Datenbanken

Also nehmen wir mal an, sie wollten eine Datenbank entwerfen. Was tun sie da?

> Ich behaupte frank und frech, daß wohl nur eine relationale DB in Frage kommt. PW meint, ich könne auch eine IMS-Struktur entwerfen, was ich jedoch entschieden ablehne. PW und der Beisitzer grinsen. Also: Boyce-Codd, *lossless join*, *dependency preservation* und Redundanzfreiheit (kurz!) erklären, den BCNF-Algorithmus vormachen und sagen, daß das nicht immer funktioniert. In diesen Fällen ist 3NF angesagt.

BCNF-Verfahren genauer erklären, insbesondere darlegen, was mit den FD's geschieht, die die BCNF verletzen.

Künstliche Intelligenz

Na was haben wir denn als drittes Fach vereinbart? Aah, KI. Herr Starke, in der KI findet die Prädikatenlogik (PL) viele Anwendungen. Wo wir eben schon bei Normalformen waren, gibt es so etwas auch in der PL?

> Klar, Klauselform. Bekanntester Vertreter ist die Hornklauselform aus PROLOG. Umwandlung von PL-Formeln in Klauselform vormachen, dabei alle Schritte erkären. Ich weise auf ein PROLOG-Programm hin, welches in der Literatur (Clocksin-Mellish) erschienen ist. PW sagt, ich könne ihm auch das Programm hinschreiben (grinst wieder). Ich lehne ab und erkläre das Verfahren, Pränex-NF, Skolem-NF, Klauseln als Menge schreiben.

Wie geht die Umwandlung in Skolem-Form?

> Beispiel

Was macht man mit freien Variablen?

> grübel, denk...quantifizieren.

Was macht man mit Konstanten?

> Nullstellige Skolemfunktionen.

Sind Klauseln und Formeln das gleiche?

> Nein, sie sind lediglich erfüllbarkeitsäquivalent.

Wie ist es mit der Entscheidbarkeit der PL?

> Semi-Entscheidbar.

Wie benutzt man PL zur Darstellung von Wissen?

> Kurze Darstellung semantischer Netze (Bildchen reichte), Frames.

Was ist Vorteil (bzw. was ist Nachteil an PL)?

> Vererbung, Spezialisierung, Generalisierung, Beispiel dazu. PL kann man nur schwer dazu benutzen, komplexe Strukturen darzustellen.

Das war's.

Praktische Informatik bei Prof. Jarke
Pruefling: Ralf Biermanns
Note: 1.7
Datum: 17. Februar 1994
Themengebiet : BS (Silberschatz 1-9 + Fallbeispiel)
Compilerbau (Aho / Hopcraft / Ullman naja ich denke so bis semantische Analyse)
Datenbanken (bei Prof. Jarke selbst, meine Literatur war Silberschatz / Korth)

(Keine Frage womit ich beginnen will, dauert ihm anscheinend alles etwas zu lange, ich bin der 7. oder 8.te Pruefling an diesem Tag. Aber trotzdem draengt PJ nicht !)

PJ: Ja , .. Compilerbau (guckt so in seine Unterlagen). Da haben wir sowas wie regulaere Ausdruecke. Definieren Sie die.
Ich: Hingeschrieben, da"s induktiv aufgebaut alle Elemente aus Alphabet direkt in RA, dann alle Verknuepfungen durch | Konkatenation oder * (Kleensche Huelle).
PJ: Ja, .. wir haben den regulaeren Ausdruck (a|b)*abb wenn wir einen Scanner schreiben wollen, der die Worte dieser Sprache erkennt, wie machen Sie das ?
Ich: Klar , Scanner fuer reg.Sprachen ist Automat. Also reg.Ausdruck --> NEA --> DEA. (Ich dachte das reicht ihm).
PJ: Ja machen Sie mal.
Ich: (Bin hingegeangen und habe mit ueblichen Verfahren regulaeren Audruck umgewandelt, und habe darauf hingewiesen, da"s ich hier schon direkt reduziere, da soviele Epsilon Uebergaenge dabei sind. PJ nickt zustimmend. Denke dann das reicht und sage noch kurz wie man diesen NEA in einen DEA umwandelt, reicht ihm nicht, will er ebenfalls sehen mache also auch dies. (Mann das braucht unnoetige Zeit))
PJ: Gut, malen sie den resultierenden DEA hin. Kommen wir zur Syntaxanalyse. Da haben wir LL-Grammatiken. Wann ist eine Grammatik denn LL(1) ?
Ich: (Schluck, damit hatte ich nicht gerechnet (wg praktische Info), Es gibt zwei Definitionen, die eine mit Begruendung das Ableitungen nicht verschieden sein d"urfen und die andere mit first und follow. All das hatte ich als nicht sooo wichtig angesehen) Also da gibt es die eine Bedingung S --> alpha A beta --> Wort (oder so) dann ist bei LL(1) klar, da"s die Terminale da verschieden sein m"ussen.
Und es gibt noch eine andre Bedingung mit first und follow.
PJ: Geben Sie mal an
Ich: Grammatik ist LL(1), wenn fuer jedes Nonterminal A mit Produktionen A -> beta und A -> gamma gilt :
first(beta follow(A)) gescnitten mit first(gamma follow(A)) die leere Menge ergibt.
PJ: Wie berechnet man first.
Ich: Gebe kurze Erklaerung der Regeln (1. first von Terminal ist Terminal selbst, first von Nonterminal ist abh"angig von Produktionen dazu (dann die Spielerei mit Produktionen auf leerem Wort erklaert)
PJ: Jaaah. wie ist das denn so mit LL(1) (scheint nicht ganz zufrieden zu sein). Wenn ich eine Grammatik haben will, die z.B> 5+7*8 erkennen soll. Wie sieht die aus.
Ich: (Habe erkannt worauf er hinaus will, 1.) die Probleme die LL(1) Grammatiken haben koennen Linksrekursion und Linksfaktorisierung und das es Probleme bei + und * gibt) Erster Ansatz ist eine Produktion der Form:
exp --> exp * exp | exp + exp
Aber die gibt Probleme. Man geht dann hin und versucht die Mehrdeutigkeit und die Linksrekursion herauszunehmen.
PJ: Gut dann machen sie das mal hier !
Ich: (Mist, Linksrekursion ist kein Problem, aber die Sache mit den Prioritaeten, die wusste ich nur so ansatzweise, Versuche das zu erlautern, aber habe keinen grossen Erfolg, meine neue Grammatik ist dabei noch fehlerhaft)

PJ: Dann gehen wir mal zu DB. (Ich froh dass ich von CB weg bin, aber...) Sie haben ja bei mir(!) gehoert (Schitt) was koennen sie so zu Speicherstrukturen sagen ?

Ich: Angefangen bei seq. --> Indexseq. --> sparse und dense Index --> B und B* Baum und die Verkettung dabei. (PJ zufrieden, aber wollte nur auf B* hinaus)

PJ: Sie erwaehnten schon die Unterschiede bei B und B*-Baeumen, und das dies n-aere Baeume sind. Wie ist denn das kleinste n oder anders: Wieviele Soehne hat ein B*-Baum mindestens und was ist da das charakteristische.

Ich: 2... NEIN 3 aber nur zwei Indizes. Und n eigentlich so gewaehlt, da"s in Seite passt.

PJ: Ja malen sie mal den Aufbau eines solchen Knotens!

Ich: (Block gemalt , aufgeteilt, und dann P_i und K_i genannt, Ordnung aufgemalt)

PJ: Wenn wir jetzt diese kleine Knotengr"o"se nehmen, wie sieht das mit dem Verschmelzen und dem Aufteilen denn genau aus.

Ich: (kurz erl"auert, aber PJ nicht zufrieden)

PJ: Ich will es noch konkreter am besten an einem Beispiel. Nehmen wird die Zahlen 1,2,3,4,5,6 und f"ugen Sie sie ein.

Ich: (Ein bischen gehangen, da ich den genauen algorithmischen Ablauf auch nicht so wu"ste, kriegs aber noch hin, da ich wu"ste der Baum ist balanciert und geordnet. Hatte ihn auch etwas verwundert, da ich bei einem Ausgleichsvorgang anscheinend einen verbessernden Ansatz gew"ahlt hatte (Wiegesagt: Ich kannte den Alg. nicht!)

