {bcdeg}
a kann nicht gelöscht werden ⇒ K = {abcdeg}

b weggelassen?: $(acdeg)^+$ = {abcdeg}
b kann gelöscht werden ⇒ K = {acdeg}

c weggelassen?: $(adeg)^+$ = {abcdeg}
c kann gelöscht werden ⇒ K = {adeg}

d weggelassen?: $(aeg)^+$ = {abcdeg}
d kann gelöscht werden ⇒ K = {aeg}

e weggelassen?: $(ag)^+$ = {abcdeg}
e kann gelöscht werden ⇒ K = {ag}

g weggelassen?: $(a)^+$ = {a}
g kann nicht gelöscht werden ⇒ K = {ag}

K = {ag}

Abbildung 5.2: Ergebnis der Ausführung der Funktion calcKeyOut()

Als Ergebnis liegt nun ein minimaler Schlüssel K der funktionalen Abhängigkeitsmenge F vor.

5.3 Berechnung der Basis eines Relationsschemas

Das Berechnen der Basis eines Relationsschemas R übernimmt die Funktion `minimalOut`, die als Rückgabewert HTML-Code zum Anzeigen der Berechnung liefert.
Die in Listing 5.4 dargestellten Ergänzungen müssen in der Datei *test.php5* vorgenommen werden.

```
1 $htmlCode = $fdSet->minimalOut();
2 html($htmlCode);
```

Listing 5.4: Aufruf der Funktion minimalOut() in test.php5

Das Ergebnis des Funktionsaufrufes ist in Abbildung 5.3 abgebildet. Als Resultat liegt die Basis der funktionalen Abhängigkeitsmenge F vor.

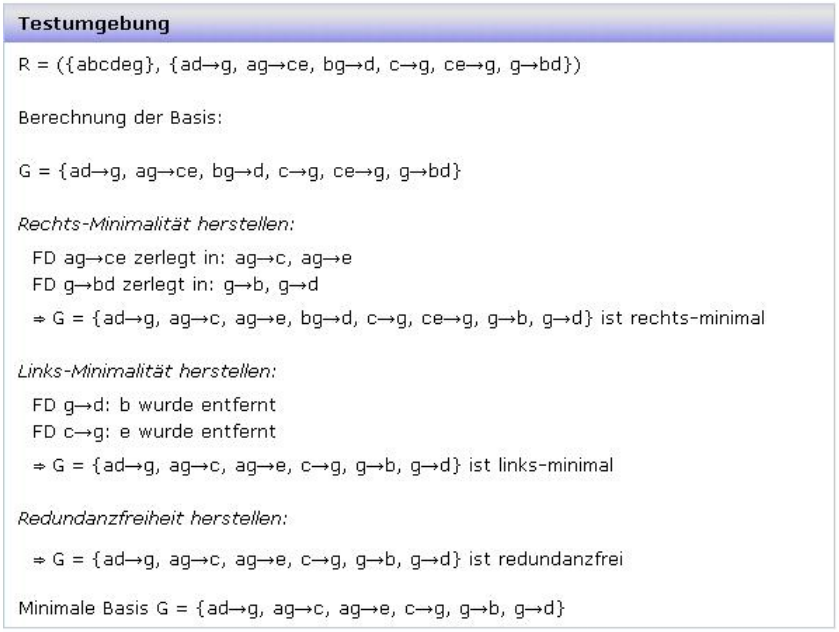

Abbildung 5.3: Ergebnis der Ausführung der Funktion minimalOut()

5.4 Testen der Lossless Join Property

Mit der Funktion `ljp` des Objekts *FDSet* wird überprüft, ob eine Zerlegung des Relationsschemas R verlustlos ist. Dazu muss der Funktion der eingangs definierte Array `$ljpExercise`, der die Attributmengen der Zerlegungsschemata enthält, übergeben werden.
In Listing 5.5 sind die nötigen Ergänzungen in der Datei *test.php5* dargestellt.

```
1  html('Ist␣Zerlegung␣in␣folgende␣Schemata␣verlustlos?');
2  // Anzeige der Zerlegungsschemata:
3  $i=1;
4  foreach($ljpSchemes as $schema) {
5          html('R<sub>'.$i.'</sub>␣=␣{'.$schema->write().'}');
6          $i++;
7  }
8  // Aufruf der Funktion und Ausgabe des Ergebnisses:
9  $htmlCode = $fdSet->ljp($ljpSchemes);
10 html($htmlCode);
```

Listing 5.5: Aufruf der Funktion ljp() in test.php5

Das Ergebnis, wie es in Abbildung 5.4 zu sehen ist, beinhaltet die vollständige Überprüfung der Lossless Join Property für die angegebenen Zerlegungsschemata.

