

## Prüfungsprotokoll Praktische Informatik (Kerngebiet) mit Vertiefungsgebiet Systemprogrammierung (P3)

Prüfer:	<b>Doz. Dr. Martin Lehmann</b> (Beisitzer: Jürgen Wolf)
Prüfling:	Stefan Witt (10. Semester)
Datum:	11. Dezember 2002 um 10 <sup>00</sup> Uhr in Raum F-629
Vorbereitung:	3 Monate
Note:	1,3
Kern-Vorlesungen:	Betriebssysteme und Systemprogrammierung (bei Martin Lehmann im SoSe 2000) Verteilte Systeme (bei Winfried Lamersdorf im SoSe 2000) Einführung in Datenbanksysteme (bei Björn Kesper im WS 1999/2000) Einführung in die Softwaretechnik (bei Guido Gryczan im WS 1999/2000)
Vertiefungs-Vorlesungen:	Compilerbau (bei Martin Lehmann im WS 2000/2001) OOSE I (bei Heinz Züllighoven im WS 2000/2001) OOSE II (bei Heinz Züllighoven im SoSe 2001) Einführung in die Robotik (bei Bärbel Mertsching im SoSe 2001) Substrukturelle Logiken (bei Berndt Farwer im WS 2000/2001) Petrietze (bei Daniel Moldt im WS 2000/2001)
Literatur:	Nur die vereinbarte Literatur war die Prüfungsgrundlage.  ⇒ <i>Maurice J. Bach: The Design of The UNIX Operating System, Prentice Hall, 1986</i> Kapitel 1–10  ⇒ <i>Ramez Elmasri, Shamkant B. Navathe: Fundamentals of Database Systems, Addison Wesley, 2nd Ed. 1994</i> Kapitel 1–7, 12, 17, 18, 19  ⇒ <i>Bertrand Meyer: Object-Oriented Software Construction, Prentice Hall, 2nd Ed. 1997</i> Kapitel 1, 2, 3, 5, 6, 7, 8, 14  ⇒ <i>Andrew S. Tanenbaum: Modern Operating Systems, Prentice Hall, 1992</i> <i>Kapitel 1–6 (für Betriebssysteme), Kapitel 9–13 (für Verteilte Systeme)</i>

Lehmann (L.): Suchen Sie sich eines der Gebiete aus.

Ich (I.): Datenbanken, Thema Relationales Modell.

L.: Dann erzählen Sie mal.

I.: Das Relationale Modell ist das konzeptionelle Datenmodell für Relationale Datenbanken. Es basiert auf Tabellen, die mathematisch durch Relationen beschrieben werden.

L.: Und wie sind die definiert?

I.: Ein Relationenschema ist definiert als  $R = (A_1, A_2, \dots, A_n)$ , und die Relationsinstanzen sind  $r(R) \subseteq \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$ , also das Kreuzprodukt der Domänen der einzelnen Attribute.

L.: Das ist ja eine recht umständliche Definition (*dem stimmte ich zu*). Und was machen Sie nun mit den Relationen der Datenbank?

I.: Die Relationen können mit Operatoren verknüpft werden, um gewünschte Informationen abzufragen.

L.: Was sind denn die elementaren Operatoren?

I.: Das sind Projektion, Selektion, Kreuzprodukt, Vereinigung und Differenz.

L.: Wie funktioniert die Differenz an einem Beispiel (*Ich schreibe eine Beispieltabelle mit Vor- und Nachnamen auf und erkläre die Differenz*)? Was ist die symmetrische Differenz?