PJ: Nun wir haben eigentlich nicht sehr viel Zeit, das Problem, welches ich den anderen gestellt hatte ist mir jetzt zu aufwendig (Er meint das Spielchen mit der Patientenrechnung (siehe Stefan Vaillant) Schade, das hatte ich mir angeguckt, wusste wo da die Probleme liegen) Stellen Sie sich vor wir haben einen Patienten mit Name, Nr, Datum und Krankheiten (Jarke ueberlegt sehr lange dazwischen, weil er das so gerade aus der Hand schuettelt und keine Fallen einbauen moechte) Krankheiten sind ebenfalls mit einer Nr und dem Namen attributiert. und Patienten koennen Krankheiten haben, wobei wir das erste Erkennen der Krankheit durch ein Datumsattribut festhalten.

Ich: Also ER ?? (Ja, male ER-Diagramm, war sehr einfach, wunderte mich innerlich) Zwei Entities mit Schluessel Nr., Relationsship "istErkranktAn" war m:n und mit Attribut Datum versehen.

PJ: Schoen das bitte jetzt transformieren, Wie geht man allgemein vor wie hier ?

Ich: Erklaere allgemein, was man sehen kann und was man bei 1:n Beziehungen machen kann. Gehe dann konkret vor (Hier war aber nur eine mit m:n, hatte hier jetzt Schwierigeres erwartet, sowas wie NF und Kalkuele / Algebra, aber was solls :)

```
Patient(Nr, Name, GDatum)
Krankheit(Nr, Name, Arznei,...)
IEARel(PNr,KNr, Datum)
```

PJ: Machen sie mal eine Anfrage mit SQL, wobei die Patientennamen und Krankheitsnamen von Patienten mit Datum 1980 ausgegeben werden sollen und eine, die die Anzahl der Personen mit Schupfen ausgibt.

Ich: SELECT Patient.Name Krankheit.Name
FROM Patient,IEARel,Krankheit
WHERE Patient.Nr = IEARel.PNr and IEARel.KNr = Krankheit.Nr and
Patient.Datum = 1980;

(Brabbel dabei von Was was ist bei SQL (SELECT Projektion, WHERE Selektion und FROM kart.Produkt) und von nat.Join. Bei der zweiten Anfrage hatte ich schon an eine GROUP BY Anfrage gedacht, doch so kompliziert wars nicht)
Erste Loesung (war identisch zu oben, hab ich hingeschrieben ist aber zu teuer)

```
SELECT count(Patient.Name)
FROM Patient,IEARel,Krankheit
```

Prüfungsprotokoll

Prüfer: **Prof. Jarke**

Fächer: **EDB** (Einführung Datenbanken)
OODB (Objektorientierte Datenbanken)
RE (Requirements Engeneering)

Datum: 08.07.1999
Note: 1,3

Allgemeines zur Prüfung:

Jarke ist ein ruhiger Prüfer. Die Protokolle (auch die alten), die es in der Fachschaft gibt, spiegeln die Atmosphäre und die Fragen ganz gut wieder.
Allerdings war es bei mir so, daß Jarke fast bei jeder zweiten Frage das Fachgebiet gewechselt hat. Durch diese permanenten Sprünge wusste ich sehr häufig nicht, ob er ein bestimmtes Gebiet jetzt vertiefen oder den Übergang zu einer anderen Vorlesung herstellen wollte. (Aus dem Grund ist das Protokoll auch nicht chronologisch sondern thematisch geordnet.)
Zudem waren seine Fragen grausig. Er fragt derart schwammig, daß ich permanent nachfragen musste und erst nach zwei bis drei Anläufen wusste, worauf er hinaus wollte. Das lässt einen ziemlich in der Luft hängen. Schließendlich hatten diese Mißverständnisse aber wohl keinen Einfluß auf die Note.

Allgemeines zur Vorbereitung:

EDB: man sollte sich nicht durch den Folien-Wust verunsichern lassen. Einfach anfangen die Teilgebiete zu lernen, der Zusammenhang kommt dann mit der Zeit !!
OODB: ist superschnell gelernt, vor allem, wenn man sich schon ein bischen mit oo-Programmierung auskennt (z.B. mit C++)
RE: vor allem UML ist eine Sache von ca. einem Tag ! Die wichtigen Stichworte von RE1 kann man an 10 Fingern abzählen.
Jarke prüft Vorlesungen, die er selbst nicht gelesen hat relativ oberflächlich ab. Er fragt Konzepte und Zusammenhänge, mehr schon fast nicht, vor allem keine syntaktischen Details (Sollte man evtl. trotzdem wissen und nebenbei erwähnen).

Allgemein habe ich ca. 3 Wochen teilweise recht intensiv gelernt. Die Vorlesungen habe ich bis auf RE1 nicht gehört!

Quellen:

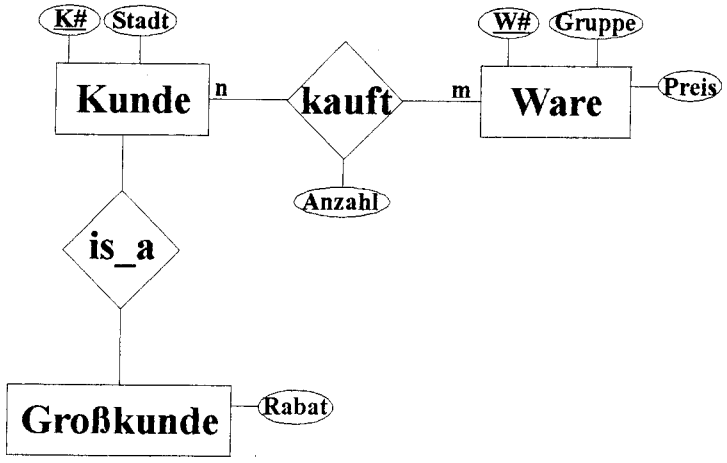
EDB: Folienkopien d. Vorlesung / Vossen "Datenmodelle ..." (zum Nachschlagen)
OODB: Kemper/Moerkotte "OODB" (Relevant: Kap. 8-15)
Kemper "Grundlagen OODB" (30 Seiten Super-Überblick)
RE1: Folienkopien d. Vorlesung
RE2: Oestereich "Objektorientierte Softwareentwicklung mit UML"

RE:

Jarke: Man hat 3 klassische Modelle beim RE, welche ?
Ich: Verhaltens-, Daten- und Funktionsorientierte ... (erklärt + Bsp)
Jarke: Was ist strukturierte Analyse (SA) ?
Ich: DFD, DataDict, MiniSpecs ... (Notation aufgemalt + erklärt)
Jarke: Integritätsbedingungen bei SA ?
Ich: Sichtbare-, Dictionary-, Datenspeicherbalancing (erklärt)
Jarke: Übersetzung SA->ER ?
Ich: (nicht gewusst, rumgedruckst)
Jarke: Informationen für ER-Diagramm stehen alle im DataDictionary
Was ist UML, was macht man damit ?
Ich: (Historie erklärt, Unterstützung bei versch. Softwarephasen erklärt, Diagramme aufgezählt und teilweise auch grob aufgemalt.)
Jarke: Klassendiagramme ?
Ich: (Notation detaillierter erklärt und teilweise hingemalt; erwähnt, daß es bei der Übersetzung von ER->OODB mehrere Möglichkeiten der Realisierung geben kann !)

EDB:

Jarke: Was ist ER-Diagramm ?
Ich: (Beispiel überlegt und hingemalt + erklärt; später enstand daraus mit Jarkes Anmerkungen folgendes ER-Diagramm:)



Jarke: Was ist "is_a" ?
Ich: Erbt alle Attribute und Beziehungen, Großkunden ist Teilmenge von Kunden
Jarke: Relationenschema von ER-Diagramm erstellen
Ich: Kunde(K#, Stadt)
Ware(W#, Gruppe, Preis)
kauft(K#, W#, Anzahl)
Großkunde(K#, Rabat)
Jarke: Leichte oder schwierige Anfrage ? ... Schwierige (hehe) Anfrage mal versuchen: folgende Tabelle aufstellen: "Stadt - Umsatz_pro_Stadt"
Ich: SELECT Kunde.Stadt, SUM(kauft.Anzahl * Ware.Preis)
FROM Kunde, kauft, Ware
WHERE Kunde.K#=kauft.K# AND kauft.W#=Ware.W#
GROUP BY (Kunde.Stadt) [ORDER BY (Kunde.Stadt) desc/inc]
Jarke: Was sind Normalformen ?