Testumgebung

R = ({abcdeg}, {ad→g, ag→ce, bg→d, c→g, ce→g, g→bd})

Ist Zerlegung in folgende Schemata verlustlos?
R_1 = {abd}
R_2 = {bdeg}
R_3 = {abcd}

Schemamatrix:

	a	b	c	d	e	g
w_1:	a	a	b1	a	b1	b1
w_2:	b2	a	b2	a	a	a
w_3:	a	a	a	a	b3	b3

Zellen angleichen für FD ad→g in den Zeilen w_1 und w_3

	a	b	c	d	e	g
w_1:	a	a	b1	a	b1	b1
w_2:	b2	a	b2	a	a	a
w_3:	a	a	a	a	b3	b1

Zellen angleichen für FD ag→ce in den Zeilen w_1 und w_3

	a	b	c	d	e	g
w_1:	a	a	a	a	b1	b1
w_2:	b2	a	b2	a	a	a
w_3:	a	a	a	a	b1	b1

Es können keine weiteren Zellen angeglichen werden.

Prüfen, ob in einer Zeile nur *a* enthalten sind...
Ergebnis: LJP ist nicht erfüllt.

Abbildung 5.4: Ergebnis der Ausführung der Funktion ljp()

5.5 Zerlegen eines Relationsschemas mit dem Syntehese-Algorithmus

Die Anwendung des Synthese-Algorithmus zur Zerlegung eines Relationsschemas R erfolgt mit der Funktion `calcSynthesisOut` der Klasse *DB*. Als Parameter wird die minimale Basis der funktionalen Abhängigkeitsmenge von R übergeben, die mit der Funktion `calcMinimal` berechnet wird.
Zum Anwenden des Synthese-Algorithmus wird die Datei *test.php5* um die in Listing 5.6 dargestellten Programmzeilen erweitert.

Das Ergebnis der Berechnung liefert die Zerlegung des Relationsschemas R in die Teilschemata R_1,R_2,R_3 und R_4, wie es in Abbildung 5.5 dargestellt ist.

```
1 // Berechnung der Basis mit calcMinimal():
2 // Das letzte Element des zurückgegebenen Arrays ist die Basis:
3 list($rMinFdSet, $lMinFdSet, $minimal) = $fdSet->calcMinimal();
4
5 html('R = ({'.$fdSet->attributeList()->write().'}, {'.$fdSet->write().'})
      ');
6 html('Zerlegung von R mittels Synthese-Algorithmus:');
7
8 // Berechnung und ausgeben der Lösung:
9 DB::calcSynthesisOut($minimal);
```

Listing 5.6: Aufruf der Funktion calcSynthesisOut() in test.php5

Testumgebung

R = ({abcdeg}, {ad→g, ag→ce, bg→d, c→g, ce→g, g→bd})

Zerlegung von R mittels Synthese-Algorithmus:

R1 = ({adg}, {ad→g, g→d})

R2 = ({aceg}, {ag→c, ag→e, c→g})

R3 = ({cg}, {c→g})

R4 = ({bdg}, {g→b, g→d})

Schlüssel vorhanden in R1

Schlüssel vorhanden in R2

Abbildung 5.5: Ergebnis der Ausführung der Funktion calcSynthesisOut()

Kapitel 6

Zusammenfassung und Ausblick

Das *Lernmodul Normalisierung* ist eine webbasierte Lernumgebung, die im Sinne des *Blended Learning* ein ergänzendes Lehr- und Lernangebot zum Themenkomplex *Normalisierung relationaler Datenbanken* liefert. In erster Linie richtet sich die Lernsoftware demzufolge an Studenten der Universität Koblenz-Landau, die im Laufe ihres Studiums an der Vorlesung *Datenbanken für Informationsmanager* teilnehmen. Den Studenten wird primär die Möglichkeit geboten, ihre Fertigkeiten zur Anwendung der behandelten Algorithmen ergänzend zur Vorlesung anhand interaktiver Übungsaufgaben über eine grafische Benutzeroberfläche zu vertiefen.
Die Aufgabenstellungen werden unter Verwendung eines Aufgabengenerators automatisch erzeugt. Nach dem Lösen einer Aufgabe werden die getätigten Eingaben bewertet und Rückmeldungen sowie Hinweise zur Korrektheit einzelner Lösungsschritte gegeben. Außerdem wird mittels textueller Informationseinheiten und Video-Tutorials sowohl theoretisches Wissen zu den einzelnen Algorithmen als auch praktische Vorführungen zu deren Anwendung geboten.

