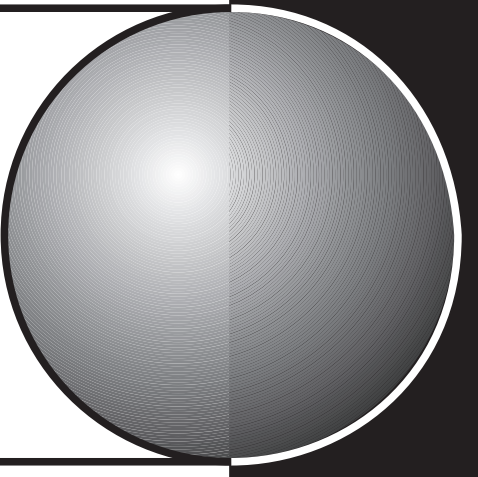
Entwickler-Dokumentation

Datenbank-Umgebung für Netzwerk-Informationssysteme

Dennis

Database Environment for Network Information Systems



28.8.98

Einleitung

Bei der Entwicklung des vorliegenden Systems standen Flexibilität und Plattform-Unabhängigkeit im Vordergrund. Dank des Schichtenaufbaus ist es möglich, Lese- und Schreibzugriff auf eine Datenbank zu bieten, deren Struktur und Format für die oberste Schicht, die Präsentationsschicht, völlig unbekannt ist. Die unteren Schichten (Dokument- und Datenbankserver) haben auch die Kontrolle darüber, in welchem Ablauf das System sich präsentiert und ob Daten anzuzeigen oder einzugeben sind. Auf diese Weise lassen sich höchst unterschiedliche Informationssysteme zur Massen-Individualkommunikation auf demselben einfachen Grundgerüst aufbauen – elektronische Tutorien, elektronische Kataloge, elektronische »Lesezirkel« etc.

Das System läßt sich mit vergleichsweise geringem Aufwand erweitern, ohne die Präsentationsschicht zu verändern – sofern das Arbeitsprinzip bekannt ist. Auf den folgenden Seiten werden daher die grundlegenden Funktionen und Arbeitsschemata erklärt.

Diese Dokumentation richtet sich an Menschen, die mit der Pflege des Systems und der Datenbank beauftragt sind, den Aufbau der Datenbank kennen, Oracle-SQL beherrschen und Grundkenntnisse in HTML und JavaScript besitzen.

Einige Erklärungen werden durch Beispiele verdeutlicht. Dabei gelten die folgenden Schriftkonventionen:

- Wörtlicher Text steht in Fettschrift. Das sind Textstücke, die unverändert abgetippt werden können.
- Variable Ausdrücke stehen in Normalschrift und können, ähnlich wie HTML-Tags, auch zusätzlich in spitzen Klammern »< >« stehen.
- Ellipsen »…« bezeichnen Auslassungen oder Werte ohne aktuellen Belang.

Schematischer Aufbau

In den folgenden Abschnitten wird grob beschrieben, wie das System aufgebaut ist, wie es auf Zugriffe von außen reagiert und welche Daten übertragen werden.

Schichtenaufbau des Systems

Das System läßt sich in drei Schichten einteilen. Die oberste, die Präsentationsschicht, besteht aus einer Mischung aus der Textsprache HTML des World-wide Web (WWW) und der Programmiersprache JavaScript (JS). Sie bildet die Anwender-Schnittstelle. Auf der Anwenderseite setzt diese Schicht zunächst einen Internetzugang und ein JS-taugliches Anzeigeprogramm (Browser) für das WWW voraus. Empfohlen wird der Netscape Navigator ab Version 3.0, da die benutzten JS-Routinen von diesem Browser am besten unterstützt werden.

Die unteren Schichten liegen für Anwender unsichtbar auf dem Dokument- und dem Datenbankserver. Der Datenbankserver arbeitet mit vorgefertigten SQL-Abfragedateien. Sowohl für die Analysedaten als auch für das Interaktionsmodell des Systems wird ein Datenbankschema verwendet.

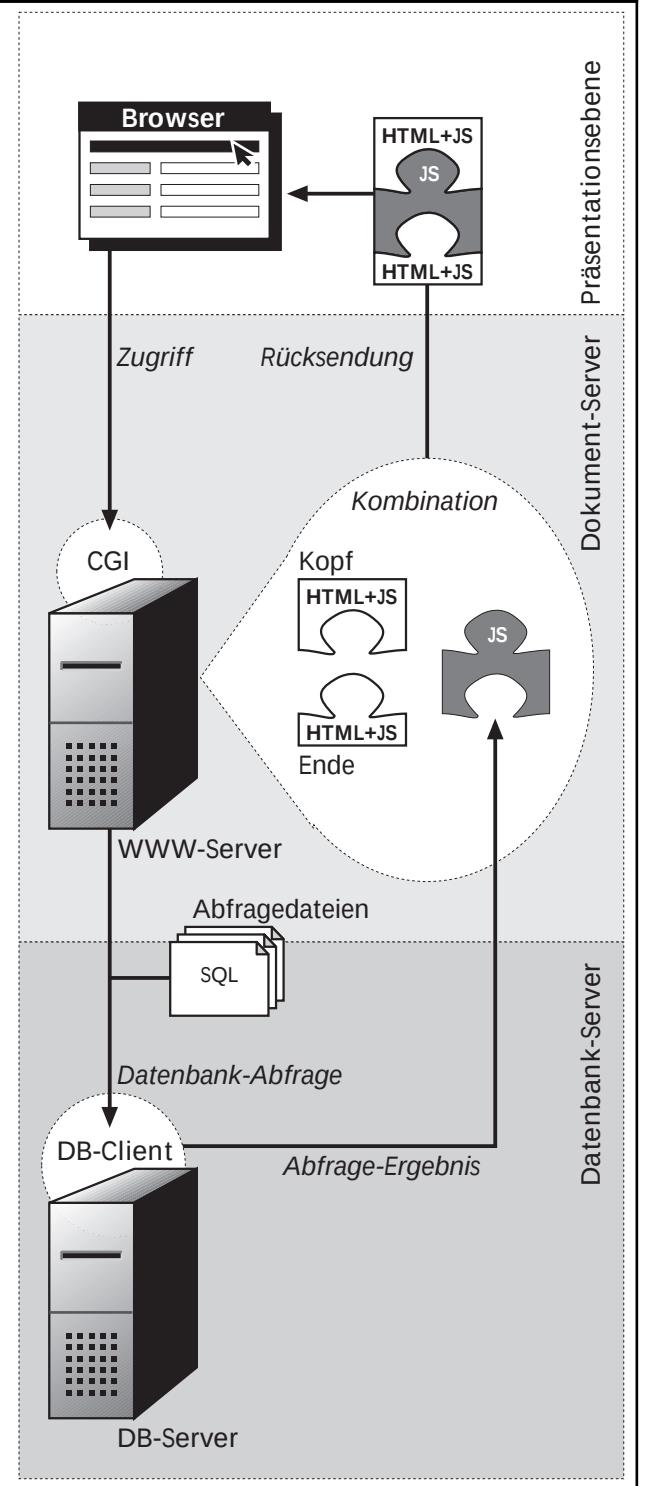
Ablauf beim Zugriff

Der Anwender gibt in seinem Browser die Internet-Startadresse des Systems ein. Zu Beginn wird er sich mit Benutzernamen und Paßwort anmelden müssen. Diese Kennung hat nicht nur den Sinn, die Daten vor unbefugten Zugriffen zu schützen, sondern sie entscheidet auch das individuelle Angebot an Systemfunktionen. Über das Interaktionsmodell wird festgelegt, welche Abläufe welcher Anwender nutzen kann.

Ist die Anwenderkennung erfolgreich abgeschlossen, dann wiederholt sich von nun an eine festgelegte Prozedur:

1. Über die gebotenen Funktionen in der Präsentationsschicht steuert der Anwender ein ganz bestimmtes ausführbares Skript im »Common Gateway Interface« (CGI) des WWW-Servers an.
2. Dies leitet die Aufgabe an ein Filterprogramm weiter, das erstens die Anwenderparameter isoliert, zweitens passend zur Aufgabe eine bestimmte Datenbank-Abfrage startet und drittens im Ergebnis dieser Abfrage die Sonderzeichen, wie z. B. Umlaute, für das WWW konvertiert.
3. Das Skript vom Anfang übernimmt das Ergebnis, verknüpft es mit vorgefertigten Kopf- und Endstücken und schickt es an den Anwender.
4. Schließlich werden alle gerade erzeugten Dateien wieder vom Server gelöscht.

Übersicht: Ablauf beim Zugriff



Aufgaben des CGI-Skripts:
– Filterprogramm starten:
  a) Parameter isolieren
  b) Datenbank-Abfrage
  c) Abfrage-Ergebnis konvertieren
– Kopf, Abfrage-Ergebnis, Ende kombinieren
– Ergebnisdatei zurücksenden
– Temporäre Dateien löschen

Layout der Präsentationsschicht

In erster Linie ergibt sich das Layout aus Konventionen, die aufgestellt wurden, um für eine feste Reihenfolge der Daten und Formularelemente zu sorgen. Daneben zwingen aber auch einige statische Systembestandteile zu einem einheitlichen Erscheinungsbild. Das Layout ist in zwei Frames aufgebaut, der linke für die Steuerung (»Steuerframe«), der rechte als Arbeitsfläche (»Arbeitsframe«). In den folgenden Abschnitten wird das Layout-Schema erklärt.

Obligatorische Elemente

Das erste, was aufgebaut wird, sind die möglichen Folgeaktionen, die als Hyperlinks dargestellt werden. An das Ende der Liste wird stets ein Hyperlink auf die Online-Hilfe eingefügt.

Damit Anwender sich im System nur ein einziges Mal (am Start) identifizieren müssen, werden Benutzername und Paßwort als Formularinhalte »mitgeschleift«. Die drei ersten Formularelemente dazu, »usr«, »pwd« und »abf«, sind auf jeder Datensatzseite enthalten – für den Anwender aber unsichtbar. Ihre Inhalte erhalten sie aus den JS-Variablen »usr_var« bzw. »pwd_var«, »abf« wird erst gefüllt, wenn eine Folgeaktion gewählt wird.

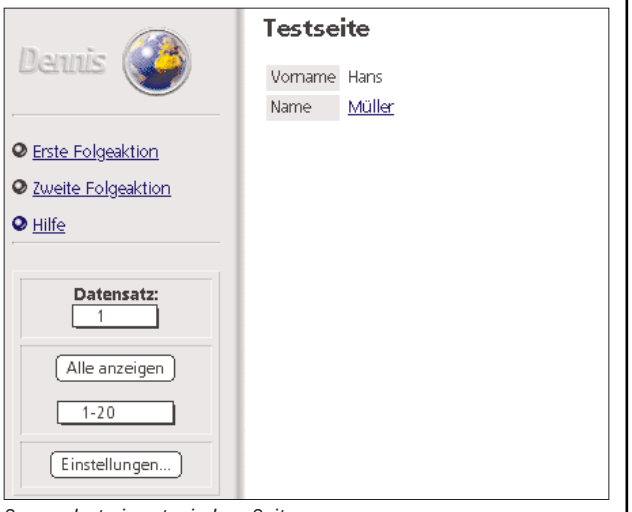
Optionale Elemente

Im Arbeitsframe wird eine zweispaltige Datentabelle eingefügt, falls die vorhergehende Aktion des Anwenders Daten zum Anzeigen aus der Datenbank gelesen hat, falls die aktuelle Aktion für die nächste Aktion Parametereingaben erwartet oder falls die aktuelle Aktion Daten in die Datenbank schreiben soll – die linke Spalte enthält die Attributsbezeichnung, die rechte das Attributsdatum oder ein Textfeld.