Ich: (erklärt)
Jarke: In welcher NF sind die aufgestellten Relationen ?
Ich: 3NF ... (alle Beziehungen geprüft) ...
BCNF ? (alle NoKey->Key -Beziehungen geprüft) BCNF ! (Jarke nickt)
Jarke: Nachteile Normierung ?
Ich: Fragmentierung der Informationen (erklärt). Muß aber sein wegen Anomalien (erklärt)
Jarke: Was sind nun die Nachteile ?
Ich: (mit Jarkes Hilfe) Zusammensetzen der Fragmente mit Join ist teuer !

OODB:

Jarke: Nachteile der Normierung ... warum nicht bei Objekten ?
Ich: Da kommt das nicht vor. (Erzählt + erklärt) Objekte, Struktur, Verhalten, Verweise auf andere Objekte eindeutig durch OID (das wollte er wissen) ... Zugriff (Zusammenbauen der Informationen) durch Zugriff entlang des Zugriffspfades als funktionaler Join =billig.
Jarke: Was ist Polymorphie ?
Ich: AdHoc-, Inklusions- und bounded-Polymorphie (erklärt+Bsp)
Jarke: Problem der Vererbung (einfach) ?
Ich: Kovarianz (=schlecht)/Kontravarianz(=gut). Kontravarianz: Spezialisiertes Objekte darf in geerbter und überschriebener Methode nur generalisiertere Argumente benutzen =>Typsicherheit

Viel Erfolg

Gedächtnisprotokoll Professor Jarke

Fach	Vertiefungsgebiet: Informationssysteme
Datum	Juni 1994
Dauer	ca. 45 Minuten
Vorlesungen	Einführung in Datenbanken (Jarke) Implementierung von Datenbanken (Kemper) Objekt–orientierte Inf.verwaltung f. ingenierwis. Anwendungen (Kemper) Verteilte Datenbanken & Interoperabilität (Jeusfeld)

Einf.i.DB

Sie kommen in einen Betrieb und sollen eine DB entwerfen, was können sie da machen?

Fragen der zukünftigen Benutzer oder Ansehen schon bestehender Lösungen in diesem oder anderen Unternehmen

Modellierung des folgendes Flugabrechnungszettels als ER-Diagram:

Fluggesellschaft Adresse					Datum
Flug#	von	nach	Preis	Anzahl	Einnahmen
...	...	...	...	...	...
					Gesamteinnahmen

Mit 4 Entitäten und 3 Relationships modelliert. (Es geht auch mit nur 3 Entitäten und 2 Rel.ships, wenn man einem Rel.ship ein Attribut anhängt.)

Relationales Schema dazu

(beachte: 4 Relationen reichen, berechnete Attribute, künstliche ID# als Primärschlüssel)

Anfragen auf diesem Schema: Wie berechnet man die Einnahmen pro Flug (also die Spalte Einnahmen in der obigen Tabelle)

*SELECT-Anfrage mit Join und berechnetem Attribut (in der SELECT-Zeile) F.Preis * B.Anzahl hingeschrieben*

Und wenn man nun die Gesamteinnahmen je Fluggesellschaft haben will?

*Wieder ein Join diesmal über 3 Relationen; SELECT-Zeile mit berechnetem Attribut SUM(F.Preis * B.Anzahl) und an das ganze wird noch ein GROUP BY A.Fluggesellschaft angehängt*

Implementierung von DB

Wie implementiert man sowas?

Mit einem DBMS! (Die Frage war mir dann doch etwas allgemein)

Wie genau? Schichtenarchitektur?

oberste Schicht: Mengenorientierte Schicht; Aufbau des Anfragebaums; Optimierung des Anfragebaums mittels Runterziehen von Selektion; Selektion & Kreuzprodukt ⇒ Join, ...

Was passiert mit dem Anfragebaum in tieferen Ebenen

Implementierungsmöglichkeiten des Joins: nested loop; nested loop mit erweitertem Pufferspeicher; nested loop unter Verwendung eines Indexes; Sort–Merge–Join; Hash–Join

Was passiert dabei wo in den tieferen Ebenen?

Erzählt

Wofür eigentlich Segmentschnittstelle & Dateiverwaltungsschnittstelle, wo es sie doch schon im BS Virtual Memory, Cache und Filesystem gibt?

Dateiverwaltungsschnittstelle: außer Portabilität nur Nachteile (Umrechnung Datei → Platten–Block im BS notwendig)

Segementschnittstelle: bessere Information über Inhalt der Seiten; eigene Ersetzungsstrategien

(von der Antwort bin ich nur begrenzt begeistert; fehlt zumindest noch: eigenständige Kontrolle über Puffergröße und Entscheidung über STEAL– und FORCE–Eigenschaften)

Wo im Schichtenmodell liegt die Transaktionsverwaltung?

Parallel zu Ebenen I_2 bis I_5; Gründe dafür

Wie implementiert man denn diese Aufteilung auf verschiedene Schichten?

Wußte ich nichts genaues zu, habe dann was über Mehrschichten–Transaktionen erzählt

Recovery: Arten?

R1 bis R4 jeweils mit Beschreibung

Wie implementiert man R3?

Physische (R3: before–images) und logische Protokollierung; Vor–/Nachteile; Arten von Sicherungspunkten; Verwendbarkeit für R3

Verteilte DB

Was wurde denn in der Vorlesung gemacht?

Äh?!?! Vorteile/Nachteile aufgezählt

Integration heterogener DB

2 Varianten bei vert. DB:

1) globales Schema gegeben ⇒ Fragmentierung ⇒ Allokation

2) lokale Schemata gegeben, globales gesucht; (← heterogene Datenbanken)

Vorgehen und Probleme: siehe Manfreds Vorlesung

Was kann man machen, wenn man aufgrund der Probleme kein globales Schema erhält?

??? Was über Interoperabilität u.ä erzählt; DB Ansatz, PS Ansatz

(Wer weiß was er da wirklich hören wollte)

OODB

Also ich als Anwender war da gerade auf einem Kongreß, wo mir erzählt wurde, OODB wären was ganz tolles. Warum haben sie das Flugabbrechnungsbeispiel nicht objektorientiert modelliert?

Vor-/Nachteile aufgezählt; bei kaufmännischen Anwendungen eher nicht sinnvoll

Modellierung des Ganzen (Er wollte eine Definition sehen)

GOM Modellierung als ein Objekt (auf ausdrücklichen Wunsch Jarkes; normalerweise furchtbare Anomalitäten)

er wollte wohl hauptsächlich auf die Methoden für die berechneten Attribute heraus, das habe ich nur nicht mitgekriegt

Allgemein: Wahrscheinlich ist es nicht so sonderlich vorteilhaft, wenn man von den 4 Vorlesungen über die man sich prüfen läßt, nur eine beim Prüfer gehört hat. Der Prüfer weiß nicht so genau was er fragen soll, und man selbst weiß nicht, worauf er eigentlich hinaus will.

Gedächtnisprotokoll einer Diplomprüfung
Praktische Informatik

Prüfer:	Prof. Jarke
Beisitzer:	Peter Haumer
Fächer:	Einführung in Datenbanken Betriebssysteme Expertensysteme
Datum:	11.04.1995
Dauer:	14.05-14.50 (45 min)
Note:	1,3

Betriebssysteme (20 min)

Beginnen wir mit dem Short-term-scheduler. Was macht er, welche Strategien gibt es?
Ich habe angefangen mit Prozessen, die in den Zuständen ready, running und waiting sein können. Die Prozesse im Zustand waiting stehen in der CPU-Queue. Die Aufgabe des Schedulers ist hier, einen geeigneten Prozeß auszuwählen. Geeignet heißt hier im Bezug auf ein Kostenmaß, z. B. kürzeste Wartezeit. Für die Auswahl gibt es verschiedene Strategien, die ich dann erklärt habe: FIFO, SJF, RR.