Die angebotenen Lerninhalte umfassen die Themenbereiche *RAP-Regeln*, *Key-Algorithmus*, *Algorithmus Minimal*, *Berechnung der Lossless Join Property* und *Synthese-Algorithmus*.

Um für die einzelnen Lerninhalte und Themenbereiche geeignete Übungsaufgaben bereitzustellen, wurde ein Aufgabengenerator implementiert, der auf Basis einer zufällig angeordneten Attributmenge V und einer „Schablone“ zur Erstellung funktionaler Abhängigkeiten f_i ein Relationsschema $R = (V, f_1, f_2, ..., f_n)$ generiert, das allen Aufgabenstellungen zu Grunde gelegt wird.
Für die Aufgabenbereiche *RAP-Regeln* und *LJP* werden darüber hinaus weitere Objekte benötigt und ebenfalls vom Aufgabengenerator erstellt: Für den Bereich *RAP-Regeln* ist dies eine Menge funktionaler Abhängigkeiten F, deren Gültigkeit in R geprüft werden muss. Für den Bereich *LJP* werden mehrere Attributmengen U benötigt, die mögliche Zerlegungsschemata des Relationsschemas R repräsentieren und deren Verlustlosigkeit mittels des *LJP-Algorithmus* geprüft werden soll.

Die weitere Funktionalität der Software wurde in einer Vielzahl von Funktionen und Klassen implementiert: Die Normalisierungsklassen stellen die benötigten Objekte und Funktionen, dabei insbesondere die Implementierungen der Algorithmen zur Normalisierung, für Attributlisten, funktionale Abhängigkeiten und funktionale Abhängigkeitsmengen zur Verfügung. Die Bewertung der Lösungen zu den einzelnen Aufgabestellungen sowie das Session Management zur Speicherung von Laufzeitdaten und zur Protokollierung der Programmzugriffe wurden jeweils in gesonderten Komponenten realisiert.
Die Generierung der Benutzeroberfläche mitsamt der Darstellung von Informationen und Laufzeitdaten aus dem Sessiondaten-Speicher erfolgt über weitere PHP-Skripte unter Verwendung von Funktionen, anhand derer die benötigten HTML-Codebausteine zusammen mit den Daten in die Webseiten integriert werden.

Bei der Konzeption der Struktur des Lernangebotes und dem Design der grafischen Benutzeroberfläche wurden lerntheoretische Ansätze und didaktische Prinzipien angewendet:
Innerhalb der offen gestalteten Lernumgebung können einzelne Aufgabenbereiche und Informationseinheiten frei ausgewählt werden. Es besteht dabei keine vorgegebene Sequenzierung des Ablaufes einer Übungssitzung. Je nach Bedarf kann der Lernende das Anwenden der Algorithmen anhand der Übungsaufgaben trainieren und sich ein Feedback zur Lösung einholen, theoretische Informationen in aufbereiteter Form nachlesen oder sich mit Hilfe von Videos die Vorgehensweise beim Lösen einer Aufgabe demonstrieren lassen.
Mit dieser Konzeption wurden grundlegende tutorielle und explorative Ansätze miteinander verknüpft, indem das freie Erschließen der Lernangebote möglich ist und mittels Video-Tutorials und Rückmeldungen zu den gelösten Aufgaben Hinweise und Hilfestellungen angeboten werden.

In der Gesamtkonzeption und Gestaltung liefert das *Lernmodul Normalisierung* eine Lernumgebung, die speziell auf die Anforderungen an ein begleitendes Lernangebot zur Vorlesung *Datenbanken für Informationsmanager* hin gestaltet und realisiert wurde.
Darüber hinaus liefert es einen konzeptionellen und methodischen Ansatz für die Entwicklung von Lernumgebungen verwandter Themenbereiche mit ähnlich strukturierten Inhalten, wie beispielsweise Informatik oder auch Mathematik.

Bei der Betrachtung von Erweiterungs- und weiteren Einsatzmöglichkeiten des entwickelten Lernmoduls sind zunächst technische und inhaltliche Gesichtspunkte zu betrachten:
Um den gesamten Stoff des Themenbereichs Normalisierung relationaler Datenbanken abzudecken, müssen zunächst entsprechende Aufgabenbereiche zur Prüfung eines Relationsschemas auf Vorliegen einer der Normalformen sowie zur Anwendung des Algorithmus *Dekomposition* in das Lernangebot der Software integriert werden. In Ergänzung dazu muss der Aufgabengenerator in seiner Funktionsweise so angepasst beziehungsweise erweitert werden, dass für die Prüfung einer bestimm-

