Vertiefungsprüfung Lakemeyer: DB, KR, KI am 27.04.2006

Von Lakemeyer gibt es ja schon viele Protokolle, aber wenig über Datenbanken, daher erzähle ich ausführlich nur was über diesen Teil. Wir haben auf seinen Vorschlag mit DB angefangen und dann mit KR und KI weitergemacht. Die Reihenfolge war nicht so günstig für mich, da er für KI nicht genug Zeit hatte und es mein stärkstes Fach war. Schlagt eventuell eine eigene Reihenfolge vor. Für Normalformen Im Gegensatz zu anderen Profs, hat Lakemeyer kein spezielle Thema gehabt, wo er extra in die Tiefe geprüft hat. Zu Beginn haben wir ein wenig geklönt, während er den Laufzettel ausfüllte. Ich vermute, er macht das um die Nervösität des Prüflings abzubauen. Generell fand ich die Prüfung nicht besonders nervenaufreibend, aber ich war auch gut vorbereitet und bin mittlerweile "prüfungserfahren". Lakemeyer stellte die Fragen so exakt, dass ich wusste worauf er hinaus wollte. Z.B. Lichter fand ich in dem Zusammenhang nicht so gut. Ausserdem waren Lakemeyers Übergänge meisterhaft. Ich hab erst nach der zweiten Frage gemerkt, dass wir von DB zu KR gegangen sind. :-) Lakemeyer als Prüfer ist ein "Laky Strike" (Verzeiung für den schlechten Kalauer, aber der musste sein. :-) Insgesamt habe ich eine 1.7 gekriegt, im nachfolgend beschriebenden Datenbankteil eine glatte Eins.

Laky: Fangen wir mal mit der physischen Ebene an. Was können sie mir dazu erzählen?

Ich: Auf der Festplatte werden die Daten in Blöcken gespeichert. Um sie schnell zu finden braucht man Indexe. Man unterscheidet Bäume und Hashing ... und dann ist da noch ISAM. ISAM funktioniert wie ein Daumenindex im Telephonbuch, also, wenn man 26 Indexeinträge hat, könnte jeder einen Block mit allen Namen mit einem bestimmten Anfangsbuchstaben referenzieren. Der ist ja auch bei MySQL bekannt.

Laky: Wie schnell ist denn die Suche?

Ich: Die ist O(Log(n)).

Laky: Und wie ist das beim Einfügen eines neuen Datensatzes?

Ich: Wenn der Block überläuft muss der Index nach rechts verschoben werden.

Laky: Und wie schnell geht das?

Ich: *murmel* *kritzel* Log(n) für die Suche im Index, Log(n) für die Suche im Block, ah ja! Das Rechtsverschieben des Index ist linear.

Laky: Richtig, Einfügen hat die Komplexität O(n). Und was haben wir noch für Indexstukturen?

Ich: B-Bäume, das sind allround Indexstukuren, Suchen und Einfügen gehen da in Log(n) und dann noch Hashing. Das ist sehr schnell, aber bei Range-Checks nicht geeignet. Dann muss man halt B-Bäume nehmen.

Laky: *mal auf Papier* Wenn sie einen Namen im Telephonbuch suchen, würden sie dann Hashing nehmen?

Ich: Ja.

Irgendwie war er noch am zweifeln, ob ich Hashing richtig verstanden hatte und fragte was zum Zusammenhang Indexeintrag/Block, worauf ich die Frage kurz beantwortete und dann erweiterbares Hashing erklärte.

Laky: *Papier rüber reich* Jetzt ist ja bald Fussball WM, deshalb habe ich hier ein WM Schema aufgezeichnet.

Mannschaft | Trainer | Datum | Stadium

Ich und Beisitzer: *lach*

Laky: In welcher NF ist denn das?

Ich: Also in 1NF ist es auf jedenfall, weil die Attribute atomar sind. Für die 2NF muss man sich die FDs angucken. Errr, ich hab ja keine Ahnung von Fussball.

Ein paar Fragen zur Organisation der Fussballspiele gefragt. Am Schluss hatte ich folgende FDs: {Mannschaft --> Trainer, Trainer --> Mannschaft} Ich wusste das was fehlt, aber hab nix gesagt und er hats nicht bemängelt.

Laky: Was würden sie denn als Schlüssel nehmen?

Ich: Mannschaft, Datum und Stadium.

Irgendwie haben wir uns dann darauf geeinigt, dass auch {Mannschaft, Datum --> Stadium} gilt.

Laky: Also, was ist dann der Schlüssel?

Ich: Mannschaft und Datum. Um das in 2NF zu bringen müssen wir die Relation aufspalten in R1 = (Mannschaft, Trainer) und R2 = (Mannschaft, Datum, Stadium).

Laky: Gut. Sie haben ja eben schon MySQL angesprochen, das hat ja was mit SQL zu tun. (Absolut genialer Übergang! lol) Wie würde man denn einen Constraint angeben, dass nur Mannschaften eingetragen werden dürfen, die in maximal zwei Stadien spielen.

Ich: *Entgeistert guck* Öhhh. Also Constraints werden ja in der Tabellendefinition angegeben. Ähhhh. Das dürfte irgendwie so gehen. *kritzel*
SELECT Mannschaft FROM R2 GROUP BY Stadium HAVING COUNT(Stadium) <= 2; Hmmm, das geht aber doch noch einfacher... *kritzel* MAX(COUNT(Stadium ... Nee, das klappt nicht.

Laky: Das erste war ja schon ok. Wir haben ja auch OO Erweiterungen gesehen. Was muss denn ein OODBMS zusätzlich haben?

Ich: Sie muss in eine Zielsprache einbettbar sein.

Laky: Wie sieht denn diese EInbettung aus?

Ich: Hmmm, ich habe was mit Hibernate gemacht, das ist ein Objekt-Relationaler Mapper. Da hat man ein Java-API und einen Class-Enhancer, der den Bytecode mit zusätzlichen Dingen anreichert. Und ich hab mal was mit PSE gemacht, das ist eine richtige OODB und da gab es auch Java oder C++-APIs. (Ich glaube, das wollte er nicht wissen, hat mich aber nicht unterbrochen.)

Laky: Und was für zusätzliche Eigenschaften braucht es, damit es sich OO nennen darf?

Ich: Naja, die Standard OO Features: Vererbung, Objektidentität, Test auf Objektgleichheit, ...ja...

Laky: Ja, das reicht ja auch. Was haben denn Datenbanken mit der Closed-World-Assumption zu tun?

Ich: Datenbanken haben die CWA, während Wissensbasen das nicht haben müssen. (CWA am Beispiel vom Bahnfahrplan erklärt)

Dann kamen weitere Fragen zur CWA und KR allgemein, danach noch zu KI.