- I.: Kenne ich nicht.
- L.: Das ist  $(A \setminus B) \cup (B \setminus A)$ . Ist das ein elementarer Operator ( $\rightarrow$ nein)? Und gehören die Joins zu den elementaren Operatoren?
- I.: Nein, die lassen sich durch Kreuzprodukt, Selektion und Projektion ausdrücken. Es gibt Theta-Join, Equi-Join und Natural Join.
- L.: Wie funktionieren die an einem Beispiel?
- I.: Tabelle gezeichnet und erklärt.
- L.: Kommen wir zu den Normalformen. Wie sehen die erste bis vierte Normalform aus?
- I.: 1NF besagt, dass jedes Attribut aus elementaren Datentypen aufgebaut ist.
- L.: Was heißt das genau? Sie können einen Wein doch z.B. durch Jahrgang und Geschmack beschreiben, und genauso kann ich doch auch eine Zahl durch Attribute beschreiben, z.B. dass die zweite Ziffer eine 3 ist und die Zahl durch 27 teilbar sein muss.
- I.: Elementare Datentypen sind hier z.B. Integer und Strings (*präziser konnte ich das nicht formulieren*).
- L.: Naja, lassen wir das Philosophieren. Was sind denn nun die Normalformen?
- I.: 1NF hatten wir ja gerade. 2NF besagt, dass jedes Attribut, das nicht Teil des Primärschlüssels ist, funktional voll vom Primärschlüssel abhängt.
- L.: Was ist ein Schlüssel, und was ist ein Primärschlüssel?
- I.: Ein Schlüssel ist eine Teilmenge von Attributen, die für jede Relationsinstanz eindeutig ist. Ein Primärschlüssel ist ein minimaler Schlüssel.
- L.: Und wie berechne ich einen minimalen Schlüssel?
- I.: Berechnen kann man einen minimalen Schlüssel nicht. Er ergibt sich aus der Anwendungsdomäne. So ist z.B. die ISBN-Nummer von Büchern eindeutig.
- L.: Nein, da liegt Ihnen eine Fehlinformation vor, die ISBN-Nummer ist nicht eindeutig (*er erklärt das ausführlich*).
- I.: Dann nehme ich als Beispiel Personalausweisnummern. Die sind eindeutig. Einige Datenbanksysteme benutzen standardmäßig automatisch inkrementierte Zahlen, so dass der Schlüssel künstlich erzeugt wird.
- L.: Das setzt aber voraus, dass der Wertebereich der Zahlen groß genug ist. Und was sind die zweite und dritte Normalform?
- I.: In der 2NF müssen alle Nicht-Schlüssel-Attribute vollständig funktional vom Primärschlüssel abhängig sein. In der 3NF werden alle funktionalen Abhängigkeiten als  $X \rightarrow A$  dargestellt ( $X$  ist eine Menge von Attributen), und sie müssen entweder trivial sein, oder  $X$  muss ein Superschlüssel sein, oder  $A$  muss in einem Kandidatenschlüssel enthalten sein.
- L.: Genau. Und was sind die vierte und die berühmte fünfte Normalform?
- I.: Die 4NF ist noch restriktiver und berücksichtigt Multivalued Dependencies, und die 5NF berücksichtigt auch Join Dependencies.
- L.: (*unterbricht*) Kommen wir zu einem handfesteren Thema. Was sind B-Bäume?
- I.: Ein B-Baum ist eine Index-Datenstruktur, um den Zugriff auf die Datensätze zu beschleunigen (*Ich zeichne einen B-Baum als Beispiel und erkläre, wie die Indexeinträge gefunden werden*).
- L.: Und wieso sind B-Bäume besser als binäre Bäume?
- I.: Die Knoten enthalten mehrere Einträge und werden daher schneller traversiert.
- L.: Wie wird ein B-Baum charakterisiert?
- I.: Es gibt einen Parameter, der besagt, wie viele Einträge ein Knoten maximal haben kann.
- L.: Es gibt auch einen zweiten Parameter, der die minimale Füllung angibt. Wieso?
- I.: Ich dachte, es gilt nur für  $B^+$ -Bäume, dass die Knoten zu mindestens 50% gefüllt sein müssen. Durch diese Bedingung wird die Baumtiefe begrenzt.
- L.: Und wieso will man die Baumtiefe begrenzen?
- I.: Das Traversieren geht schneller je geringer die Baumtiefe ist.
- L.: Das sehe ich nicht so ein. Beim Traversieren innerhalb jedes Knotens muss eine lineare Suche durchgeführt werden, und die kostet Zeit. Wo ist der Index denn gespeichert?
- I.: Er ist im Speicher bzw. auf Festplatte.

