Einleitung

Datum: 18.02.2009
Prüfung: **Diplomprüfung Praktische Informatik**
• Web Technologien
• Programmierung von Webbasierten Software Portalen (Web Technologien 2)
• Introduction to Database Systems
• Indexstrukturen
Prüfer: **Professor Schroeder, Professor Seidl**
Dauer: ca. 50 Minuten
Note: **1,3**

Schroeder: Ich werde Fragen zu Portaltechnologie an passenden Stellen in die Fragen zu Webtechnologie einstreuen und beides nicht so streng trennen.
Ich: OK.

Schroeder: Was unterscheidet denn Web-Engineering von Softwareengineering?
Ich: Charakteristika von Web-Applikationen erklärt (siehe Lernskript Anna-Lea)
Schroeder: Und was sind Web-Portale?
Ich: vereinigen den Zugriff auf verteilte Informationen und Dienste. Es gibt zwei Varianten: personalisierte Portale und spezialisierte
Schroeder: Genau. Das sind ja die horizontalen und vertikalen.

Schroeder: HTTP ist ja ein zustandloses Protokoll, aber wir kennen ja auch Web-Applikationen wie z.B. Amazon bei denen eine Warenliste geführt wird. Wie kann man denn so was realisieren?
Ich: mittels Cookies, Hidden Fields oder URL Rewriting. Alles erklärt.
Schroeder: (hat meine Erklärung zu URL-Rewriting nicht ganz gefallen) Wie funktioniert das denn so richtig? Wie macht der Server das? Der gibt ja keine URL zurück, sondern?
Ich: ein Dokument
Schroeder: Genau! Und wie geht das jetzt mit dem URL-Rewriting?
Ich: Ach so. Der Server encodeURL alle Hyperlinks des Dokuments.
Schroeder: Was sind denn Nachteile von Hidden Fields?
Ich: Na ja, eigentlich haben alle 3 Varianten den Nachteil clientseitig ausgelesen und manipuliert werden zu können.
Schroeder: Ja und was sind die speziellen Nachteile von Hidden Fields, wenn sie mal an Sharepoint denken? Hier wird das ja mit HF gemacht?
Ich: Mh, die HF werden ja in ein Form eingebettet.
Schroeder: Genau und dabei kann die Formatierung des Forms kaputtgehen.

Schroeder: Wie funktioniert Sessionmanagement?
Ich: Client meldet sich erstmalig beim Server. Der erstellt SessionID und sendet diese mit einer der Methoden von oben zum Client. Der Client sendet bei jeder nächsten Anfrage die SessionID wieder mit zurück zum Server, damit der ihn identifizieren kann.
Schroeder: Mh, und was macht der Server noch?
Ich: Er allokiert Speicherplatz für Session-Variablen und so.

Schroeder: (weiß nicht mehr, wie die Frage wirklich war, wollte aber auf Seperation of Concern hinaus)
Ich: Mit Model-View-Controller-Konzept. Zum Beispiel bei HTML und CSS.

Schroeder: Welche CSS-Konzepte kennen sie? Wir hatten da ja 10 oder so verschiedene.
Ich: Elementselektoren, Klassenselektoren, ID-Selektoren, Attributselektoren...
Schroeder: Danke, das waren die wichtigsten. Was ist denn mit Kaskadierung in diesem Zusammenhang

gemeint?
Ich: Das sind die Regeln nach denen entschieden wird, welche Formatierungsanweisung bevorzugt wird, wenn mehrere Anweisungen auf ein Element zutreffen.
Schroeder: Und wie geht das?
Ich: Die Anweisung die näher am Element ist, zieht.
Schroeder: Richtig.