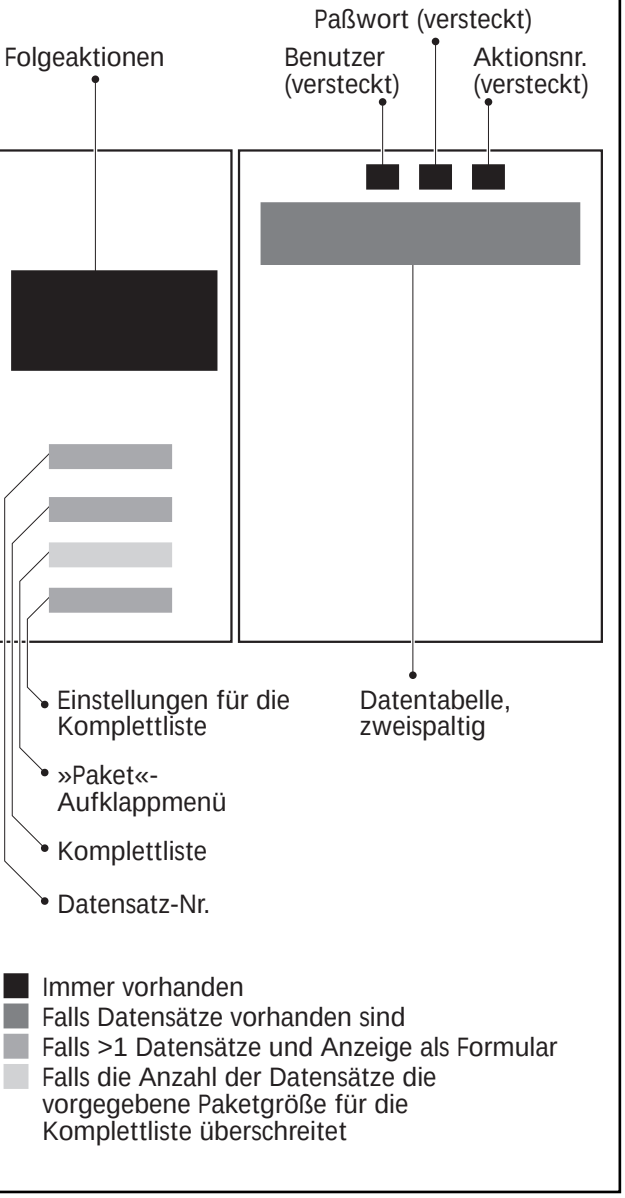
Liegen mehrere Datensätze vor, dann wird unter den Zeilen der Attribute eine Zeile mit Spezialelementen eingefügt. Enthalten sind:

- Ein Aufklappmenü mit Indizes, mit dem der Anwender einen anderen Datensatz für die Anzeige auswählen kann.
- Ein Knopf, der in einem neuen Fenster eine Komplettliste aller Datensätze anzeigt.
- Ein Aufklappmenü für paketweise Komplettlisten bei Überschreitung einer festgelegten Datensatzmenge.
- Ein Knopf, der im Arbeitsframe den Anwender die Attribute auswählen läßt, die in der Komplettliste angezeigt werden sollen.

Übersicht: Seitenlayout



Screenshot einer typischen Seite



Daten an den Server schicken

Mit jeder seiner Aktionen schickt der Anwender dem WWW-Server einige Informationen zu, die Auskunft über seine Rechte, über den erwünschten Folgezustand und eventuell über ausgefüllte Formularfelder geben. Diese Informationen hängt der Browser in einem ganz bestimmten Format an das Ende der Internet-Adresse des CGI-Skripts. Art und Format der Informationen werden in den folgenden Abschnitten erläutert.

Das Sendeformat des Browsers

Wird ein HTML-Formular quittiert, dann werden die Inhalte seiner Elemente, die einen Namen besitzen und ausgefüllt wurden, an die Zieladresse angehängt. Diese Zieladresse zeigt auf das Programm, das die Daten übernehmen und verarbeiten soll. Es wird folgendes Schema verfolgt:

http://zieladresse**?**name**=**inhalt**&**name**=**inhalt …

Ein Fragezeichen trennt Adresse und Parameterangaben. Dann folgen die Formularelemente mit Namen, Gleichheitszeichen und Inhalt, jeweils durch ein Et-Zeichen (»&«) getrennt. Sonder- und Leerzeichen werden in ein Format namens »x-www-form-urlencoded« umgewandelt. Dieses ersetzt Zeichen durch ein Prozentsymbol und ihren ASCII-Wert (nach dem Standard ISO 8859-1) als zweistellige Hexadezimalzahl. Das Leerzeichen beispielsweise hat den dezimalen ASCII-Wert 32 und heißt dann codiert %20.

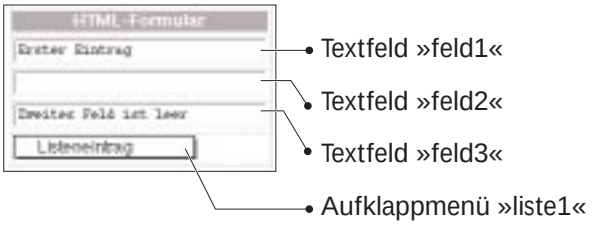
Obligatorische Parameter

Als Konvention sind auf jeder Seite des Arbeitsframes zunächst drei versteckte Formularelemente vorhanden, die den Benutzernamen, das Paßwort und die ID der gewählten Folgeaktion beinhalten. Damit sind die obligatorischen Parameter schon abgedeckt. Alle weiteren Elemente erhalten nur eine Bedeutung, wenn sie über Platzhalter in die SQL-Abfragedateien eingebracht werden. Die drei obligatorischen Elemente heißen »usr«, »pwd« und »abf«. Daher beginnt jeder Parameterstring mit folgenden Parametern:

… ?usr=user**&pwd=**paßwort**&abf=**aktion …

Ein Grund für den Aufbau in Frames war, daß so die Anzeige des Parameterstrings (mit dem Paßwort) im Browser-Fenster unterdrückt werden kann.

Beispiel: Parameterübergabe aus Formularen



Der ausgewählte Eintrag im Aufklappmenü habe den Übergabe-Wert »01«, und die Zieladresse des HTML-Formulars soll lauten: »http://www.bsp.de/cgi-bin/bspskript«

Die Inhalte werden kodiert:
feld1=Erster%20Eintrag
feld3=Zweites%20Feld%20ist%20leer
liste1=01

Zieladresse mit kodierten Parametern:
http://www.bsp.de/cgi-bin/bspskript?feld1=Erster%20Eintrag&feld3=Zweites%20Feld%20ist%20leer&liste1=01

Datenstruktur der Ergebnisdatei

Die fertige Datei, die an den Anwender zurückgesandt wird, hat stets die gleiche Struktur, und ihre Bestandteile variieren nur wenig. Wie statisch ein Großteil der Datei ist, kann schon daran erkannt werden, daß das Ergebnis aus der Datenbank-Abfrage mit einem vorgefertigten Kopf- und Endstück gekoppelt wird. Was variiert sind die Inhalte einiger weniger JS-Variablen. Die wichtigsten davon werden in den folgenden Abschnitten erklärt.

»daten«

Die zweidimensionale Feldvariable »daten[y][x]« ist von zentraler Bedeutung, denn sie bestimmt den Datenteil der Präsentationsschicht. Ihr Inhalt erzeugt die Datentabelle, die auf der vorhergehenden Seite beschrieben wurde. Ist »daten« leer, wird keine Tabelle erzeugt. Die erste Dimension der Variablen entspricht dem Datensatz, die zweite den Attributen. Für einfache Abfragen, die nur ein einziges Ergebnis liefern, oder für Formulare zur Dateneingabe genügt ein einziger Datensatz, also Index 0 der ersten Dimension. Unter Umständen kann ein Attribut von »daten« statt eines Werts sogar eine weitere Feldvariable enthalten, also dreidimensional sein – mehr dazu bei der Erläuterung von »layout« weiter unten. Beispiel:

daten[0][0] = “1. Datensatz, 1. Attribut“
daten[0][1] = “1. Datensatz, 2. Attribut“

»global«

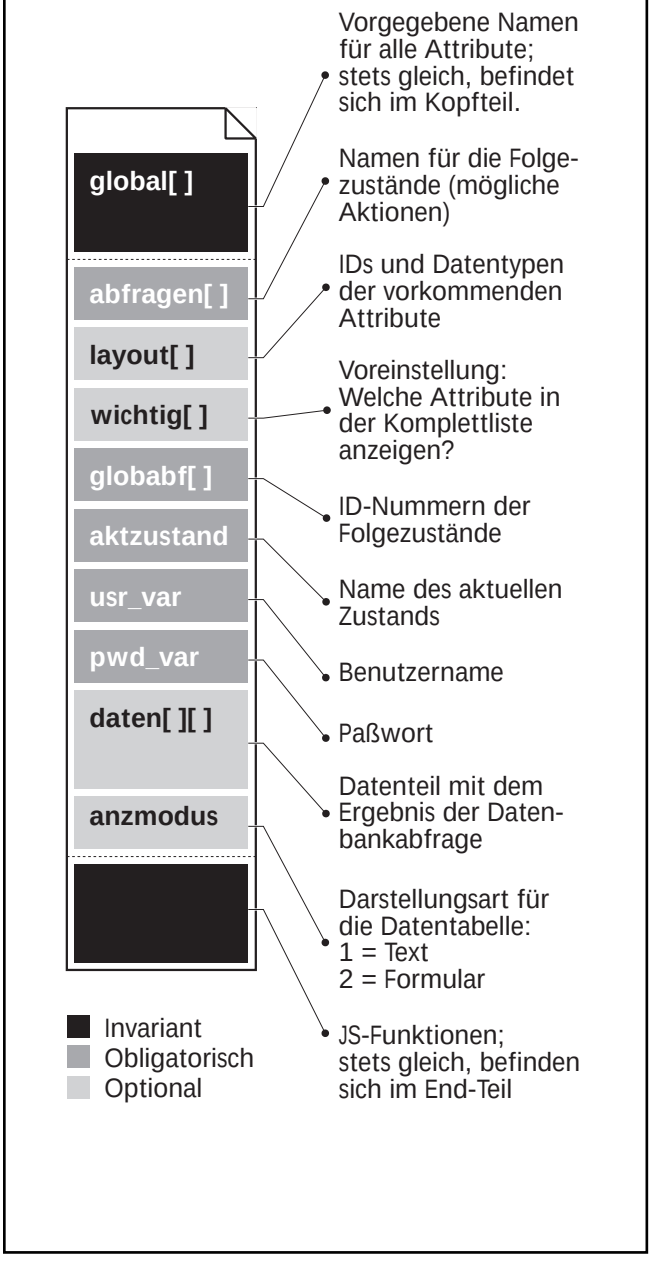
Bei dieser eindimensionalen Feldvariablen handelt es sich um eine reine Auflistung der Namen aller möglichen Attribute. So können Daten später einer sprechenden Bezeichnung zugeordnet werden. Die Variable ist statisch und wird bereits im vorgefertigten Kopfteil der Ergebnisdatei komplett definiert. Beispiel:

global[0] = “Erste Attributsbezeichnung“
global[1] = “Zweite Attributsbezeichnung“

»layout«

Das Bindeglied zwischen »daten« und »global«, das Attributen eine Bezeichnung zuordnet, ist die eindimensionale Feldvariable »layout«. Jedem Attribut stellt sie eine Datenstruktur namens »layout_struktur« bereit, deren Elemente »num« und »typ« erstens die passende Stelle aus der Variablen »global« und zweitens den Datentyp des Attributs angeben. Hat beispielsweise ein Datensatz zwei Attribute, dann muß auch »layout« zwei Strukturen enthalten.