ten Normalform entsprechende Relationsschemas generiert werden können. Damit ein Relationsschema erstellt werden kann, das sich in zweiter, dritter oder einer höheren Normalform befindet, muss der Aufgabengenerator so erweitert werden, dass er die jeweiligen Eigenschaften der Normalformen kennt um diese bei der Generierung zu berücksichtigen. Als Ergebnis sollen Relationsschemas erstellt werden können, bei denen die Eigenschaften einer Normalform sowohl vorliegen können als auch verletzt werden. Um den Aufgabengenerator mit der dafür benötigten künstlichen Intelligenz auszustatten, ist jedoch ein erheblicher Programmieraufwand verbunden, der im Rahmen dieses Projektes nicht realisierbar gewesen wäre.
Eine weitere technische und sinnvolle Erweiterung besteht in der Möglichkeit, eine Speicherfunktion zu implementieren, mit der Lernende ihren aktuellen Trainingsstand speichern und zu einem späteren Zeitpunkt an der gleichen Stelle fortzufahren. Die nötigen Daten dazu liefert der Sessiondaten-Speicher des Lernmoduls. Dessen Inhalt muss mit einer *save* und *load* Funktion in der Datenbank gespeichert und wieder geladen werden. Diese beiden Funktionen müssen es also ermöglichen, die Daten in der Struktur, wie sie im Sessiondaten-Speicher vorliegen, in die Datenbank zu exportieren und aus dieser wieder in die ursprüngliche Struktur zu importieren.

Inhaltlich kann das Lernmodul außerdem mit weiteren Materialsammlungen, Informationen oder auch Video-Tutorials angereichert werden, mit denen der explorative Ansatz weiter ausgebaut werden kann: Durch die umfassende und didaktisch aufbereitete Darstellung des Wissens zur Normalisierung relationaler Datenbanken und einer offenen Struktur mit Verknüpfungen zwischen einzelnen verwandten Themenbereichen kann eine Lernumgebung geschaffen werden, in der sich Lernende entdeckend bewegen und sich selbst Wissen aneignen können (vgl. Kapitel 2.2.3, Seite 12).
Durch eine Darstellung der Inhalte von verschiedenen Standpunkten aus und unter Verwendung praxisnaher Anwendungsbeispiele kann außerdem das Konzept des situierten Lernens realisiert werden, um es den Lerndenden zu ermöglichen, das Gelernte auf ähnliche Anwendungskontexte zu transferieren (siehe Kapitel 2.1.3, Seite 6).
Denkbar wäre beispielsweise die Vorführung der Erstellung einer Datenbank in einem realistischen Anwendungskontext angefangen bei der Modellierung mittels Entity-Relationship-Diagrammen, über die Identifikation funktionaler Abhängigkeiten bis hin zur Zerlegung des Relationsschemas mit Hilfe des Synthese-Algorithmus.
In diesem Zusammenhang ist auch eine Erweiterung des Lernmoduls um Kommunikationsmöglichkeiten wie Chat-Funktionen denkbar, um das kooperative Lernen in Gruppen zu ermöglichen.

Erweiterungsmöglichkeiten ergeben sich auch im Hinblick auf die Bewertungskomponente und die daraus generierten Rückmeldungen der Software, die noch nicht ausgereift genug sind, um didaktisch konstruktive Rückmeldungen zu geben oder um von einer *dialogähnlichen Kommunikation* zwischen Lerner und System zu sprechen (Niegemann, 2001, Seite 119). Um das zu erreichen, müssen die bestehenden Bewertungsverfahren um Analyseverfahren erweitert werden, mit denen es möglich ist, neben der Existenz von Fehlern in den Lösungen auch deren Ursachen zu erkennen

und gegebenenfalls entsprechende Verbesserungsvorschläge zu liefern.

Schließlich besteht die Möglichkeit der Integration dieses Lernmoduls in ein Gesamtsystem an Lehr- und Lernangeboten, beispielsweise eine Integration in eine Lernplattform wie beispielsweise *WebCT* oder *IBT Server* (siehe hierzu: Schulmeister, 2003, Seite 251 ff.). In einem solchen System können nicht nur umfangreiche Materialsammlungen und Vorlesungsunterlagen abgelegt oder Kurse verwaltet und gehalten werden, sondern auch solche interaktiven Übungsprogramme wie das *Lernmodul Normalisierung* eingebunden werden, um Trainingsmöglichkeiten zu einzelnen Lehrinhalten anzubieten.