Schroeder: Wie würden Sie denn Javascript in Hinsicht von „großen" objektorientierten Sprachen einschätzen?
Ich: JS wird nicht für Rich-Applikationen verwendet sondern nur im Zusammenhang mit HTML. Variablen werden nicht deklariert, hat keine Klassen und damit keine Klassenvererbung und alles sind Objekte auch Funktionen. DOM kurz erwähnt. event-getrieben
Schroeder: mh und wie funktioniert das mit den Objekten?
Ich: (hingeschrieben) var s = new String('x')
Schroeder: (aber das wollte er wohl nicht) Ja das ist ein natives? Objekt. Wie ist das mit den Funktionen?
Ich: (hingeschrieben) Funktion point(x,y) { this.x=x; this.y=y; }
Schroeder: Ja genau. Und kann man so einem Objekt auch Methoden hinzufügen?
Ich: Klar.(hingeschrieben) Funktion point(x,y) { this.x=x; this.y=y; this.dist = Distance} (erwähnt das die Funktion Distance innerhalb von point oder auch ausserhalb von point sein kann, weil Funktionen sowieso global sind)

Schroeder: Welche Programmierungsmodelle kennen Sie?
Ich: dokumentenzentriertes wie PHP und komponentenzentriertes wie Servlets. Da gibt's noch ein drittes das hab ich vergessen.
Schroeder: das programmorientierte wie CGI. (weiß die Frage nicht mehr, aber er wollte einen Lifecycle sehen)
Ich: den für Servlets oder jsp's?
Schroeder: JSP
Ich: Browser fragt JSP Dokument an. Der Container empfängt den Request...
Schroeder: Erst der Webserver, dann der Container.
Ich: Ja klar. Also der Container mappt aus dem Deployment Descriptor den Namen..
Schroeder: Erklären Sie das mit dem Mapping genauer!
Ich: Browser hat url-pattern; in DD ex dazu DD-Name über diesen die eigentliche Klasse (hab hier eigentlich Servlet-Mapping erklärt, war aber wohl genau das was er hören wollte)
Schroeder: So und wie funktioniert das jetzt weiter, wir haben die jsp.
Ich: Die wird in ein Servlet.java umgewandelt, dann in eine class-File
Schroeder: Wird das immer gemacht?
Ich: Bei jsp ja. Ach nein! Nur nachdem jsp deployed wurde. Servlets leben dann ja ewig.
Schroeder: (grinst) Genau. Und wie geht's dann weiter, wenn wir das Servlet haben?
Ich: Container erzeugt Thread, Request- und Responseobjekte und gibt die an den Thread; ruft _jspService auf, das dann DoGet oder DoPost aufruft.

Schroeder: Was sind denn die Basiskomponenten von Web-Portalen?
Ich: Container als funktionale Einheit eines Portals - "kleines" Stück Software, was im Portal läuft, Suche, Login und Layoutänderungen
Schroeder: Ja genau. Und wenn Sie jetzt mal an die Authentifizierung denken, wie funktioniert das im Sharepointaufbau?
Ich: (kuck wohl etwas ratlos)
Schroeder: das fängt ja mit der Webanwendung an
Ich: Ah, darunter die SPSite, dann SPWeb, SPList und SPListItem
Schroeder: Ja und wie ist das jetzt mit der Autentifizierung?

Ich: Die vererbt sich nach unten. Kann eingeschränkt oder erweitert werden, was sich auch wieder weitervererbt.
Schroeder: Richtig. Ich bin fertig.

Seidl: Was ist eine Relation?
Ich: ist die endliche Teilmenge des kartesischen Produkts der Attribute

Seidl: Was brauchen wir alles für eine Relation?
Ich: Attribute die dann die Spaltennamen der Tabelle werden (ich glaube wir haben uns hier etwas missverstanden, ging eine Weile drumherum bis ich auf die Datentypen, Domänen wollte er wohl hören, kam)

Seidl: Was ist eine funktionale Abhängigkeit?
Ich: Eine Attributmenge bestimmt eine andere Attributmenge eindeutig.

Seidl: Was ist ein Schlüssel?
Ich: Eine Attributmenge die eine andere Attributmenge eindeutig bestimmt. PK ist minimale Menge, die eine andere bestimmt.
Seidl: Was ist ein Super-Schlüssel?
Ich: das ist die maximale Menge
Seidl: Nein
Ich: nee, eigentlich jeder Schlüssel der nicht minimal ist
Seidl: na ja, genaugenommen ist der PK auch ein Superschlüssel
Ich: mh, ok. Also alle Schlüssel.