Übersicht: Inhalt der Ergebnisdatei



Derzeit stehen für die Attribute drei unterschiedliche Datentypen bereit, die beeinflussen, wie die Daten im Arbeitsframe dargestellt werden. Außer dem Standardtyp »Text« (Wert 1) gibt es noch die Typen »Liste« (2) und »Referenz« (3). Listen stellen vielwertige Attribute über ein Aufklappmenü dar, das entsprechende Attribut muß in »daten« eine zusätzliche, dritte Dimension enthalten. Referenzen verfügen bei der Formulardarstellung neben dem Textfeld über einen Knopf, der ein neues Browser-Fenster mit dem Inhalt des Textfelds als Internet-Adresse öffnet – bei der Tetdarstellung wird einfach ein Hyperlink eingefügt. Die Datenstruktur von »layout«, vergleichbar mit einem »Record« aus Pascal oder einem »Struct« aus C, ist folgendermaßen definiert:

```
function layout_struktur(num,typ)
{
    this.num = num;
    this.typ = typ;
}
```

Den Elementen »num« und »typ« können schon bei der Initialisierung Werte zugewiesen werden. Beispiel:

```
layout[0] = new layout_struktur(0,1)
```

Entspricht:

```
layout[0] = new layout_struktur()
layout[0].num = 0
layout[0].typ = 1
```

Die Variable »daten« muß an dieser Stelle eine dritte Dimension enthalten. Beispiel:

```
daten[0][0] = new Array("Wert1","Wert2")
```

Anhand aller drei bisher beschriebenen Variablen werden die Daten einer Bedeutung zugeordnet. Das funktioniert über ein »Mapping«, eine maskenartige Abbildung der verschiedenen Werte aufeinander, wie in der Grafik rechts oben gezeigt.

Im Beispiel wird dem ersten Attribut in »layout« die Zahl 0, dem zweiten die 1 und dem dritten die 8 zugeordnet. An den Positionen 0, 1 und 8 von »global« werden die Bezeichnungen abgelesen.

»wichtig«

Der Netscape Navigator 3.0 benötigt relativ viel Rechenzeit für die Darstellung großer Tabellen. Bei der Komplettliste aller Datensätze kann der Bildaufbau gar so langsam werden, daß er einen Programmabsturz vermuten läßt. Verschiedene Mechanismen dienen nun dem Zweck, diesen Effekt zu minimieren.

Beispiel: Zuordnung der Attributsnamen

Globale Attributsnamen:

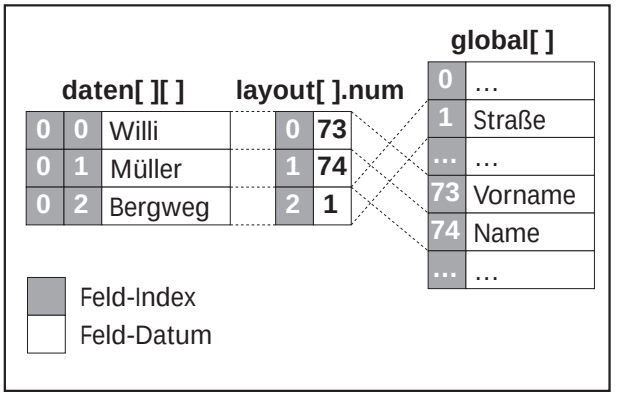
```
…
global[1]="Straße";
…
global[73]="Vorname";
global[74]="Name";
…
```

Datensatz:

```
daten[0][0]="Willi";
daten[0][1]="Müller";
daten[0][2]="Bergweg";
```

Attribut-IDs:

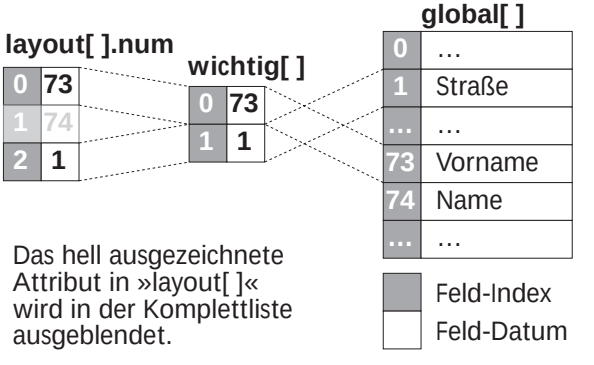
```
layout[0].num=73;
layout[1].num=74;
layout[2].num=1;
```



»daten«
Die erste Dimension bestimmt den Datensatz, die zweite das Attribut.

»layout«
Für jedes Attribut steht ein Eintrag in »layout« zur Verfügung, der die Bedeutung des Attributs bestimmt.

Beispiel: Filterung bei der Komplettliste



Das hell ausgezeichnete Attribut in »layout[ ]« wird in der Komplettliste ausgeblendet.

Um den Bildaufbau zu beschleunigen, werden bei der Komplettliste nicht von vornherein alle Attribute aufgelistet, sondern nur die, deren Nummern in der eindimensionalen Feldvariablen »wichtig« aufgelistet sind. Wie »layout« enthält diese Variable Attributindizes passend zur Variablen »global«, sie setzt aber keine besondere Reihenfolge voraus. Über eines der Spezialelemente bei der Anzeige vieler Datensätze kann der Anwender »wichtig« seinen Interessen anpassen. Beispiel:

wichtig[0] = 1 // Anzeige: Attribut global[1]
wichtig[1] = 0 // Anzeige: Attribut global[0]

»abfragen«

Der Name dieser eindimensionalen Feldvariablen läßt schon auf Datenbank-Abfragen schließen. In ihr befinden sich die Namen aller Folgeaktionen, die dem Anwender auf der aktuellen Seite zur Verfügung stehen. Gefüllt wird die Variable über eine spezielle Abfrage, die sich in jeder SQL-Abfragedatei befinden muß. Welche Aktionen möglich sind, hängt davon ab, an welcher Stelle innerhalb des Systems sich der Anwender gerade befindet (siehe: »Das Interaktionsmodell«). Alle Elemente von »abfragen« werden als Hyperlinks im Steuerframe eingetragen. Der Parameterwert, der an den WWW-Server übermittelt wird, ist jedoch abhängig von »globabf« (siehe unten). Beispiel:

abfragen[0] = “Erste Aktion“
abfragen[1] = “Zweite Aktion“

»globabf«

Ähnlich wie »layout« zu »daten« weist diese eindimensionale Feldvariable jeder Folgeaktion einen Wert zu, der sie eindeutig bestimmt. Im aktuellen System sind es vierstellige Zahlen. Diese Zahlen werden in den Hyperlinks der Folgeaktionen als Werte eingetragen, und beim Anklicken wird die Zahl des ausgewählten Eintrags in das Element »abf« des Arbeitsframes geschrieben, das Formular abgeschickt und somit an das CGI-Skript übergeben. Über den zurückgesandten Wert wiederum kann das Filterprogramm auf dem WWW-Server die passende Abfragedatei bestimmen. Beispiel:

globabf[0] = “0001“
globabf[1] = “0103“

»anzmodus«

Der Inhalt dieser einfachen numerischen Variablen bestimmt, ob Attribute aus »daten« als normaler HTML-Text (1) oder als Inhalt von Formularelementen (2) angezeigt werden sollen. Der Datentyp »Liste« wird in beiden Fällen gleich angezeigt, lediglich »Text« und »Referenz«

ändern sich. Solange der Anwender keine Parameterwerte eintippen sollen, empfiehlt es sich, »anzmodus« auf 1 zu belassen, denn auch bei der Anzeige als Text werden die Daten wieder an den Server verschickt – zu jedem Datensatz-Attribut wird nämlich ein verstecktes Formularelement mit dem Attributswert als Inhalt eingefügt. Beispiel:

anzmodus = 2 // Anzeige als Formular

»aktzustand«

Um dem Anwender jeweils seinen Standort vermitteln zu können, wird der Name des aktuellen Zustands in der Variablen »aktzustand« eingetragen. Damit wird die Überschrift des Arbeitsframes bestimmt. Beispiel:

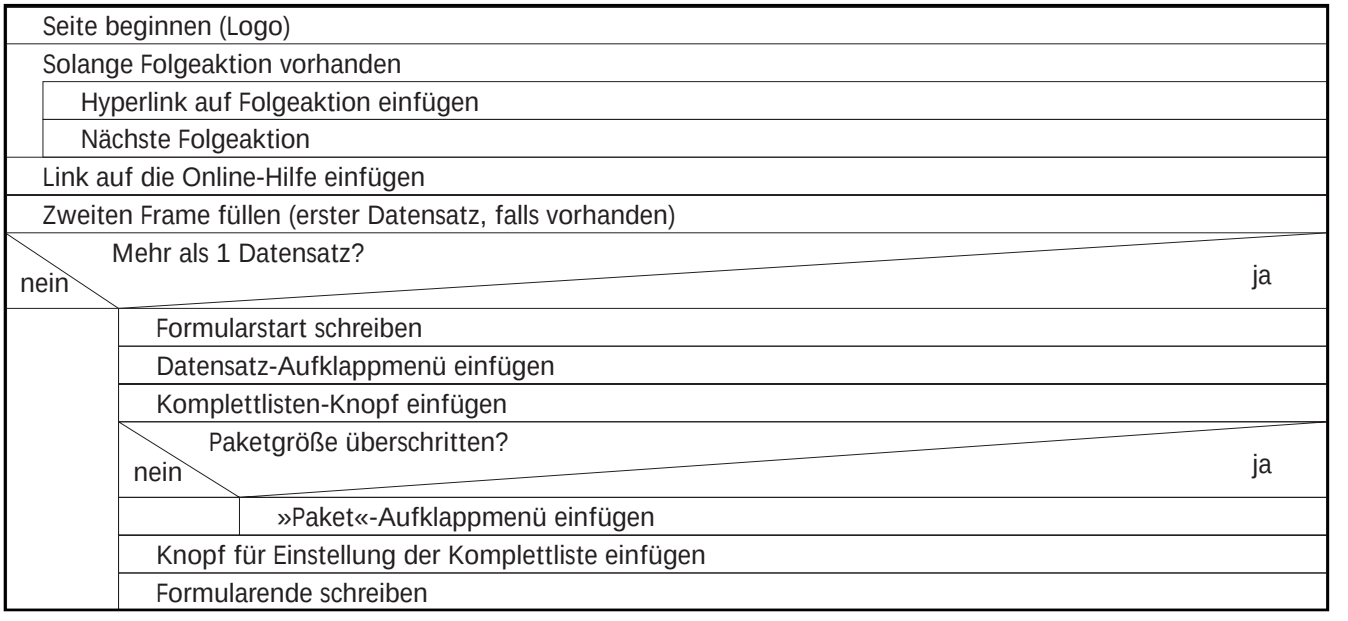
aktzustand = “Einstiegsseite“

Ablauf beim Seitenaufbau

JavaScript wird entweder unmittelbar ausgeführt, während die Seite geladen wird, oder in Form von Prozeduren und Funktionen, ausgelöst durch Ereignisse.

Dennis macht sich beide Varianten zunutze, verwendet für den ersten Seitenaufbau aber größtenteils die unmittelbare Interpretation. Der kleinste Teil des Seitenlayouts entstammt HTML-Code; eingebaute JS-Befehle erzeugen den Großteil »on-the-fly«. Die JS-Prozeduren und -Funktionen hingegen kommen hauptsächlich dann zum Einsatz, wenn mehr als 1 Ergebnis-Datensatz vorliegt. Lediglich die Funktion »einzeln_anzeigen( )« wird auch beim Seitenaufbau eingesetzt, und zwar zum Aufbau des Arbeitsframes.