- L.: Genau, und Festplattenzugriffe dauern lange, und daher sollen möglichst wenige durchgeführt werden. Wissen Sie, wofür das „B“ in „B-Baum“ steht?
- I.: Nein.
- L.: Es gibt da zwei Theorien. Entweder es steht für „Boeing“, weil es dort zuerst eingesetzt wurde, oder es könnte für „Bayer“ stehen, dem Erfinder der B-Bäume. Machen wir mal einen Schnitt. Im Gebiet Betriebssysteme haben Sie ein Kapitel über Deadlocks. Was sind das?
- I.: Ein Deadlock ist eine Situation, in der mehrere Prozesse jeweils auf ein Ereignis warten, das nur ein anderer Prozess auslösen kann. Es gibt 4 Bedingungen, die hinreichend für einen Deadlock sind: Wechselseitiger Ausschluss von Betriebsmitteln, Nichtentziehbarkeit von Betriebsmitteln, zirkuläres Warten und Halten und Warten (*das musste ich noch genauer erklären*).
- L.: Wie kann man Deadlocks grundsätzlich vermeiden?
- I.: Man muss mindestens eine der 4 Bedingungen beseitigen, z.B. Betriebsmittel durch das Betriebssystem entziehen. Eine richtige Lösung wäre das allerdings nicht. . .
- L.: Nein, das stimmt.
- I.: Man könnte auch die Betriebsmittel nummerieren und nur Reservierungen in dieser bestimmten Reihenfolge zulassen. Im Falle von Speicherreservierung geht das aber nicht.
- L.: Doch, das geht auch. Man könnte Hauptspeicherreservierungen vor Hintergrundspeicherreservierungen durchführen, und zum Reservieren von mehr Hauptspeicher muss erst der Hintergrundspeicher wieder freigegeben werden. Es gibt ein bekanntes Betriebssystem, das so arbeitet. . . Wie können Deadlocks erkannt werden?
- I.: Man kann einen Graphen konstruieren, der die reservierten und angeforderten Betriebsmittel darstellt. Wenn der Graph einen Zyklus enthält, liegt ein Deadlock vor (*Beispiel gezeichnet*).
- L.: Wie aufwändig ist das Finden von Zyklen in einem Graph?
- I.: Das hat die Komplexität ist  $O(n^3)$  und ist lösbar mit dem Warshall-Algorithmus.
- L.: Der wird ja zum Auffinden der reflexiven, transitiven Hülle benutzt. Sie können zwar testen, ob dort ein Knoten doppelt vorkommt, aber das ist etwas umständlich.
- I.: Ich könnte auch einfach Tiefensuche im Graphen benutzen und so testen, ob ich einen Knoten mehrfach kreuze.
- L.: Ja. Kommen wir zu Verteilten Systemen. Deadlocks gibt es dort ja auch, aber wie sieht es dort mit der Synchronisation aus?
- I.: In einem verteilten System kommunizieren die Prozesse nur durch Nachrichtenaustausch. Um Ereignisse zu synchronisieren und abzugleichen, müssen die Nachrichten in einer bestimmten Reihenfolge an die Prozesse geliefert werden. Es gibt verschieden starke Anforderungen an die Ordnung der Nachrichten: ungeordnet, FIFO-Ordnung, kausale Ordnung und totale Ordnung.
- L.: Ja, schon. Zu Anfang geht Tanenbaum in seinem Buch aber auf Uhrensynchronisation ein.
- I.: Es gibt logische Uhren, die die verteilten Ereignisse durchnummerieren, und physikalische Uhren.
- L.: Wie funktionieren logische Uhren, und wie gewährleisten sie totale Ordnung?
- I.: Sie gewährleisten nur eine kausale Ordnung, soweit ich weiß. Dazu wird die „happens before“-Relation definiert. Jeder Prozess hat eine logische Uhr, die bei jedem Ereignis inkrementiert wird und mit jeder Nachricht mitgesendet wird. Beim Empfang einer Nachricht wird die logische Uhr vorgestellt, wenn die Uhrzeit der empfangenen Nachricht größer als die eigene logische Uhrzeit ist (*Anmerkung/Nachtrag: Für eine totale Ordnung müssen die Prozesse durchnummeriert werden; darauf ist L. aber nicht eingegangen*).
- L.: Und wie funktioniert die Synchronisation physikalischer Uhren?
- I.: Es gibt verschiedene Algorithmen. Bei „Cristian’s Algorithmus“ gibt es einen zentralen Zeitserver, der von den Clients nach der aktuellen Zeit gefragt wird. Allerdings ist der Wert, den der Server übermittelt, wegen der Signallaufzeit und der Netzwerkverzögerung nicht exakt. Die Clients bilden einen Mittelwert, um die aktuelle Zeit zu bekommen.
- L.: Wie bilden sie einen Mittelwert, wenn sie nur einen Wert als Antwort bekommen?
- I.: Die Clients haben einen Sende- und einen Empfangszeitpunkt. Die Uhrzeit, die sie vom Server empfangen, muss die Zeit von einem Zeitpunkt zwischen Senden und Empfangen sein. Um einen genaueren Wert zu erhalten, können sie mehrere Anfragen senden.
- L.: Kommen wir noch zum Thema Softwaretechnik. Was ist ein Abstrakter Datentyp?
- I.: Ein ADT ist ein Datentyp, dessen Implementation nicht offengelegt ist. Es sind nur seine Funktionen definiert und die Vorbedingungen, die zum Aufruf der Funktionen erfüllt sein müssen.
- L.: Ja. Bitte warten Sie kurz vor der Tür.