Literaturverzeichnis

[Balzert 1996] BALZERT, Helmut: *Lehrbuch der Software-Technik: Software-Entwicklung*. Heidelberg, Berlin, Oxford : Spektrum Akademischer Verlag GmbH, 1996

[Bruns und Gajewski 2002] BRUNS, Beate ; GAJEWSKI, Petra: *Multimediales Lernen im Netz - Leitfaden für Entscheider und Planer*. 3., bearb. Auflage. Berlin, Heidelberg, New York : Springer-Verlag, 2002

[Buschmann u. a. 1998] BUSCHMANN, Frank ; MEUNIER, Regine ; ROHNERT, Hans ; SOMMERLAD, Peter ; STAL, Michael: *Pattern-orientierte Software-Architektur: Ein Pattern-System*. Bonn, Paris : Addison Wesley Longman Verlag GmbH, 1998

[Dunkel und Holitschke 2003] DUNKEL, Jürgen ; HOLITSCHKE, Andreas: *Softwarearchitektur für die Praxis*. Berlin, Heidelberg : Springer-Verlag, 2003

[Euler 1992] EULER, Dieter: *Didaktik des computerunterstützten Lernens: Praktische Gestaltung und theoretische Grundlagen*. Bd. 3. Multimediales Lernen in der Berufsbildung. Nürnberg : BW Bildung und Wissen Verlag und Software, 1992

[Gamma u. a. 1995] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, MA, USA : Addison-Wesley Publishing, 1995

[Holzinger 2001] HOLZINGER, Andreas: *Basiswissen Multimedia*. Bd. 2. Lernen: Kognitive Grundlagen multimedialer Informationssysteme. Würzburg : Vogel Fachbuch, 2001

[Kerres 2001] KERRES, Michael: *Multimediale und telemediale Lernumgebungen: Konzeption und Entwicklung*. 2., bearb. Auflage. München, Wien : Oldenbourg Verlag, 2001

[Mandl u. a. 2002] MANDL, Heinz ; GRUBER, Hans ; RENKL, Alexander: *Situiertes Lernen in multimedialen Lernumgebungen*. Kap. 10. In: ISSING, Ludwig (Hrsg.) ; KLIMSA, Paul (Hrsg.): *Information und Lernen mit Multimedia und Internet*. Weinheim : Psychologie Verlags Union, 2002

[Niegemann 2001] NIEGEMANN, Helmut M.: *Neue Lernmedien: Konzipieren, entwickeln, einsetzen*. 1. Auflage. Bern, Göttingen, Toronto, Seattle : Springer-Verlag, 2001

[Ostheimer und Freisler 2006] OSTHEIMER, Bernhard ; FREISLER, Peter: *Konzeption und Realisierung einer WBT-Serie zu datenbankgestützten dynamischen Web Sites mit PHP und MySQL*. Justus-Liebig-Universität Gießen : Hrsg.: Professur BWL - Wirtschaftsinformatik, 2006 (in: Arbeitspapiere WI, Nr. 7/2006)

[Riser u. a. 2002] RISER, Urs ; KEUNEKE, Jürgen ; FREIBICHLER, Hans: *Konzeption und Entwicklung interaktiver Lernprogramme. Kompendium und multimedialer Workshop. Lernen interaktiv*. Berlin, Heidelberg, New York : Springer-Verlag, 2002

[Schulmeister 2001] SCHULMEISTER, Rolf: *Virtuelle Universität - Virtuelles Lernen*. München, Wien : Oldenbourg Verlag, 2001

[Schulmeister 2003] SCHULMEISTER, Rolf: *Lernplattformen für das virtuelle Lernen: Evaluation und Didaktik*. München, Wien : Oldenbourg Verlag, 2003

[Schulmeister 2007] SCHULMEISTER, Rolf: *Grundlagen hypermedialer Lernsysteme: Theorie-Didaktik-Design*. 4., überarb. Auflage. München, Wien : Oldenbourg Verlag, 2007

[Steinmetz 2000] STEINMETZ, Ralf: *Multimedia-Technologie. Grundlagen, Komponenten und Systeme*. 3., überarb. Auflage. Berlin : Springer-Verlag, 2000

[Strittmatter und Niegemann 2000] STRITTMATTER, Peter ; NIEGEMANN, Helmut: *Lehren und lernen mit Medien - Eine Einführung*. Darmstadt : Wissenschaftliche Buchgesellschaft, 2000

[Vogel u. a. 2005] VOGEL, O. ; ARNOLD, I. ; CHUGHTAI, A. ; IHLER, E. ; MEHLIG, U. ; NEUMANN, Th. ; VÖLTER, M. ; ZDUN, U.: *Software-Architektur: Grundlagen - Konzepte - Praxis*. 1. Auflage. München : Elsevier GmbH, 2005