Das folgende Struktogramm soll den Ablauf beim Seitenaufbau verdeutlichen:



Einstellung der Komplettliste

Bei der Komplettliste aller Datensätze werden nicht alle Attribute von vornherein angezeigt. Der Anwender kann allerdings mit dem Knopf »Einstellungen…« ein Formular im Arbeitsframe anzeigen, das es ihm ermöglicht, die Attribute für die Komplettliste auszuwählen. Die Verarbeitung wird von der Prozedur »einstellung( )« übernommen. In den folgenden Abschnitten wird der interne Ablauf dieser Aktion beschrieben.

Der Ablauf in Worten

Beim Laden der Ergebnisseite wird die boolsche Variable »modus« auf wahr gesetzt. Klickt der Anwender im Steuerframe auf den Knopf »Einstellungen…«, dann wird die JS-Prozedur »einstellung( )« – siehe Struktogramm unten – ausgelöst. Sie überprüft als erstes den Status von »modus«.

Ist »modus« wahr, dann setzt die Prozedur voraus, daß das Einstellungsformular nicht vorhanden ist. Also wird es angezeigt. Die Variable »modus« wird auf unwahr gesetzt. Damit ist die Prozedur für diesen Fall beendet.

Ist »modus« unwahr, dann setzt die Prozedur voraus, daß das Einstellungsformular gerade angezeigt wird. Dann wird das Formular im Einstellungsfenster ausgewertet und als Ergebnis der Feldvariablen »wichtig« zugewiesen. Am Ende wird der erste Datensatz im Arbeitsframe angezeigt.

Das folgende Struktogramm soll den Ablauf der Prozedur »einstellung( )« verdeutlichen:

<table>
<tr><td colspan="2">Modus == wahr?
nein / ja</td></tr>
<tr><td>Einstellungen sammeln</td><td>Frame-Dokument öffnen</td></tr>
<tr><td rowspan="6">Modus = wahr</td><td>Anfang schreiben</td></tr>
<tr><td>Solange Datensatz-Attribute vorhanden sind</td></tr>
<tr><td>Attribut in »wichtig[ ]«?
nein / ja</td></tr>
<tr><td>Auswahlfeld leer | Auswahlfeld markiert</td></tr>
<tr><td>Attributsname schreiben</td></tr>
<tr><td>Nächstes Attribut</td></tr>
<tr><td></td><td>Ende schreiben</td></tr>
<tr><td></td><td>Modus = unwahr</td></tr>
</table>

Sonstige verwendete Funktionen

JavaScript-Funktionen werden nur wenige verwendet. Sie befinden sich allesamt im vorgefertigten Endteil und dienen als Hilfs- und Steuerfunktionen für die Anzeige, wenn viele Datensätze vorliegen. Die folgenden Abschnitte erklären deren Bedeutung und den groben Ablauf. Ausgenommen ist die Funktion »einstellung( )«, die bereits auf der vorhergehenden Seite beschrieben wurde.

»umlaute( )«

Vor dem Abschicken wird die Ergebnisdatei auf dem Dokumentserver in das Zeichenformat von SGML konvertiert. Diese Konvertierung betrifft in erster Linie nationale Sonderzeichen, wie zum Beispiel Umlaute. JavaScript verwendet intern allerdings einen ASCII-Zeichensatz nach dem Standard ISO 8859-1, was bedeutet, daß die SGML-Entities beispielsweise in Formularfeldern uninterpretiert angezeigt werden.

Die Funktion »umlaute( )« hat den Zweck, eine Zeichenkette nach einer vorgegebene Mengen von SGML-Entities zu durchsuchen und diese durch ihre ASCII-Entsprechungen zu ersetzen. ASCII-Sonderzeichen sind in der Datei im »Escape«-Format vorhanden, was im Prinzip dem Format »x-www-form-urlencoded« entspricht. Über die Standardfunktion »unEscape( )« werden die Zeichen schließlich umgewandelt.

»einstell_sammeln( )«

Wenn die Voreinstellungen für die Komplettliste übernommen und in die Variable »wichtig« geschrieben werden sollen, wird diese Funktion den Status der Formularelemente im Arbeitsframe auslesen. Die Länge der Feldvariablen »wichtig« wird dabei nötigenfalls auch verkürzt.

»einzeln_anzeigen( )«

Über ein Aufklappmenü kann der Anwender bestimmen, welcher Datensatz im Arbeitsframe angezeigt werden soll. Die Funktion »einzeln_anzeigen( )« reagiert auf eine Veränderung des Aufklappmenüs.

»alle_anzeigen( )«

Um sich einen Überblick über die erhaltenen Ergebnisse zu verschaffen, kann der Anwender die Komplettliste anzeigen lassen. Aufgerufen wird dabei die Funktion »alle_anzeigen( )«, die ein neues Fenster öffnet und eine größere Anzahl Datensätze in Form einer Tabelle hineinschreibt. Da der Bildaufbau speziell bei größeren Tabel-

Struktogramm: »umlaute( )«

- Liste mit Sonderzeichen aufbauen
- Solange nicht am Ende der Liste
  - Aktuelles Sonderzeichen im Text suchen
  - Solange das Sonderzeichen im Text vorhanden
    - Ergebnis = Teilstring bis Sonderzeichen
    - Ergebnis += konvertiertes Zeichen
    - Ergebnis += Rest des Ursprungstextes
    - Ursprungstext = Ergebnis
    - Das Sonderzeichen erneut suchen
  - Ergebnis = Ursprungstext
  - Nächstes Sonderzeichen in der Liste

Struktogramm: »einstell_sammeln( )«

- Alle Attribute ausgeschaltet?
  - ja: Ersten Datensatz anzeigen
  - nein: Solange Attribute
    - Attribut markiert?
      - nein:
      - ja: In »wichtig[ ]«
    - Nächstes Attribut
    - »wichtig[ ]« zu lang?
      - nein:
      - ja: Verkürzen
    - Ersten Datensatz anzeigen

Struktogramm: »einzeln_anzeigen( )«

- Versteckte Elemente: User, Paßwort, Folgeaktion
- Solange Attribute vorhanden
  - Typ »Referenz« oder »Text«?
    - nein:
    - ja: Text eintragen
  - Solange Listenelement
    - Menülänge reicht?
      - nein:
      - ja: Erweitern+eintragen | Eintragen
    - Nächstes Listenelement
  - Menü zu lang?
    - nein:
    - ja: Verkürzen

len sehr langsam werden kann, wird die Komplettliste bei Überschreitung einer gewissen Datensatzmenge in Pakete unterteilt. Momentan ist die Paketlänge auf 20 eingestellt. Das bedeutet, daß ein zusätzliches Spezialelement (»Paket«-Aufklappmenü) für die Auswahl des Pakets im Hauptfenster eingebaut wird, sobald mehr als 20 Datensätze vorhanden sind. Attribute des Typs »Referenz« werden direkt als Hyperlinks angezeigt.

»referenzieren( )«

Attribute des Typs »Referenz« werden im Hauptfenster nicht nur in einem Textfeld angezeigt, sondern sie erhalten zusätzlich einen Knopf, der ihr Ziel in ein neues Browser-Fenster lädt. Dieser Knopf ruft dabei die Funktion »referenzieren( )« auf. Weil diese sehr einfach aufgebaut ist, wird kein Struktogramm dazu abgebildet.

Struktogramm: »alle_anzeigen( )«

- Startpunkt: erster Datensatz
- Endpunkt: letzter Datensatz
- Paketmenge überschritten?
  - nein:
  - ja:
    - Startpunkt: Paketstart
    - Letzter Datensatz darin?
      - nein:
      - ja:
    - Endpunkt: Paketende
- Fenster öffnen
- Tabellenanfang schreiben
- Solange Attribut vorhanden
  - Attribut anzeigen?
    - nein:
    - ja:
      - Attributsname schreiben
      - Nächste Tabellenspalte
      - Nächstes Attribut
- Nächste Tabellenzeile
- Solange nicht am Endpunkt
  - Solange Attribut vorhanden
    - Attribut anzeigen?
      - nein:
      - ja:
        - Attribut leer?
          - nein:
          - ja:
            - »?« schreiben
        - Typ »Referenz«?
          - nein:
          - ja:
            - Als Hyperlink schreiben
        - Typ »Text«?
          - nein:
          - ja:
            - Als Text schreiben
        - Typ »Liste«?
          - nein:
          - ja:
            - Als Kommaliste schreiben
    - Nächstes Attribut
    - Nächste Tabellenspalte
  - Nächster Datensatz
  - Nächste Tabellenzeile
- Tabelle schließen

Das Interaktionsmodell

Die Anwenderführung innerhalb des Systems wird auf Basis sogenannter »endlicher Automaten« aufgebaut und gesteuert, wobei auch andere Interaktionsmodelle denkbar sind. Endliche Automaten beschreiben Zustände, die das System annehmen kann, und Verknüpfungen (»Links«) zwischen diesen Zuständen. Ein Zustand der Automaten entspricht einer Seite der Präsentationsschicht, ein Link entspricht einer Folgeaktion.

Rechts ist als Beispiel ein einfacher endlicher Automat mit drei Zuständen (Z_0, Z_1 und Z_2) abgebildet. Gestartet wird bei Zustand Z_0. Von dort aus gibt es nur den Link L_0 zu Z_1, und von Z_1 aus gibt es nur den Link L_1 zu Z_2. Zustand Z_2 bietet nun zwei Verzweigemöglichkeiten: Link L_2 führt zurück zu Zustand Z_1 und Link L_3 zum Startzustand Z_0. Es sei angemerkt, daß das realisierte System nicht zwischen Link L_0 und Link L_2 unterscheidet, da beide auf denselben Zustand verweisen.

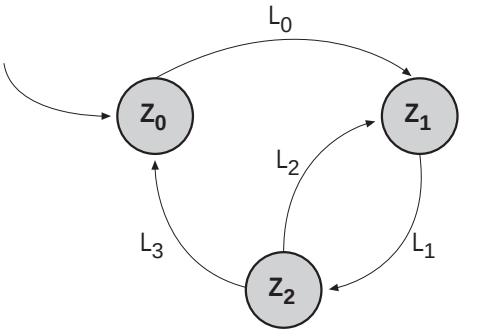
Ein solcher Automat kann in Form einer Tabelle beschrieben werden, in der die Links den Spalten und die Zustände den Zeilen entsprechen.

Automaten können in sich abgeschlossen gebaut, aber auch mit anderen Automaten verbunden werden. Aus diesem Grund wurde für unser System die Konvention aufgestellt, für eine eindeutige Identifizierung einzelner Links oder Zustände die Nummer des Automaten mit der Nummer des Links oder Zustands zu kombinieren.

Da jede Seite der Präsentationsschicht einem Automatenzustand entspricht, wird auch für jede Seite eine ganz bestimmte SQL-Abfragedatei gestartet, in der unter anderem die möglichen Folgezustände ermittelt werden. Gespeichert sind die Folgezustände in einer Datenbanktabelle, die die rechts abgebildete Struktur hat. In der Abbildung ist der Beispiel-Automat von oben bereits umgesetzt, mit zweistelligen Nummern für alle enthaltenen Objekte.

Gelesen würde z. B. die zweite Datenzeile folgendermaßen: Im Automaten 00 hat der Zustand 01 nur einen Link, nämlich 01, der im Automaten 00 auf den Zustand 02 verweist.