In der Diskussion zum Short-term-scheduler wurde auch der Long-term-scheduler angesprochen. Was können Sie zu diesem Thema sagen?
Durch ihn soll ein geeignetes Prozeß-Mix (Verhältnis von IO- und CPU-Bound) und ein geeigneter Grad von Multiprogramming eingestellt werden.

Wie kann dieser Grad ermittelt werden?
Das passiert mit Hilfe der WorkingSets, die das Lokalitätsprinzip ausnutzen. Man betrachtet die letzten n Seiten, auf die zugegriffen wurde. Diese Menge wird vom Prozeß benötigt, um ohne Thrashing zu laufen.

Paging: Wie sieht ein Page-fault im Detail aus?
Bei einem Page-fault wird ein Interrupt ausgelöst, die Kontrolle wird also an das Betriebssystem übertragen. Hier wird zuerst der Prozessorstatus in den PCB gespeichert, also der PC vom Stack und die CPU-Register. Der Prozeß kommt dann in den Zustand waiting. Das BS holt sich dann die gesuchte Seite und gibt die Kontrolle zurück an den Scheduler.

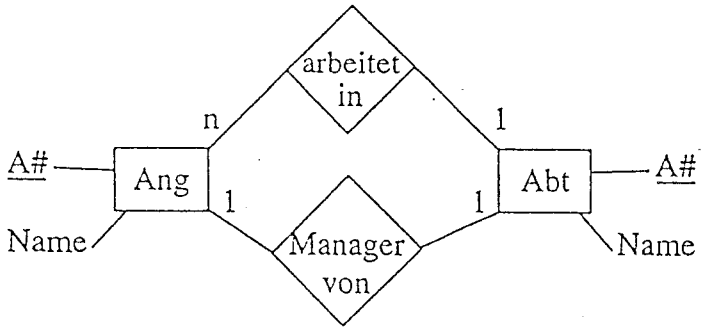
Datenbanken (15 min)

Da wir gerade über BS gesprochen haben, betrachten wir jetzt die unteren Schichten einer DB. Welche Datenstrukturen kennen Sie und welche Arten von Datenzugriffen werden unterstützt?
Ich habe einige Verfahren aufgezählt und erklärt: sequentiell, index-sequentiell, Index (dense und sparse), B- und B*-Bäume, Multilisten und mehrfach verkettete Bäume.

Die Frage nach einem Beispiel, bei dem sequentielle Abspeicherung gut funktioniert, habe ich etwas falsch verstanden.
Ich habe nach einem konkreten Anwendungsbeispiel gesucht, mir ist aber nichts passendes eingefallen. Die gewünschte Antwort war: Bei Zugriffen der Art SOME (10% - 99% aller Datensätze) und ALL (100%).

Eine wichtige Struktur fehlt noch, welche ist das?
Nach einigem Überlegen bin ich auf das Hashing gekommen. Hier habe ich die verschiedenen Arten der Konfliktbehandlung angesprochen.

Nehmen wir an, wir haben folgende Relationen gegeben:
Angestellter (A#, Name, Abt#) und Abteilung (B#, Ort, Mgr#)
Ein Angestellter arbeitet in einer Abteilung, diese hat wiederum einen Manager. Wie sieht das zugehörige ER-Diagramm aus?



Wenn man jetzt wieder versucht, daraus Relationen zu erstellen, wie gehen Sie da vor?
Aus jeder Entität wird eine Relation, außerdem wird jede Beziehung auf eine Relation abgebildet. Da es sich hier um 1:n- und 1:1-Relationen handelt, die immer vorhanden sind, werden diese an die entsprechenden Entitäten-Relationen angehängt. Wir erhalten also obiges Ergebnis.

Formulieren sie hierzu die SQL-Anfrage, die die Namen aller Angestellten, die in Düsseldorf arbeiten, ermittelt.
Beim Aufschreiben habe ich noch ein wenig die Abbildung von relationaler Algebra auf SQL erläutert.
SELECT Name
FROM Ang, Abt
WHERE Abt# = B# AND Ort=Ddorf

Wie sieht die Anfrage im TRK aus?
$\{n \mid \exists a \in Ang\ \exists b \in Abt : a.Abt\# = b.B\# \wedge b.Ort = Ddorf \wedge n = a.Name\}$

Und im DRK?
$\{n \mid Ang(A\#, n, Abt\#) \wedge Abt(Abt\#, Ddorf, Mgr\#)\}$

Expertensysteme (10 min)

Die obige Anfrage sieht ja fast aus wie Prolog. Wie schreibt man das konkret hin?
düsseldorfer(Name) :- ang(A#, Name, Abt#), abt(Abt#, Ddorf, Mgr#).
Hierbei kann A# und Mgr# durch ein _ (anonyme Variable) ersetzt werden.

Angenommen, wir haben ein Prädikat dVorg(A, V), das zu einem Arbeitnehmer den direkten Vorgesetzten berechnet. Schreiben Sie eine Regel, die alle (auch indirekten) Vorgesetzten ermittelt.
vorg(A, V) :- dVorg(A, V).
vorg(A, V) :- vorg(A, X), dVorg(X, V). (Wird später noch berichtigt.)

Fach	Praktische Informatik
Prüfer	Prof. Jarke
Gebiete	Datenbanken (Vorlesung Jarke)
	Betriebssysteme (Peterson/Silberschatz)
	OODB (Vorlesung Kemper)
Datum	1.Quartual 94
Note	1.3

Er begann mit der Frage, mit welchem Gebiet ich denn gerne Anfangen möchte. Auf meine Antwort "Datenbanken" hin haben wir dann mit diesem Gebiet begonnen.

Datenbanken (30min)

- Jarke legte mit eine (skizzierte) Rechnung eines Arztes vor. Form war ca.:

```
Patient-Nr.
Patient-Name
Geburtsdatum

                                        Rechnung-Nr
                                        Rechnung-Datum
                                        Diagnose

Behandlungs-    Massnahme    P/T    Preis
datum
-----------------------------------------
...             ...          ...    ...
...             ...          ...    ...
...             ...          ...    ...
...             ...          ...    ...
===             ===          ===    ===
                                    Betrag
```

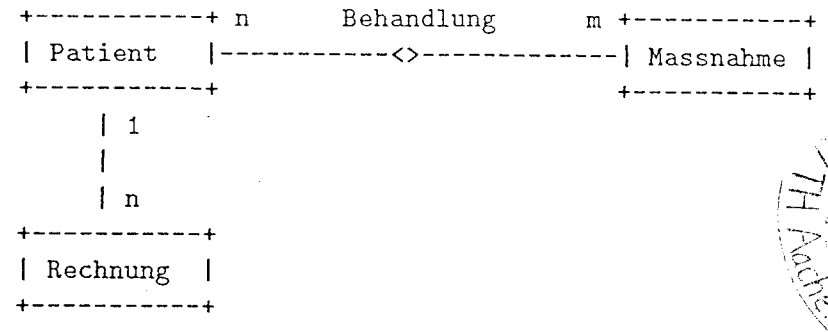
Er hat dann noch erklaert: die Rechnung-Nr ist eindeutig (also keine eigene Nummerierung pro Kunde), An einem Tag pro Patient nur eine Behandlung, P/T bedeutet irgendwas mit Persoenlich oder technische Behandlung und noch irgendwas.

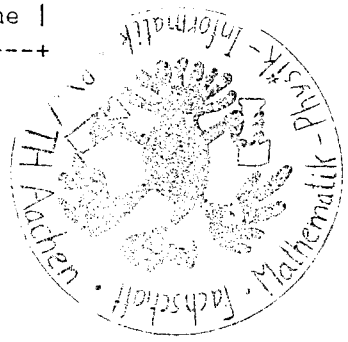
Nach ca. 3 Minuten dann die erste Frage: Der Arzt will diese Rechnungen mit dem Computer erstellen. Wie gehen sie denn so ran?

Ich: ER oder Universelle Relation mit Funktionalen Abhaengigkeiten

OK, machen wir beides. Erst mal ER bitte.

Ich: Obwohl ich eigentlich der ER-Profi bin, hatte ich bei dem Beispiel dann doch meine Probleme. Nach einer kurzen Diskussion mit mir selbst, ob ich jetzt Massnahme oder Behandlung als Entity machen soll, haben ich mich fuer ersteres entschieden und gelangte dann zu:





An die Behandlung hatte ich das Datum gehaengt.