Seidl: und was sind Fremdschlüssel?
Ich: referenziert einen PK aus der Mastertabelle. Bild Master- und Slavetabelle aufgemalt.
PK ist in der Mastertabelle eindeutig, in der Slave kann er mehrfach auftreten (1:n), muss aber als PK in der Master vorhanden sein.

Seidl: (das übliche {MNo, Name, Vo, Title} Beispiel) Ist das Beispiel gut?
Ich: Nein, da es mehrere Konzepte vereinigt
Seidl: Was ist nicht so gut daran?
Ich: Es können Anomalien auftreten.
Seidl: Was passiert z.B. beim Löschen?
Ich: Werden alle Studenten gelöscht, die die einzigen in einer Vorlesung waren, verschwindet auch die Vorlesung.
Seidl: Wie kann man das Beispiel besser machen?
Ich: Mit dem Synthesealgorithmus oder dem Dekompositionsalgorithmus.
Seidl: Machen Sie mal Synthese.
Ich: (gemacht: funkt. Abhgk.: M-->N; V-->T; MV->NT, linksreduziert: M-->N; V-->T; MV->N; MV->T, rechtsreduziert & überflüssige (Armstrongaxiome) weg: M-->N; V-->T,
Tabellen: {{M,V},{M-->V}}, {{V,T},{V—>T}}, prüfen, ob Superschlüssel für alle Attribute ex- hier nicht → noch eine Tabelle mit ursprünglichem Schlüssel: {M,V}

Seidl: (ich habe keine Ahnung mehr wie wir darauf kamen, aber es ging um's kartesische Produkt) Schreiben Sie es bitte mal in relationaler Algebra.
Ich: was? Wie soll das denn gehen? Kenne nur Natural Join <liegendes geschlossenes Kreuz>. (Haben über Natural Join in RA und SQL gesprochen, wie wird der gebildet und so; alle Konzepte aus RA in SQL „übersetzt")

Seidl: (am Ende zeigt er mir das Kreuz, das für das kartesische Produkt steht <liegendes offenes Kreuz>)
Ich: (Kannte ich nicht, habe ich ihm gesagt. Ich glaube, ich habe hier keine Punkte verloren, weil wir insgesamt viel über RA und SQL geredet haben)

Seidl: Brauchen wir Indexe überhaupt? Und Warum?
Ich: Effizienteres Suchen von phys. Datensätzen
Seidl: Wie ist die Zeitkomplexität ohne Indexe?
Ich: O(n), jeder Datensatz wird angepackt.

Seidl: Welche Indexstrukturen kennen Sie denn im 1-dim?
Ich: Hashing in diversen Varianten und Bitmap-Indexe
Seidl: Und was noch?
Ich: mh, weiß nicht mehr.
Seidl: jetzt haben sie die wichtigste Struktur vergessen!
Ich: Bäume! B-Baum im 1-dim!
Seidl: schön

Seidl: Wie funktionieren den Bitmap-Indexe?
Ich: (will malen) Wissen Sie ein Beispiel?
Seidl: Wochentage.
Ich: (male Bitvektoren Mo, Di, Mi) alles erklärt
Seidl: Sind Wochentage eigentlich ok für Bitmapindexierung?
Ich: Ja klar, es sind ja nur 7.
Seidl: Wie ist eigentlich die Komplexität? Ich weiß, das stand nicht im Skript, aber?
Ich: mh, man müsste die Mapping-Tabelle miteinbeziehen und...
Seidl: na ja, haben Sie recht. Ich wollte aber nur die Komplexität der Vektoren wissen.
Ich: Klar auch O(n) (ich kuck wohl ziemlich fragend, was der Sinn wohl sei)
Seidl: ja klar, die Bitmapindexierung ist schneller als sequenzieller Scan, aber trotzdem O(n)
(nach gemeinsamen Überlegungen kamen wir darauf, dass für 100Byte pro Datum (einzahl Daten) die durch Bits ersetzt werden, die Laufzeit 800mal schneller ist, wenn man die Mapping-Tabelle und so ausser Acht lässt)