Beispiel: Einfacher Automat mit drei Zuständen



Beispiel: Zustandsinformationen in Tabellenform

Automat	Zustand	Link	Folgeautomat	Folgezustand
00	00	00	00	01
00	01	01	00	02
00	02	02	00	01
00	02	03	00	00

Planung für TRDB

Eine der bestehenden Datenbanken, an die das System angepaßt wurde, ist die Praxissemesterstellen- und Studentenverwaltung »TRDB«. Auf den folgenden Seiten werden konkrete Einstellungen und Festlegungen für diese Datenbank aufgelistet.

Der Automat 10

Die Planung begann mit der Sammlung der Funktionen, die dem Anwender geboten werden sollen. Da sich das System jederzeit einfach erweitern läßt, darf die Sammlung nicht als endgültige Festlegung, sondern als vorläufiges Konzept angesehen werden. Alle Funktionen wurden in einen Automaten umgesetzt, dessen momentane Struktur auf der nächsten Seite abgebildet ist.

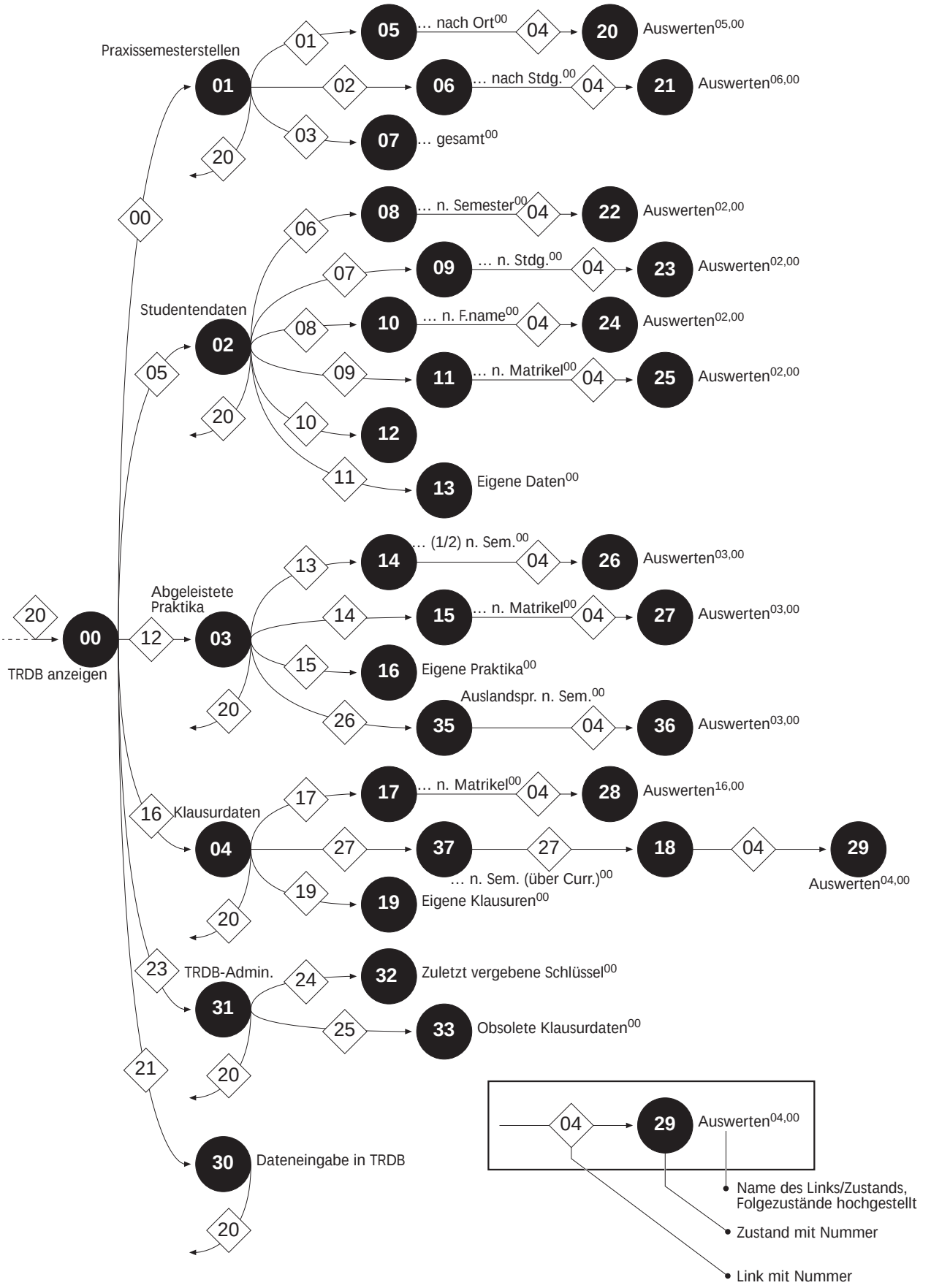
Als Konvention werden für Automaten, Zustände und Links jeweils zweistellige Nummern verwendet, wobei diese Nummern wegen führender Nullen intern als Zeichenketten gehalten werden – was bei der Überführung in JavaScript aber keinen Unterschied bedeutet. Für TRDB wurden die Automatennummern von 10 bis 19 reserviert, benutzt wird bisher nur die 10. Da jedem Automatenzustand eine Seite im Browser und eine spezielle Datenbankabfrage entspricht, wurden in die Dateinamen der Abfragen jeweils Automaten- und Zustandsnummer eingebracht. Konvention:

trdbaazz**.sql**

wobei »aa« den Automaten und »zz« den Zustand definiert.

Alle Zustände werden in der Datenbanktabelle »tut_inf.Zustand« eingetragen. Dort gibt es die Attribute »tab_nr«, »znr«, »snr«, »f_tab« und »f_zust«. Die Bezeichnungen entstammen der Sicht, daß Automaten als Tabellen beschrieben werden können – mit Zuständen als Zeilen und Links als Spalten.

Das Attribut »tab_nr« (Tabellennummer) führt die Nummer des aktuellen Automaten, »znr« (Zeilennummer) die Nummer des aktuellen Zustands und »snr« (Spaltennummer) die des aktuellen Links. Unter dem Attribut »f_tab« (Folgetabelle) wird die Nummer des Automaten eingetragen, auf den der Link zeigt, und »f_zust« gibt den entsprechenden Zustand innerhalb dieses Folgeautomaten an.


TRDB anzeigen
Praxissemesterstellen
Studentendaten
Abgeleistete Praktika
Klausurdaten
TRDB-Admin.
… nach Ort00
… nach Stdg.00
… gesamt00
… n. Semester00
… n. Stdg.00
… n. F.name00
… n. Matrikel00
Eigene Daten00
… (1/2) n. Sem.00
… n. Matrikel00
Eigene Praktika00
Auslandspr. n. Sem.00
… n. Matrikel00
… n. Sem. (über Curr.)00
Eigene Klausuren00
Zuletzt vergebene Schlüssel00
Obsolete Klausurdaten00
Dateneingabe in TRDB
Auswerten05,00
Auswerten06,00
Auswerten02,00
Auswerten02,00
Auswerten02,00
Auswerten02,00
Auswerten03,00
Auswerten03,00
Auswerten03,00
Auswerten16,00
Auswerten04,00
Auswerten04,00
Name des Links/Zustands, Folgezustände hochgestellt
Zustand mit Nummer
Link mit Nummer

Globale Attributsbezeich-nungen

Auf dieser Seite sind alle Attributsbezeichnungen aufgelistet, die für die TRDB-Struktur eingerichtet wurden. Das sind die Einträge der Feldvariablen »global«, über deren Indizes zusammen mit der Feldvariablen »layout« den Ergebnisdaten sprechende Namen zugewiesen werden können. Definiert wird »global« in der Datei »trdb-kopf.htm« – das ist der Kopfteil, an den die Ergebnisse der Datenbankabfragen gehängt werden. Der neueste Stand ist stets dieser Datei zu entnehmen.

```
global[0] = "Adre&szlig;-Nr."        // adrnr
global[1] = "Stra&szlig;e"           // strasse
global[2] = "Hausnr."                // hausnr
global[3] = "Zimmernr."              // znr
global[4] = "Postfach"               // postfach
global[5] = "Staat"                  // staat
global[6] = "PLZ"                    // plz
global[7] = "Ort"                    // wohnort
global[8] = "Telefon"                // telnr
global[9] = "Fax"                    // faxnr
///
/// arbin
global[15] = "Rolle zur FH"          // rolle
global[16] = "Funktion in Firma"     // funktion
///
/// fchprf
global[20] = "Fachpr&uuml;f.-Nr."    // fprfnr
global[21] = "Teilleistungen"        // teillanz
global[22] = "Fachpr&uuml;f.-Art"    // fprfart
global[23] = "Abschlu&szlig;art"     // absart
global[24] = "Fachpr&uuml;f.-Name"   // fchprfn
global[25] = "Gewichtung"            // gwf
///
/// indprf
global[30] = "Datum"                 // datum
global[31] = "Versuche"              // versnr
global[32] = "Note"                  // note
global[33] = "Pr&uuml;fungstyp"      // prftyp
global[34] = "Bemerkung"          // begr,bemerk
global[35] = "Curriculum"            // version
///
/// indprfer
global[40] = "Pr&uuml;fer"           // prfer
///
/// indpss
global[45] = "Praktikum"             // ps
global[46] = "Start"                 // von
global[47] = "Ende"                  // bis
global[48] = "Verg&uuml;tung"        // verg
global[49] = "Sprache"               // sprache
global[50] = "Firmenbescheinigung"   // fbesch
///
/// inst
global[55] = "Institutions-Nr."      // instnr
global[56] = "Name"                  // instname
global[57] = "Abteilung"             // bereich
global[58] = "Branche"               // branche
global[59] = "Mitarbeiter"           // manz
global[60] = "Umsatz"                // umsatz
///
/// instadr
global[65] = "Versandadresse"        // versadr
///
/// person
global[70] = "Personen-Nr."          // personnr
global[71] = "Anrede"                // anrede
global[72] = "Titel"                 // titel
global[73] = "Vorname"               // vname
global[74] = "Name"                  // name
global[75] = "Geburtsdatum"          // gdatum
global[76] = "Geburtsort"            // gebort
///
/// prfanf
global[80] = "Pr&uuml;fungsname"     // prfname
global[81] = "Semesterwochenstunden" // sws
global[82] = "Wiederholbar"          // wdhb
global[83] = "Pr&uuml;fungs-Nr."     // prfanfnr
///
/// psstelle
global[85] = "Stellen-Nr."           // pssnr
global[86] = "Anerkannt"             // anerk
global[87] = "Studenten max."        // maxstud
global[88] = "Zuerst benutzt"        // erstnutz
global[89] = "Zuletzt benutzt"       // letznutz
global[90] = "M&ouml;gliche Dauer"   // potzeitr
global[91] = "Empfohlene Dauer"      // empfzeit
global[92] = "Weitergabe erlaubt"    // weiterg
global[93] = "Praxissemesterordnung" // psovorh
///
/// student
global[95] = "Matrikelnummer"        // matrnr
global[96] = "Semester"              // semester
global[97] = "Studiengang"           // stdgng
global[98] = "Status"                // stand
///
/// vorleist
global[100] = "Vorleistungs-Nr."     // vorleinr
global[101] = "Leistungsname"        // leisname
```

Benutzte Datenbanktabellen

Diese Stelle bietet eine grobe Übersicht, welche Datenbanktabellen alleine für die Systemsteuerung erzeugt wurden und welche Daten ihnen entnommen werden können. Zu bemerken sei, daß in unserem System alle diese Tabellen dem Benutzer »tut_inf« zugeschrieben wurden, um eine Trennlinie zum reinen Datenbestand der TRDB zu ziehen.