Damit war Jarke aber nicht zufrieden. Ich habe dann noch was erzaehlt, bis Jarke dann mit seiner Loesung kam: Die n-Seite der Behandlung sollte ich nach Rechnung legen.

Ich habe dann noch mit ANSI/Sparc konzeptionelle Ebene und View geredet, das naemlich meine Loesung auf der konzeptionelle Ebene ist und seine auf der View Ebene, aber Jarke wollte dann doch seine haben. Wenn ich wirklich nur die eine View-Ebene beachte hat Jarke wohl auch Recht.

OK, jetzt mal die andere Moeglichkeit.

Ich habe dann die Attribute hingeschrieben, habe eine paar Abhaengigkeiten eingetragen und da schon mal 2./3.NF erklaert. Ich sollte dann noch alle eintragen und dann Dekompositionieren. Evtl. hat sich da noch ein Fehler eingeschlichen.

Jetzt nehmen wir mal nur die Relationen Patient und Rechnung (wg. $1:n$ gibt es fuer den Relationship ja keine eigene Relation). Neben einer trivialen SQL-Anfrage mit einer Join-Bedingung hat er dann nach einer Tabelle gefragt in der fuer jede Diagnose die Einnahmen im Jahr 1990 stehen.

Ich: Also eine GROUP BY aufgemalt.

OODB (10min)

Dieser Teil verlief unstrukturiert. Ich hatte den Eindruck, Jarke wusste selber nicht was er eigentlich Fragen wollte bzw. auf was er hinnauswollte. Ich habe mal so ein paar Fragen zusammengefasst:

- Wie modellieren sie denn Patient in GOM?
- Wenn ich jetzt noch Kassen und Privatpatienten habe, wie geht das?
- (Wohl Standardfragen) Was muss ich bei der Verfeinerung einer Operation denn beachten?

Meine Antworten: Unterscheid ER/OODB ein bisschen motiviert (explizite Identitaet = keine Patienten Nummer, Kapselung), Erstellung der Rechnung als Operation, Ko-/Kontravarianz, virtuelle Methoden, Polymorphie, late-binding usw.)

Ich glaube, Jarke konnte mit nicht immer folgen, i.b. bei schien er die Begriffe Ko-/Kontravarianz nicht zu kennen.

Betriebssysteme (10min)

- Scheduling bei time-sharing Systemen?
- Virtuelle Speicherverwaltung?
- Trashing, wie kann man es verhindern?

Bei RR-Scheduling wollte er genau wissen, wie das funktioniert (Hardware Timer und Interrrupt)

Prüfungsprotokoll
Diplomprüfung: Praktische Informatik
Prüfer: Prof. Jarke

Themen: Einführung in Datenbanken (Jarke)
Implementierung von Datenbanksystemen (Kemper)
Betriebssysteme (Silberschatz, Peterson, Galvin)

Referenzen:
Datenbanken: *Fundamentals of Database Systems* (Elmasri, Navathe) für Einführung in Datenbanken und Implementierung von Datenbanksysteme (Anfrageoptimierung).
Datenbankhandbuch für Implementierung von Datenbanksystemen (Schichtenmodell und Transaktionsverwaltung)
Verteilte Datenbanksysteme (Bayer, Elhardt, Kießling, Killar in Informatik-Spektrum, 1984, Nr 7) für Implementierung von Datenbanksysteme (Verteilte Datenbanken)
Principles of Database and Knowledge-Base Systems (Ullman) für Einführung in Datenbanken (Deduktive DB und Datalog) und Implementierung von Datenbanksystemen (Deduktive DB)
Betriebssysteme: *Operating System Concepts*, 3rd Edition (Silberschatz, Peterson, Galvin)

Datum: 21.04.1993
Note: 1.0

Einführung in Datenbanken

Was für Sprachen gibt es im relationalen Modell?
Theoretisch: Relationale Algebra (operational) und relationales Kalkül in Form von Tupelkalkül und Domänenkalkül (deklarativ)
Praktisch: SQL, QUEL (Tupelkalkül) und QBE (Domänenkalkül)

Was für eine Grundstruktur hat eine SQL-Anfrage?
SELECT...FROM...WHERE... mit einem Beispiel, aus einer Relation, hingeschrieben und Bedeutung der einzelnen Teile erklärt.

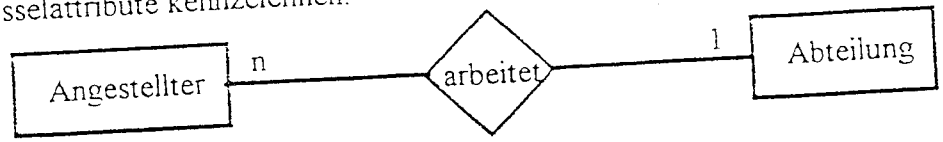
Welchen Operatoren der relationalen Algebra entsprechen die einzelnen Teile der SELECT... FROM...WHERE...-Anweisung?
SELECT entspricht Projektion (π), WHERE entspricht der Selektion (σ) und FROM der Auswahl der zu bearbeitenden Relationen (Argumente).

Übertragen Sie die SQL-Anweisung in einen Ausdruck der relationalen Algebra.
Relationenalgebra-Ausdruck hingeschrieben und währenddessen meine Ausführung der vorherigen Frage noch mal erläutert. Irgendwie löste dies alles immer noch keine vollständige Zufriedenheit aus...

Das ist ja nun ein sehr spezielles Beispiel, der FROM-Teil hat in der Regel ein allgemeineres Aussehen...

...mehrere Relationen, anstatt nur einer.

Na gut, nehmen wir mal an wir hätten Entity-Relationship-Diagramm mit zwei Entitäten Angestellter und Abteilung und einer n:1-Beziehung. Sowohl Entitäten als auch Beziehung haben Attribute. Schlüsselattribute sind Name für Angestellter und AbtNr für Abteilung. Wie kann man dieses Diagramm durch Relationen darstellen?
Entity-Relationship-Diagramm aufmalen, Kardinalitäten eintragen, Attribute einzeichnen und dabei Schlüsselattribute kennzeichnen.



Dann die beiden Relationenschemata hingeschrieben.

So, jetzt formulieren sie auf diesen Relationen die SQL-Anfrage: Name der Angestellten die älter als 56 sind und in der F&E-Abteilung beschäftigt sind.
Also die SQL-Anfrage lautet

SELECT *Angestellter.Name*
FROM *Angestellter, Abteilung*
WHERE *Angestellter.Alter* > 56 AND *Abteilung.Name* = *F&E*

Welche Entsprechung hat das FROM denn nun in der relationalen Algebra?
Verbund (⋈) oder kartesisches Produkt.

Wenn nun die 1:n-Beziehung in eine m:n-Beziehung umgewandelt wird, d.h. ein Angestellter kann in mehreren Abteilungen arbeiten, dann kann man diese beiden Relationen erstmal weiter verwenden, um diese Beziehung auszudrücken. Was passiert, wenn man das macht?
Es entstehen Redundanzen in der einen Relation (Angestellter), da mehrere Tupel für einen Angestellten eingefügt werden müssen, die jeweils unterschiedliche Abteilungsnummern enthalten.

Und was muß bei der Relation Angestellter daher geändert werden?
Der Primärschlüssel ist nicht mehr nur Name sondern zusätzlich noch AbtNr..

Was ist denn nun das Problem der Redundanzen?
Durch Redundanzen entstehen Anomalien (Einfüge-, Lösch-, Änderungsanomalie).

Was ist ein Beispiel einer Löschanomalie?
Ich habe mir hier ein beliebiges Beispiel, d.h. nicht obiges, herausgesucht und daran die Löschanomalie erläutert.

Jetzt haben wir also Anomalien und Redundanzen und was macht man dagegen?
Man beseitigt die Redundanzen indem die Relation durch Transformation in Normalformen dekomponiert wird.

In welcher Normalform ist denn die obige Angestellten-Relation und in welcher nicht, d.h. in welche muß man sie als nächstes überführen?
Die Relation ist in erster Normalform, aber nicht in zweiter, daher muß sie erstmal in die zweite Normalform überführt werden.
Was heißt denn zweite Normalform, wo verletzt die Relation die zweite Normalform und wie sieht die Dekomposition der obigen Relation aus?