Variablen aus der Datenbank

Die einzigen Variablen, die aus der Datenbank heraus gefüllt werden, sind »abfragen«, »aktzustand« und »daten« – alle anderen werden anhand der Hilfstabelle »dual« in den SQL-Abfragedateien selbst erzeugt. Ihre Werte in der Datenbank zu verwalten lohnt sich nicht. Außerdem sind auf diese Weise Inhalt und Bedeutung örtlich verbunden, was die Pflege ein wenig erleichtert. Am Beispiel von »layout« bedeutet das, daß ihr Inhalt direkt in der zentralen Abfrage für »daten« erkannt werden kann.

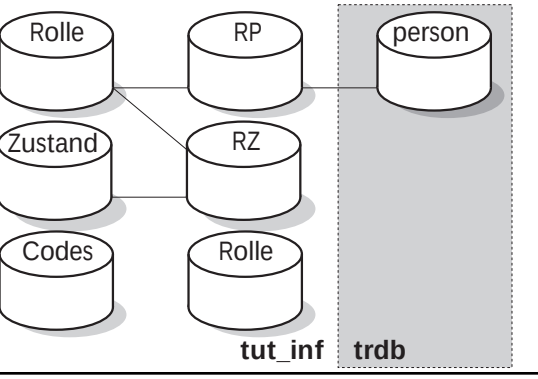
Während die Automaten mit ihren Folgezuständen in der Tabelle »tut_inf.Zustand« verwaltet werden, bekommt »abfragen« die Namen der Folgezustände und »aktzustand« den Namen des jeweils aktuellen Zustands aus »tut_inf.Codes«. Dort gibt es für Zustände und Links ein gemeinsames Attribut »dim_nr«, dessen näherer Bezug über den Wert des Attributs »sorte« entschieden wird.

Kontrolle der Anwenderrechte

Bei der Abfrage der Werte für »abfragen« ist es an vielen Stellen wichtig, den Anwender auf seine Rechte hin zu überprüfen. So lassen sich die Automaten, mit denen die Aktionen und ihre Reihenfolge verwaltet werden, anwenderspezifisch differenzieren. Durchgeführt wird das schließlich über die Kombination der Tabellen »Rolle«, »Zustand«, »RP« (Rolle – Person) und »RZ« (Rolle – Zustand). Jeder Anwender wird einer Rolle zugeordnet, das entspricht einer Anwendergruppe mit besonderen Rechten und einem ganz bestimmten Einstiegszustand.

Ermittelt wird die Anwenderkennung über den anfangs eingegebenen Benutzernamen – dieser besteht aus dem Buchstaben »m« und der Personennummer des jeweiligen Anwenders gemäß Tabelle »trdb.person«. Das bedeutet natürlich, daß nur Personen zugreifen können, die auch im Datenbestand von TRDB enthalten sind.

Übersicht: Tabellen für die Systemsteuerung



Übersicht: Datenstrukturen der Tabellen

Rolle

Rname	Rollenbezeichnung
RNr	Rollen-ID

RP

personnr	Personen-ID aus »trdb.person«
RNr	Rollen-ID

RZ

tab_nr	Einstiegstabelle (=Automat)
RNr	Rollen-ID

Zustand

tab_nr	Tabellennummer (=Automat)
ZNr	Zeilennummer (=Zustand)
SNr	Spaltennummer (=Link)
F_Tab	Folgetabelle
F_Zust	Folgezustand
RNr	Rollen-ID

Codes

tab_nr	Tabellennummer (=Automat)
dim_nr	Dimensionsnummer (Link/Zustand)
sorte	Dimensionssorte*
ztypn	Bezeichnung

* *Zustand: 1*
Link: 2

Verwendete Dateien

Für die Realisierung werden relativ wenig Dateien benötigt, von denen sich einige praktisch nicht verändern. Die Hauptarbeit liegt darin, für jeden neuen Zustand eine entsprechende Abfragedatei zu editieren. Da aber auch hier große Teile des Dateiinhalts unverändert bleiben, beschränkt sich diese Aufgabe darauf, den Inhalt zu kopieren und wenige Schlüsselinformationen zu verändern.

Dateien für die Präsentationsschicht

Nur drei dieser Dateien entsprechen dem gängigen Schema statischer HTML-Seiten, nämlich »index.htm«, »inhalt.htm« und »dummy.htm«. Sie liegen auf dem WWW-Server und bilden den Eingang zu unserem System. Das Layout wird durch zwei Frames bestimmt, die auch schon bei der Login-Seite vorhanden sein müssen. Der Steuerframe wird zunächst mit der Datei »dummy.htm« gefüllt, die nichts anderes als einen weißen Hintergrund enthält. Alle Folgeaktionen werden in den Steuerframe geladen. Vorteilhaft ist, daß bei Seiten mit Frames Netscape die Zieladresse verbirgt.

Dateiname	Zweck
index.htm	HTML-Startdatei (Frame-Definition)
inhalt.htm	HTML-Startseite
dummy.htm	Leere Seite für Steuerframe
trdbkopf.htm	Kopf für Ergebnisdatei
trdbende.htm	Ende für Ergebnisdatei
trdberg.htm	Kombinierte Ergebnisdatei (temporär)

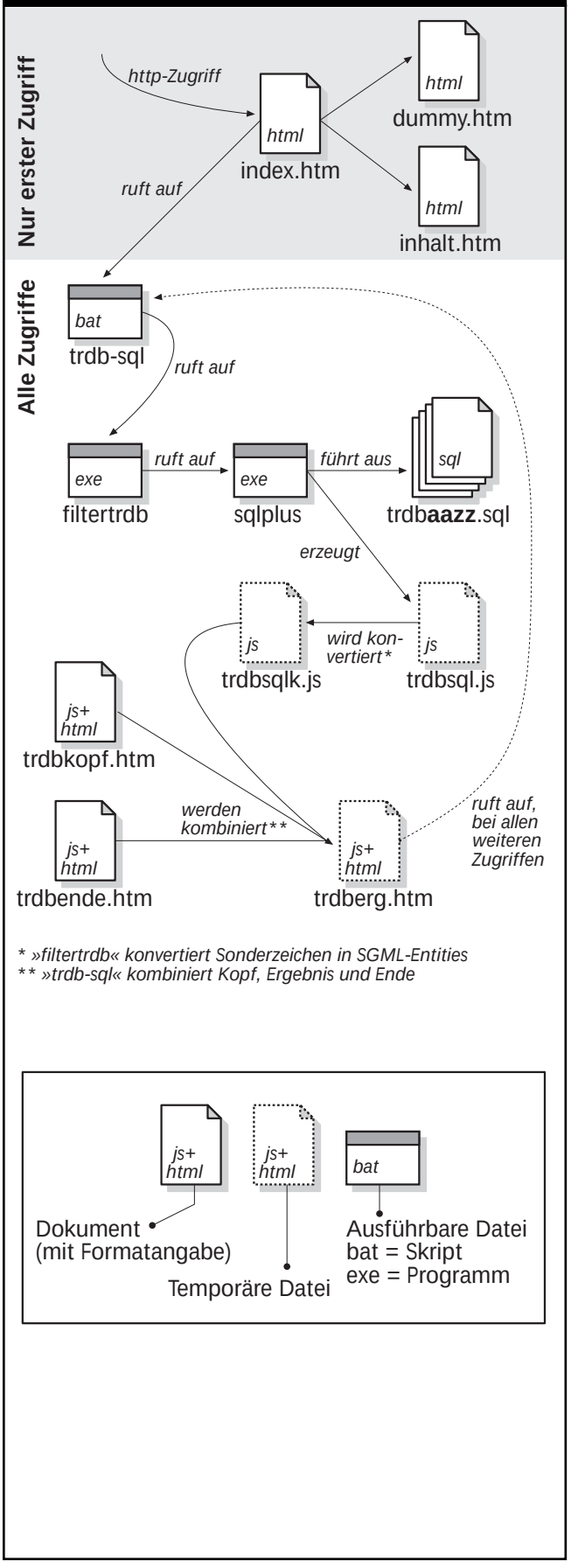
Tabelle 1: Dateien für die Präsentationsschicht

Die Dateien »trdbkopf.htm« und »trdbende.htm« sind keine vollständigen HTML-Dateien. Sie werden mit dem Abfrageergebnis zur Datei »trdberg.htm« kombiniert, die wiederum zum Anwender geschickt und schließlich gelöscht wird.

Dateien für die Steuerschicht

Im CGI-Verzeichnis des WWW-Servers liegt die Skriptdatei »trdb-sql«, die alle Aktionen des Anwenders entgegennimmt. Sie steuert »filtertrdb« an, das in der Programmiersprache »C« geschrieben wurde. Die SQL-

Übersicht: Verwendete Dateien



Abfragedateien liegen in einem Systemverzeichnis der Datenbank. Oracles Datenbank-Client selbst heißt »sqlplus«.

Dateiname	Zweck
filtertrdb	Filter-/Steuerprogramm
trdb-sql	CGI-Skript (UNIX-Skript)
trdb<aazz>.sql	Abfragedatei (z. B. »trdb1000.sql«)
trdberg.js	Abfrageergebnis (temporär)

Tabelle 2: Dateien für die Steuerschicht

Beispiel: Aufruf des CGI-Skripts »trdb-sql«

Internet-Adresse:
http://tr-inf.fh-hannover.de/cgi-bin/trdb.sql

Einbau in das HTML-Formular:
<FORM METHOD="GET" ACTION="http://
tr-inf.tr.fh-hannover.de/cgi-bin/trdb-sql">

Struktur der Abfragedateien

Abfragedateien bestehen vollständig aus Oracle-SQL-Anweisungen und erzeugen JavaScript-Code, der zwischen die vorgefertigten Dateien »trdbkopf.htm« und »trdbende.htm« gehängt und zum Anwender zurückgeschickt wird. Wie der Großteil des Systems variieren die Abfragedateien nur in wenigen Details. In den folgenden Abschnitten werden typische Bestandteile beschrieben, rechts befindet sich ein grafisches Grobschema.

Die Umgebung einrichten

Begonnen wird jede Datei mit einem Satz an SET-Anweisungen, um die Umgebungsbedingungen festzulegen. Wichtig ist zum Beispiel, daß Oracle keine Statusmeldungen zwischen die Ergebnisdaten mischt. Auch wird die Zeilenlänge auf 255 Zeichen beschränkt, da JavaScript längere Zeilen nicht verwerten kann.

Umleitung in eine Datei

Es folgt die Anweisung, mit der die Ergebnisse in eine Datei umgelenkt werden. Diese Datei bekommt stets denselben Namen, sie wird später mit den Kopf- und Endteilen verknüpft und nach dem Abschicken zum Anwender wieder gelöscht.