Also habe ich die Geschichte mit der funktionalen Abhängigkeit, der vollen funktionalen Abhängigkeit und der zweiten Normalform erzählt. Danach habe ich dann erklärt wo die zweite Normalform verletzt wird und die Relation entsprechend aufgeteilt.

(Völlig ansatzlos) *Moment mal, die SQL-Anfrage da oben ist ja noch gar nicht vollständig, machen Sie das doch mal eben.*
Wie nicht vollständig? Nach einigem hin und her wurde mir dann klar, daß ich bei der SQL-Abfrage mit den beiden Relationen die JOIN-Bedingung vergessen hatte und bisher noch das kartesische Produkt berechnet wurde, auf das dann die Selektionsprädikate angewendet wurden. Also habe ich die Anfrage zu

SELECT *Angestellter.Name*
FROM *Angestellter, Abteilung*
WHERE *Angestellter.Alter* > 56 AND *Abteilung.Name* = *F&E*
AND *Abteilung.AbtNr* = *Angestellter.AbtNr*

ergänzt.

Gut, zurück zu den Normalformen. Welche Bedingungen sollten die Relationen nach der Normalisierung haben?
Nichtadditiver-Verbund (lossless-join) und Abhängigkeitserhaltung (dependency preservation). Dann habe ich diese Begriffe anhand eines abstrakten Relationsschemas erklärt. Dabei fällt unwillkürlich der Begriff des Abschlusses einer Menge funktionaler Abhängigkeiten.

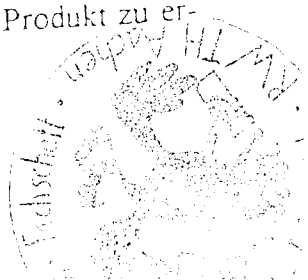
Was ist denn der Abschluß einer Menge funktionaler Abhängigkeiten? → wofür gut
Alle funktionalen Abhängigkeiten die aus der Menge abgeleitet werden können. Im Zusammenhang damit die Armstrong-Axiome erwähnt und aufgeschrieben.

Implementierung von Datenbanksystemen und Betriebssysteme

Welche mögliche Architektur existiert, wenn ein Datenbanksystem implementiert werden soll....
...Schichtenmodell...

...nee, das wird irgendwie zu langweilig, wenn Sie die ganzen Schichten erklären müssen. Stellen Sie sich mal eine SQL-Anfrage vor die in das Datenbanksystem gesteckt wird und dann über die verschiedenen Schichten gereicht werden muß, was passiert denn da in jeder Schicht?
Also doch Schichtenmodell. Die erste Schicht ist mengenorientiert, dort startet Anfrage und wird in entsprechenden Relationenalgebra-Ausdruck übersetzt. Dann wird Zugriffskontrolle für die entsprechenden Daten ausgeführt und die Überprüfung von Integritätsbedingungen durchgeführt. Dann erfolgt die Anfrageoptimierung.

Wie optimiert man denn da?
Es gibt da so'ne Methode mit Anfragegraphen, die basiert auf dem Tupelkalkül, da wir jedoch hier bereits einen Ausdruck der Relationenalgebra haben ist es sinnvoll algebraisch zu optimieren. Bei der algebraischen Optimierung ist Maß für die Güte der Optimierung die Größe der zu verarbeitenden Relationen bzw. die Anzahl der Tupel. Dazu wird Operatorbaum erzeugt und dort nach heuristischen Regeln die Operatoren der Relationenalgebra vertauscht. Dabei ist es wichtig Projektion und Selektion möglichst weit zu den Blätter zu verlagern. Verbund so weit wie möglichst nach oben zu propagieren und das kartesisches Produkt zu ersetzen.



Warum wird der Verbund denn soweit wie möglich nach oben propagiert?
Weil er eine sehr teuere Operation ist.

Wie teuer ist er denn und welche Realisierungen gibt es für den Verbund?
Falls zwei Relationen R und S die Argumente des Verbunds sind gilt: Falls naive Realisierung (*nested-loop*) gewählt wird ist der Aufwand |R|*|S|. Beim *Sort-Merge-Join* müssen die Relationen sortiert sein, falls dann nach dem Schlüsselattribut 'ge-joint' wird, reduziert sich der Aufwand auf das einmalige Durchlaufen jeder Relation. Optimierung durch Hash-Join, bei dem die Join-Attribute durch eine Hashfunktion auf einen Behälter abgebildet werden und dann im entsprechendem Behälter gesucht werden kann (Der Vorteil ist, daß Relationen nicht sortiert sein müssen).

Also dann weiter mit den Schichten.
Unter der mengenorientierten Schnittstelle liegt die externe Satzschnittstelle, diese entspricht im wesentlichen dem Netzwerkmodell und liefert logische Zugriffsstrukturen auf die Datensätze. Diese Schnittstelle wird in der Regel dann weggelassen, wenn die mengenorientierte Schnittstelle auch noch realisiert wird (Perfomance-Gründe). Unter der externen Satzschnittstelle liegt die interne Satzschnittstelle, die die physischen Zugriffsstrukturen...

...nennen Sie mal welche...
...B-Baum, B*-Baum, dynamisches Hashing...

...o.k. weiter...
und die Satzadressierung realisiert. Darunter liegt die Segmentschnittstelle und die Systempufferverwaltung. Die Segmentschnittstelle stellt einen virtuellen linearen Adreßraum, der in Seiten fester Größe aufgeteilt ist, zur Verfügung. Die Systempufferverwaltung stellt den Systempuffer zur Verfügung und sorgt für das Laden von angeforderten Seiten.

Da die Zeit nun doch etwas knapp wird, und es hier gut paßt eine Frage zu Betriebssystemen: Was passiert denn nun beim paging und virtueller Adressierung, bzw. was für Probleme können auftreten (Bezug zur Systempufferverwaltung eines Datenbanksystems) und was sind mögliche Lösungen?
Wenn zuviele Prozesse bzw. Transaktionen zugelassen werden ist das System irgendwann, aufgrund der hohen *Page-Fault-Rate*, nur noch damit beschäftigt Seiten vom Sekundärspeicher zu holen (*Thrashing*). Um das zu verhindern, muß der *Long-Time-Scheduler* dafür sorgen, daß nicht zuviele Prozesse geladen werden (Begrenzung der Parallelität) und die Anzahl der Seiten die ein Prozeß im Hauptspeicher benötigt muß sinnvoll approximiert werden (*Working-Set*).

Wenn das alles gemacht wird, was kann denn noch passieren, wenn eine Seite in den Hauptspeicher geladen werden soll?
Es soll eine Seite geladen werden, jedoch existiert kein freier Hauptspeicherrahmen bzw. Pufferrahmen. Dann muß nach einer gewählten Seitenersetzungsstrategie die Seite ausgewählt werden die ersetzt werden soll. Mögliche Strategien sind: FIFO (schlecht, da Lokalität nicht berücksichtigt wird), LFU (schlecht, da anfangs häufig referenzierte Seiten nicht mehr verdrängt werden), LRU, CLOCK (heißt in Betriebssystemen *second chance*). Diese Verfahren werden meist von der Hardware unterstützt und finden daher bei Betriebssystemen oft Anwendung, da bei Datenbanksystemen die Systempufferverwaltung sowieso softwaremäßig implementiert wird, können Seitenersetzungsverfahren auch softwaremäßig implementiert werden, um so höhere Flexibilität zu erreichen (Bsp.: GCLOCK als Variante von CLOCK jedoch mit Referenzzählern und Seitengewichten)

Was kommt nun nach der Segmentschnittstelle und was wird bei diesem Übergang wie abgebildet?
Als nächstes folgt die Dateischnittstelle die hauptsächlich aus den Ressourcen des Betriebssystems für die Dateiverwaltung besteht. Die Aufgabe besteht nun darin die Segmente auf Dateien abzubilden und die Seiten der Segmente auf die Blöcke der Datei.