Standardbenutzer als Generalschlüssel

Nun kann der CONNECT-Befehl eingesetzt werden, um in einen anderen Benutzermodus zu wechseln. Das hat den Vorteil, daß die individuellen Anwender nur das Recht benötigen, sich mit der Datenbank zu verbinden. Alle weiteren Zugriffsrechte können über einen einzigen Standardbenutzer geregelt werden. Auf diese Weise läßt sich der Verwaltungsaufwand verringern. Für TRDB sieht das Schema folgendermaßen aus:

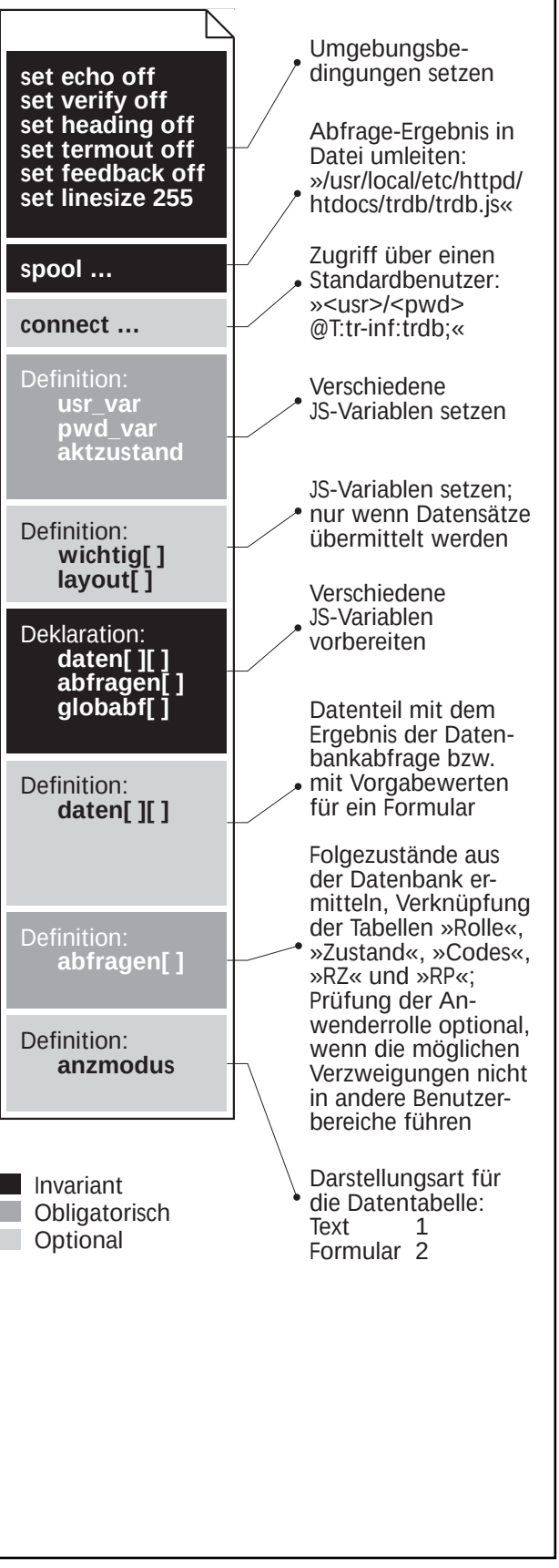
CONNECT benutzer**/**paßwort **@T:tr-inf:trdb;**

Die Platzhalter »benutzer« und »paßwort« müssen durch den Namen des Standardbenutzers und sein Paßwort ersetzt werden.

Anwenderkennung mit Gedächtnis

Auch die nächsten zwei Anweisungen haben mit Benutzername und Paßwort zu tun. Um die Kennung des Anwenders für die Dauer aller seiner Zugriffe »mitzuschleifen«, werden beide Werte in jeder Abfragedatei in zwei Variablen geschrieben. Unserer Konvention nach sind die beiden ersten Parameter, die Oracle beim Aufruf mitsamt der Abfragedatei übergeben werden, Benutzername und Paßwort des Anwenders. Also sehen die zwei Variablendefinitionen jedesmal so aus:

Übersicht: Struktur der Abfragedateien



SELECT 'var usr_var = "&1";' FROM dual;
SELECT 'var pwd_var = "&2";' FROM dual;

Aufgerufen wird der Datenbank-Client dann nach dem folgenden Kommandozeilen-Schema:

sqlplus <usr>**/**<pwd> @<datei> <parameter1> …

Mit <datei> wird eine SQL-Abfragedatei angegeben. Das »&« in SQL-Anweisungen markiert für Oracle Platzhalter für die Parameter. Dabei ist zu beachten, daß die Parameter der Reihe nach an die Platzhalter vergeben werden, wenn diese mit Zahlen benannt sind. Den ersten Parameter erhält z. B. der Platzhalter »&1«, den zweiten »&2« etc. Was es mit der Tabelle »dual« auf sich hat, wird im folgenden Abschnitt erläutert.

Allgemeine Deklarationen

Es folgen einige Variablen-Deklarationen, d. h. Variablen für JavaScript werden erzeugt, aber noch nicht mit Werten belegt. Da aus SQL-Abfragedateien heraus keine direkten Textausgaben möglich sind und die Deklarationen der Einfachheit halber nicht in einer Datenbanktabelle gespeichert werden sollen, wird eine spezielle Strategie eingesetzt:

In SELECT-Anweisungen dürfen Zeichenketten angegeben werden, die Oracle dann in jeden gefundenen Datensatz einfügt. Deklarationen werden als solche Zeichenketten eingetippt und mit der Tabelle »dual« abgefragt. Damit jede Variable auch nur ein einziges Mal deklariert wird, enthält diese Hilfstabelle einen einzigen Eintrag. Beispiel:

SELECT 'var daten=new Array();' FROM dual;

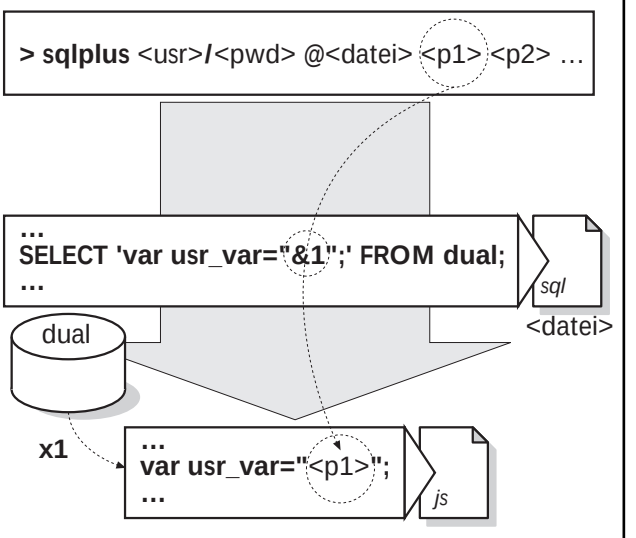
So sieht die Deklaration der Feldvariablen »daten« aus. Da aus »dual« keine Attribute abgefragt werden, besteht das Ergebnis lediglich aus der angegebenen Zeichenkette.

Optionale und obligatorische Variablen

Variablen, die unbedingt definiert werden müssen, sind »abfragen«, »globabf«, »usr_var«, »pwd_var«, »aktzustand« und »anzmodus«. Die Variable »daten« wird nur dann definiert, wenn Daten angezeigt oder durch Eingabe ermittelt werden sollen – ist »daten« leer, dann bietet die Seite im Browser nur Verzweigemöglichkeiten.

Bei »layout« ist es ähnlich, aber sie muß unbedingt definiert werden, wenn »daten« definiert wird. Die Variable »wichtig« muß gar nur bei Abfragen definiert werden, die mehr als einen Datensatz zurückliefern –

Beispiel: Abfragedateien und Parameter



Übersicht: Obligatorische und optionale Variablen

Obligatorisch	Optional	
abfragen[ ] **aktzustand** **anzmodus** **globabf[ ]** **pwd_var** **usr_var**	**daten[ ][ ]**	Wenn Datensätze oder Formulare übermittelt werden.
	layout[ ]	Wenn »daten[ ][ ]« definiert ist.
	wichtig[ ]	Wenn mehr als 1 Datensatz und Anzeige als Formular (anzmodus=2).

außerdem muß »anzmodus« dann auf »2« gesetzt werden, da ohne Formularelemente nicht zwischen den Datensätzen hin- und hergeschaltet werden kann.

Die zentrale Datenbankabfrage

In der Mitte der Abfragedatei kann die Variable »daten« definiert werden. Sie enthält die für Anwender interessanten Daten, und ihr Inhalt wird üblicherweise über eine normale Datenbankabfrage gebildet. Einzige Ausnahme: Wenn ein Eingabeformular aufgebaut werden soll, muß lediglich die erste Dimension von »daten« definiert und mit Leerstrings gefüllt sein:

```
SELECT 'daten[0]=new Array("",...,"");'
    FROM dual;
```

Ansonsten wird, entsprechend der Aufgabe, eine SELECT-Anweisung nach klassischem Muster erstellt und für JavaScript erweitert. Die Flexibilität dieser Erweiterung entsteht durch den Vorteil, daß sich Feldvariablen in JavaScript dynamisch erweitern lassen: Wird ein Feldindex benutzt, der bisher nicht existierte, dann wird die Feldvariable automatisch bis zu diesem Index erweitert. Außerdem kann die Anzahl ihrer Elemente jederzeit über ihr Objektattribut »length« ermittelt werden. Beides zusammen ermöglicht eine Zuweisung, die statisch aussieht, aber dynamisch funktioniert:

array[array.length] = <wert>

Zu Beginn ist die Variable noch leer, ihre Länge ist 0. Das steht auch im Objektattribut »length«, daher wird der Wert array[0] zugewiesen, dem ersten Element. Da es bisher nicht existierte, wird die Feldvariable erweitert bis zum Index 0 – um ein Element. Sie besitzt nach der Zuweisung die Länge 1. Ergo füllt die Zuweisung bei jeder Ausführung ein neues Element.

Damit JavaScript die Werte aus der Datenbank als Zeichenketten akzeptiert, müssen sie zwischen doppelten Anführungszeichen »"« stehen – einfache Anführungszeichen »'« markieren SQL-Zeichenketten. Damit Oracle den Ergebnistext nicht innerhalb einer Zeichenkette umbricht, werden die Anführungszeichen mit dem »||«-Operator konkateniert. Das Schema für die Abfrage sieht dann so aus:

```
SELECT 'daten[daten.length] = new Array(',
'"'||<attribut 1>||'",',
'"'||<attribut 2>||'",',
...
'"'||<attribut n>||'");'
FROM <tabellenliste>
WHERE <bedingungen>
ORDER BY <attributsliste>;
```

Beispiel: Aufarbeitung der SQL-Abfragen

	Änderung	Ergebnis
Name des Attributs für SQL		**name**
Doppelte Anführungszeichen markieren in JavaScript Zeichenketten	**" "**	**"name"**
Das Komma trennt in JavaScript einzelne Parameter ab	**,**	**"name",**
Einfache Anführungszeichen markieren in SQL die JavaScript-Bestandteile als Zeichenketten	**' '**	**'"'name'",'**
Der Konkatenierungsoperator verbindet die JavaScript-Bestandteile mit dem SQL-Ergebnis und verhindert, daß SQL innerhalb der Zeichenkette umbricht	**|| ||**	**'"'||name||'",'**

Beispiel: Variableninhalte aus der Datenbank

Abfrage

```
SELECT 'daten[daten.length]=new Array(',
            '"'||anrede||'",',
            '"'||vname||'",',
            '"'||name||'");'
FROM person ORDER BY name;
```

sql

anrede	vname	name
Herr	Mark	Miller
Frau	Karin	Müller
Frau	Anne	Moulin

person

DB-Tabelle

Ergebnis

```
daten[daten.length]=
    new Array("Herr","Mark","Miller");
daten[daten.length]=
    new Array("Frau","Anne","Moulin");
daten[daten.length]=
    new Array("Frau","Karin","Müller");
```

js

Veränderungen vornehmen

Bisher wurden lediglich Aufbau und Arbeitsweise des Systems beschrieben. Wie es gepflegt wird, also wie Zustände eingebaut und wieder gelöscht werden können, wird auf den folgenden Seiten in Form eines Leitfadens – nicht Regelwerks – aufgezeigt.

Grundsätzliche Überlegungen

Jeder Ausführung muß eine gründliche Planung vorangehen, wenn ihre Auswirkungen berechenbar bleiben sollen. Die Planung für die Systempflege beinhaltet, außer einer Abwägung des Nutzens neuer und alter Zustände, eine aktuelle Dokumentation, so daß Aufbau, Identifikation und Zusammenwirken der Zustände stets schnell und einfach ermittelt werden können. Es ist mühselig, den aktuellen Aufbau jeweils direkt aus den Steuertabellen der Datenbank herauslesen zu müssen.

Der Nutzen eines Zustands ist nicht einfach zu bestimmen. Grundsätzlich muß natürlich eine Nachfrage seitens der Anwender bestehen. Darüber hinaus hat sich aber auch gezeigt, daß simple und mengenreiche Datenlisten das Interaktionspotential des Systems nicht ausnutzen, einen hohen Aufwand für Rechenzeit und Übertragung bedeuten und nur selten für die gezielte Informationsrecherche taugen. Je stärker sich eine Abfrage durch Parameter eingrenzen läßt, desto besser stellt sich das Verhältnis vom Aufwand für das System zum Informationsbedarf des Anwenders dar. Auf der anderen Seite darf die Interaktion nicht übertrieben und der Anwender nicht auf eine lange Reise über viele Formulare geschickt werden, bis er letztendlich die gesuchte Information erhält (oder auch nicht).

Neue Zustände einbauen

Ist ein neuer, sinnvoller Zustand einmal gefunden, kann er mit Hilfe einer aktuellen Systemdokumentation schnell umgesetzt werden. Als erstes sind bisher freie Zustands- und Linknummern zu ermitteln. Die Nummern von kleinen Werten an aufsteigend der Reihe nach zu vergeben, ist nicht zwingend, erhöht aber die Übersichtlichkeit. Weiterhin ist noch zu klären, für welche Anwenderrollen die neuen Zustände verfügbar sein sollen. Dann muß für jeden Zustand eine eigene Abfragedatei erstellt werden, und schließlich sind die jeweiligen Daten für das Interaktionsmodell in die Steuertabellen einzutragen. Sobald das geschehen ist, kennt das System die neuen Zustände, und Anwender können auf sie zugreifen.

Eine Reihenfolge bei den einzelnen Arbeitsschritten ist nicht vorgeschrieben, aber wenn das System während des Betriebs erweitert wird, müssen die Abfragedateien und die Zustands- und Linkdaten (z. B. ihre Bezeichnun-

gen) vorhanden sein, bevor das Interaktionsmodell mit Folgezuständen ausgebaut wird. Auch empfiehlt es sich, die Zustände in umgekehrter Reihenfolge in das Interaktionsmodell »einzuhaken«, so daß nicht kurzfristig offene Verzweigungen entstehen.

Formulare mit Aufklappmenüs bauen

Bei Formularen, meist für die Eingabe von Suchparametern, kann es gewünscht sein, die Eingabe des Anwenders auf einige wenige vorgegebene Werte zu beschränken. Ein Attribut des Typs »Liste« eignet sich dafür bestens, sofern ein Nachteil dieser Lösung beachtet wird: Aufgrund der aktuellen Implementierung entspricht bei Listenattributen der Texteintrag des Aufklappmenüs dem Wert, der beim Abschicken an den Server weitergereicht wird. Falls die Bedeutung dieser Werte für den Anwender nicht ersichtlich ist, ist der Liste ein normales Textattribut mit Vorgabe und Erklärung vorzuziehen.

Wenn das Ergebnis nur einen einzigen Datensatz haben kann, ist ein Listenattribut sehr einfach einzubauen. Dazu wird es in der Variablen »layout« als Listenattribut definiert und in der Variablen »daten« anstatt mit einem einzelnen Wert mit einem ganzen Feld gefüllt. Werte können entweder manuell über die Tabelle »dual« oder über eine Datenbankabfrage bestimmt werden. Es ist aber sicherzustellen, daß die Datenbankabfrage auf jeden Fall ein Ergebnis liefert, da der Anwender sonst nichts auswählen oder eingeben kann. Beispiel:

```
select 'layout[1]=new layout_struktur(73,2);'
    from dual;
select 'daten[0][1]=new Array("Gerd","Georg");'
    from dual;
```

Es ist nur ein Datensatz vorhanden (Index 0). Das zweite Attribut (Index 1), globales Attribut 73 (erstes Element von »layout_struktur«), soll vom Typ »Liste« sein (zweites Element von »layout_struktur«, Listen haben hier den Wert 2). Aus der Tabelle »dual« werden keine Daten abgefragt – sie garantiert lediglich dafür, daß der vorangestellte Text genau einmal im Ergebnis ausgegeben wird. Weiteres Beispiel:

```
select 'layout[1]=new layout_struktur(73,2);'
    from dual;
select 'daten[0][1]=new Array();' from dual;
select 'daten[0][1][daten[0][1].length]=',
    '"'||vname||'";' from person
    where vname like 'G%';
```

Dieses Beispiel füllt »daten« über eine Abfrage der Tabelle »person«, gesucht werden alle Vornamen mit »G«. Der Ausdruck »daten[0][1][daten[0][1].length]«

Kurzübersicht: Neue Zustände einbauen

- Freie Zustands- und Linknummer suchen
- Für jeden Zustand eine SQL-Abfragedatei erzeugen und in ihren Namen die Automaten- und Zustandsnummer einbauen: **»trdb**aazz**.sql«**

  Beispiel: Die Datei für den Zustand »23« im Automaten »10« heißt »trdb1023.sql«.

  Tip: Die Datei nicht völlig neu erstellen, sondern eine bereits bestehende Datei mit ähnlicher Funktion kopieren und die relevanten Inhalte anpassen.

- Die Bezeichnungen und Nummern der neuen Zustände und Links in die Steuertabelle »tut_inf.Codes« eintragen.
- Für jede berechtigte Anwenderrolle alle neuen Folgezustände in die Steuertabelle »tut_inf.Zustand« eintragen.
- Dokumentation des Automaten aktualisieren.

erzeugt für jeden Treffer ein neues Element im Feld von »daten[0][1]«. Feste Indizes sind nicht möglich, da die Anzahl der Treffer nicht im voraus bekannt ist.

Bei Formularen, die mehr als einen Datensatz enthalten können, gestaltet sich der Einbau von Aufklappmenüs aufwendiger. Zunächst einmal wird »daten« über eine Datenbankabfrage gefüllt, wobei die Stelle des Listenattributs mit der Deklaration einer Feldvariablen besetzt wird. Danach wird über weitere Abfragen eine JavaScript-Programmschleife generiert, die in jedem Datensatz an die Stelle des Listenattributs dieselben Werte einfügt. Beispiel:

```
select 'layout[0]=new layout_struktur(95,1);'
    from dual;
select 'layout[1]=new layout_struktur(45,2);'
    from dual;
select 'layout[2]=new layout_struktur(97,1);'
    from dual;
```

… bis jetzt sind die Attribute bestimmt: das zweite soll eine Liste enthalten, die anderen normalen Text …

```
select 'daten[daten.length]=new Array(',
    '"'||matrnr||'",',new Array(),',
    '"'||stdgng||'");'
    from student
    where semester=7;
```

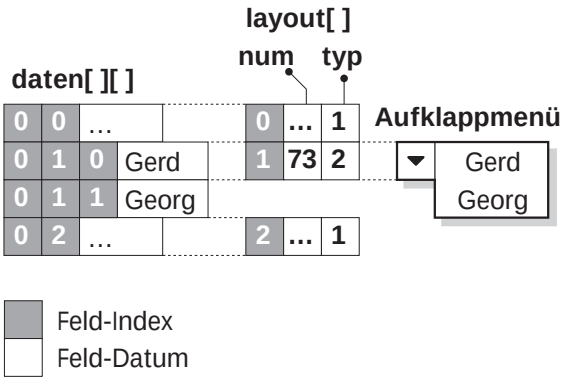
… jetzt sind die Textattribute gefüllt, die Listen enthalten erst leere Felder …

```
select 'for (var i=0;i < daten.length;i++) {'
    from dual;
select distinct 'daten[i][1][daten[i][1].length]=',
    '"'||TO_CHAR(ps)||'";' from indpss;
select '};' from dual;
```

Damit ist auch die Schleife erzeugt, die die Listenwerte einträgt. Die beiden Abfragen über »dual« erzeugen Schleifenkopf und -ende, sie bleiben für jede Abfragedatei unverändert. Die andere erzeugt die Zuweisungen dazwischen, wobei als zweiter Feldindex die Stelle des Listenattributs eingetragen werden muß. Die SQL-Anweisung »TO_CHAR( )« konvertiert Zahlen in Zeichenketten und muß bei Attributen eingesetzt werden, die in der Datenbank als numerischer Typ geführt werden.

Dieses Beispiel erzeugt ein Formular, das zu Studenten des siebten Semesters Matrikelnummern und Studiengang heraussucht. Zusätzlich wird ein Aufklappmenü erzeugt, das die Praktikumsnummern enthält, die überhaupt möglich sind (also 1 oder 2). Für ein reales Formular ist es natürlich kaum sinnvoll, wenige und bekannte

Beispiel: Daten für Aufklappmenüs



Beispiel: JavaScript-Schleifen erzeugen

Abfragedatei:

```
select 'for (var i=0;i<daten.length;i++) {'
    from dual;
select distinct
    'daten[i][1][daten[i][1].length]=',
    '"'||TO_CHAR(ps)||'";' from indpss;
select '};' from dual;
```

Ergebnisdatei:

```
for (var i=0;i<daten.length;i++) {
daten[i][1][daten[i][1].length]="1";
daten[i][1][daten[i][1].length]="2";
};
```

Werte über eine Abfrage zu füllen – manuell über »dual« erzeugte Zuweisungen sind schneller. Dafür muß die Beispielabfrage aber nicht geändert werden, falls zukünftig beispielsweise auch ein drittes Praktikum möglich ist. Denkbarer Einsatz: Als Vorgängerzustand für eine Abfrage, die zu Matrikel- und Praktikumsnummer die entsprechenden Praktikumsdaten heraussucht.

Zustände wieder entfernen

Um einen Zustand aus dem System zu entfernen, müssen zunächst die auf ihn verweisenden Einträge anderer Zustände aus der Tabelle »tut_inf.Zustand« gelöscht werden, danach die Folgezustände des Zustands selbst. Aus der Tabelle »tut_inf.Codes« müssen dann seine Zustands- und Linkdaten gelöscht werden. Schließlich sind noch die SQL-Abfragedateien zu entfernen. Falls der gelöschte Zustand einzigartige Folgezustände hat, auf die sonst kein anderer Zustand verweist, müssen auch sie gelöscht werden.

Übersicht: Zustände entfernen

- Verweise auf den Zustand löschen:
  delete from tut_inf.Zustand where
  f_tab='aa**' and f_zust='**zz**';***
- Seine Verweise auf Folgezustände löschen:
  delete from tut_inf.Zustand where
  tab_nr='aa**' and znr='**zz**';**
- Zustands- und Linkdaten löschen:
  delete from tut_inf.Codes where
  tab_nr='aa**' and**
  dim_nr='zz**' and**
  sorte=1;*

  delete from tut_inf.Codes where
  tab_nr='aa**' and**
  dim_nr='ll**' and**
  sorte=2;*
- SQL-Abfragedatei entfernen:
  rm trdbaazz**.sql***
- Einzigartige Folgezustände löschen, auf die kein anderer Zustand verweist
- Dokumentation des Automaten aktualisieren

* *aa = Nummer des Automaten, in dem sich der Zustand befindet*
zz = Nummer des Zustands
ll = Nummer des Links, der auf den Zustand verweist

Index