Wenn nun die Daten der Datenbank bearbeitet werden sollen, womit geschieht die Bearbeitung und welche Probleme entstehen?
Die Bearbeitung der Daten erfolgt durch Transaktionen. Da mehrere Transaktionen gleichzeitig (verzahnt) auf dem System ablaufen können und die Transaktionen auf gemeinsame Objekte zugreifen können, entstehen die gleichen Probleme wie für Prozesse bei Betriebssystemen: Durch Änderungen können ein inkonsistenter Datenbankzustand (lost updates, inkonsistente Analyse, ...) und falsche Abfrageergebnisse (Phantom-Problem) entstehen. Daher ist eine Transaktionsverwaltung notwendig, die die Transaktionen koordiniert.

Wie werden die Probleme denn nun im Speziellen gelöst?
Die Transaktionen werden synchronisiert, so daß die Serialisierung der Transaktionsabläufe sichergestellt werden kann...

...was heißt denn Serialisierung?
Konzept der Serialisierung an zwei Transaktionen kurz erläutert, d.h. Beschreibung von Konfliktoperationen und deren Behandlung im ursprünglichen und im äquivalenten serialisierten Transaktionsablauf.

Wie wird die Serialisierung denn nun sichergestellt bzw. in der Praxis realisiert?
Im wesentlichen durch das zwei Phasen Sperrprotokoll...

...was ist denn das zwei Phasen Sperrprotokoll?
Zwei Phasen (Sperrphase und Sperrlösephase) erklärt, wobei das zwei Phasen Sperrprotokoll den Nachteil hat das die zweite Phase (lösen der Sperren) sich über einen bestimmten Zeitraum erstreckt und daher zu einem kaskadischen Rücksetzen von Transaktionen führen kann, wenn eine Transaktion durch einen Fehler abgebrochen wird. Daher wird eine Variante des zwei Phasen Sperrprotokolls verwendet, das strenge zwei Phasen Sperrprotokoll, dessen zweite Phase ununterbrechbar ist.

Jetzt haben Sie durch das strenge zwei Phasen Sperrprotokoll die Probleme in der zweiten Phase vermieden, welche Probleme können denn nun trotzdem in der ersten Phase auftreten?
Da die erste Phase nach wie vor Zeit benötigt, kann eine Transaktion irgendwann versuchen ein Objekt zu sperren das bereits gesperrt ist. Dies kann zu Deadlocks führen, wenn Transaktionen Objekte gegenseitig sperren.

Nun zu der nächsten Frage aus Betriebssystemen: Was sind denn Deadlocks bzw. was sind die vier Bedingungen für Deadlocks?
Vier Bedingungen für Deadlocks aus Betriebssystemen aufgezählt

Wie können denn Deadlocks verhindert bzw. vermieden werden, insbesondere mit Bezug auf die Transaktionsverwaltung in Datenbanksystemen?
Entweder wird eine der vier Bedingungen nicht zulassen (ist in der Regel unschön) oder Vermeidung von Deadlocks durch die Vergabe von Zeitmarken. Bei letzterer Methode werden Transaktionen mit Zeitmarken versehen und eine Transaktion belegt das Objekt auf das sie zugreift mit ihrer Zeitmarke. Die Transaktionsverwaltung erlaubt dann nur noch den Zugriff

Prüfungsprotokoll

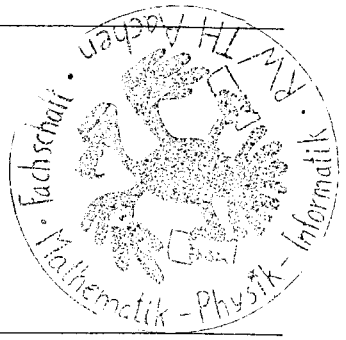
Gebiet: Praktische Informatik

Prüfer: Prof. Jarke

Gebiete: Betriebssysteme (Silberschatz)

Compilerbau (Aho / Seti / Ullmann)

Datenbanken (Jarkes Voerlesung / Silberschatz)



Allgemeines

Die Pruefung war recht locker und Jarke war trotz der großen Anzahl von Prüflingen guter Dinge. Er hat zuerst einmal nicht gefragt mit welchen Themengebiet ich beginnen möchte, sondern direkt mit Betriebssystemen begonnen.

Betriebssysteme

Hier begann er mit einem Griff mittendrin.

Jarke: Erläutern Sie doch einmal die Producer-Consumer-Problematik.

Ich habe dann erstmal etwas über Interprocess-Communication erzählt und die prinzipielle Problematik (shared memory / messages) erläutert. Jarke wollte aber eigentlich auf Deadlocks hinaus und hat natürlich auch direkt die entsprechende Folgefrage gestellt.

Jarke: Können dort Deadlocks auftreten?

Ja, können. Definition von Deadlock angegeben und langsam auf die notwendigen Bedingungen hingearbeitet.

Jarke: Dann nenn Sie doch mal die vier notwendigen Bedingungen

Also mutual exclusion, hold and wait, no preemption und circular wait erläutert.

Jarke: Welche Möglichkeiten gibt es denn nun?

Prevention, Avoidance und Detection mit Recovery. Jeweils die entsprechenden Möglichkeiten angeführt und erläutert. Hierbei wollte er dann noch den Bankers-Algorithmus näher erläutert haben. Dabei hat er wohl erwartet, daß ich den in Pseudo-Code aufschreibe, habe ihn aber nur mündlich erläutert. Dabei hat er die ganze Zeit das Blatt vor mir auf dem Tisch betrachtet und nachdem ich mit meiner Ausführung fertig war und immer noch kein Algorithmus auf dem Papier stand, fragte er mich ob ich denn jetzt den Algorithmus eigentlich erklärt hatte. Ich sagte Ja und er schaute etwas verdutzt und war dann auch meiner Meinung.

Jarke: Dann erzählen Sie mal etwas zu Paging und virtueller Speicherverwaltun

Hier habe ich dann erstmal etwas kreuz und quer alles von mir gegeben, was mir spontan so einfiel und das dann noch in ein "Konzept" gepackt. Also pages, frames, pagetable, Seitenaustausch, freier framepool etc. erläutert.

Jarke: Wie sieht das denn nun mit Thrashing aus.

Hier habe ich dann die beiden Möglichkeiten Bestimmung der pagefaultrate und workingsets erklärt. Danach wechselte er zum nächsten Thema.

Compilerbau

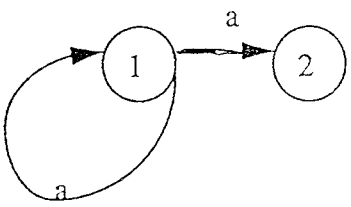
Jarke: zeichnen sie die Phasen des Compilerbaus auf.

Also das Bildchen aus dem Aho/Seti/Ullman aufgemalt

Jarke: Welche Sprachen und Tools werden, denn in den ersten drei Phasen benötigt?

reguläre Ausdrücke -> (nicht-)deterministische endliche Automaten -> Symboltokenstrom, kontextfreie Grammatiken -> Kellerautomaten -> Parsebaum, S- oder L- oder allgemein attributierte Grammatiken -> Dekorartion des Parsebaums rekursiv oder top-down oder bottom-up.

Jarke: Schreiben Sie mal einen nichtdeterministischen endlichen Automaten hin.



Dieser Automat ist zwar sehr einfach wie Jarke schnell feststellte, aber reichte vollkommen aus.

Jarke: Welcher reguläre Ausdruck wird denn damit beschrieben?

aa*

Jarke: Überführen Sie diesen Automat in einen deterministischen Automaten.

Hier habe ich nun den Algoritmus mit der Epsilon-Closure erklärt und vorgeführt. Also Closure (Startzustand, Epsilon) = {1} = A

Closure (A, a) = {1,2} = B

Closure (B, a) = {1,2} = B und fertig. Den zugehörigen Graphen habe ich dann auch noch hingeschrieben und er war glücklich.

Jarke: Jetzt kann es ja auch große Automaten geben. Gibt es einen minimalen DEA?

Ja. Hier habe ich dann kurz das Prinzip erläutert mit der Gruppenbildung akzeptierende und nichtakzeptierende Zustände und die weitere Bildung von Untergruppen. Prinzipiell werden so Zustände mit gleichem In- und Output gemergt.

Jarke: Nun zur nchsten Phase. Es gibt da eine kontextfreie Grammatik, die nicht regulär ist.

Dies war wieder eine Standardfrage. Also klar: a^ib^i

Ich habe dann auch die Grammatik S-> aSb | Epsilon angegeben und direkt die Beweisskizze erzählt. Ein Automat hat endlich viele Zustände und erkennt mit beispielsweise 2*i Zuständen a^ib^i . Soll das i nun um eins erhöht werden, so wird mindestens ein Zustand mehrfach durchlaufen und der Abgleich von a's und b's unmöglich (Pumping-Lemma).

Jarke: es gibt ja nun auch schlechte und auch gute Grammitken. (An dieser Stelle erweiterte er meine Grammatik etwas.) Wie kann man den nun feststellen ob die Grammatik gut ist?

Hier habe ich etwas von Normalformen (Greibach / Chomsky) erzählt. Dies gehört eigentlich in die Automatentheorie, aber was soll's. Als er dann jedoch meine Grammatik in CNF sehen wollte mußte ich nach einer halben Minute passen. (Vielleicht habe ich eine mögliche Regel vergessen gehabt, bei dem allgemeinen Modell)

Jarke: Ist ihre Grammatk LL(1) ?

Ja.

Jarke: Dann geben sie doch mal den Top-Down-Automaten an.

Hier habe ich ihm den Parsebaum aufgemalt und erläutert, wie an diesen gekommen bin. Das hat ihm auch gereicht.

Jarke: Dann wollen wir jetzt mal zu Datenbanken gehen.

Datenbanken

Jarke: Ich habe hier eine Patientenrechnung eines Arztes...

Dieses Beispiel findet sich in einem weiteren Prüfungsprotokoll ausführlich.

Jarke: Wie gehen Sie vor um eine SQL-Datenbank zu erstellen?

Entweder ich modelliere mit ER oder bilde universelle Relation, die in 3.NF gebracht wird.

Jarke: Nehmen Sie den 2. Ansatz

Ich habe dann die Relation aufgeschrieben und die funktionalen Abhängigkeiten eingetragen und den Schlüssel unterstrichen.

Jarke: Überführen Sie die Relation in die nächste NF

Ich habe dann die 2.NF erstellt und die 3.NF noch erläutert.

Jarke: Stellen sie nun folgende Anfrage in SQL. Geben sie die Namen, Nummern und Diagnosen aller Patienten, die im Jahr 1930 geboren sind aus.

Hierzu muss ich nun doch erstmal die Relationen genauer beschreiben:

Relation ..... Attribute.....

Patient..... Name, PNr, GebDat

Rechnung..... RechNr, Diagnose, PNr, ...

Also die Anfrage lautet dann:

select Name, Patient.PNr, Diagnose

from Patient, Rechnung

where GebDat = '1930' and Patient.PNr = rechnung.PNr

Jarke: Und jetzt in der relationalen Algebra:

Hier erwähnte ich, daß man diese überführung aus der SQL-Anfrage sofort machen kann, denn select entspricht der Projektion, das from dem kartesischen Produkt und die where Bedingungen der Selektion. In diesem Fall stellt aber das kartesische Produkt zusammen mit der letzten Bedingungen eine Natural-Join dar. Ich habe dann die Umsetzung hingeschrieben, worauf er direkt die nächste Frage (?) stellte:

Jarke: Und nun relationales Kalkül

Hier konnte ich es mir nicht verkneifen die Gegenfrage zu stellen: Tupel- oder Domänenkalkül?

Jarke: Vorzugsweise Domänenkalkül.

Hier habe ich erstmal etwas geschwitzt, da ich das nicht sofort konnte, sondern mir das erst herleiten mußte. Das hat dann auch noch geklappt und die Prüfung mit einer 1.3 bewertet.

So das wars von meiner Seite. Viel Spass beim Lernen und viel Erfolg bei Euren Prüfungen.

vom Mitnehmen auch Zusammenfassung BS

Prüfungsprotokoll : DB + BS

Prüfer : Prof. Jarke
Termin : 12.1.96
Fächer : Einführung - Implementierung DB
Betriebssystem von Silberschatz

Angefangen hat es wohl mit dem Betriebssystem, jedoch gab er die Frage nicht getrennt jenach die Thema sondern gemischt.

- Bakery-Algorithmus :
  Das Problemdarstellung, die Idee des Algorithmus erklären. Jedoch wollte er nicht explizite Angabe des Algo.
- Semaphor :
  Explizit aufschreiben und erklären, mit Busywaiting und die Vermeidung
- Deadlock :
  Die Bedingungen und die Vermeidung, Verhindern, Erkennen und beseitigen. Eine Vorlesung halten. Er stellt einige kleine Frage dabei.
- In DB-System :
  Wie wird Deadlock in DB-System verhindert? In anderen Protokolle und Transaktionsverwaltung schauen.
- Short-term Scheduler :
  Methode aufzählen und besonders mit dem Bild (siehe im Buch von Silberschatz) erklären, wie die Vorgang ist.
- 2PL-Scheduler :
  Was es ist, welche Probleme da gibt's (Deadlock), Bild malen, in welchen Phase auftritt, ein Bsp von Schedule mit Deadlock angeben, die Lösungsweg.... Hier muss man viel wissen.
- B*-Baum :
  Wie sieht er aus, die Kosten, ...wie übliche
- relationale Algebra :
  alle sorte von Operatoren angeben
- Dekompositon :
  zu beachten : Verlustlosigkeit, Abhängigkeitserhaltung. Die genaue Definition und erklärung
  Die Test-Methode.

Viele sagen, dass der Professor sein Stil geändert hat. Aber bleibt die Frage immer in selben Thema, finde ich. Ich habe kein einzige Frage gehabt, die nicht in Pruefungsprotokoll stand. Nur muss man einbisschen tiefer kennen. viel glueck.

- Seitenersetzungsstrategie
  'Pre Petching', ...

Gedächtnisprotokoll
Diplomprüfung Informatik
Vertiefungsgebiet Informationssysteme

Prüfer:	Prof. Jarke
Datum:	18.06.96
Dauer:	ca. 45 Minuten
Fächer:	Einführung in Datenbanken
	Implementierung von Datenbanken
	Objektorientierte Datenbanken (Buch von Kemper/Moerkotte)

Note: 1.3

1 Einführung in Datenbanken

- **Beschreiben Sie bitte die Entwicklung und wichtige Merkmale von Datenmodellen, ausgehend vom relationalen Datenmodell, über ER und erweitertes ER bis hin zu Objektmodellen.**
  Hier habe ich drei oder vier Minuten frei referiert. Wichtige Punkte: Relationales Datenmodell schafft Daten(repräsentations)unabhängigkeit und stellt damit einen Fortschritt gegenüber den früheren, d.h. den hierarchischen und netzwerkbasierten, Datenmodellen dar. ER und erweitertes ER dienen der konzeptuellen Modellierung, abstrahieren vom logischen Datenbankschema und sind daher intuitiv verständlicher als das relationale Modell. Objektmodelle wiederum erlauben adäquate Modellierung auch komplexer Objekte z.B. in ingenieurwissenschaftlichen Disziplinen. Weitere Charakteristika sind Verbindung von Daten und Operationen, Vererbung und spätes Binden.

- **Wenn ich ein ER- oder EER-Schema auf relationale Schemata abbilde, was gibt es da zu beachten?**
  Hier nannte ich zuerst konkrete Transformationsoptionen, z.B. für 1:N-Beziehungen und Generalisierungen, aber Jarke wollte es etwas allgemeiner haben. Also habe ich relationale Entwurfsziele genannt: geringe Relationzahl wegen teurer Joins; potentiell wenige Nullwerte wegen Platzverbrauch und Semantikproblemen; einfacher Charakter von Integritätsbedingungen (z.B. Schlüsselbedingungen, Domänenbedingungen) wegen leichterer Implementierbarkeit; gute Normalformeigenschaften zur Verhinderung von Anomalien.

- **Wie sind Normalformen definiert?**
  Hier erwähnte ich die 1NF, 2NF, 3NF, BCNF, 4NF und 5NF. Von 1NF bis BCNF wollte Jarke die genaue Definition haben. Die auf multivalued dependencies und join dependencies basierenden Normalformen (4NF und 5NF) haben ihn nicht weiter interessiert.

- **Wie gelangt man zu einer gewünschten Normalform? Grundlegende Prinzipien?**
  Top-down (Dekomposition) oder bottom-up (Synthese).

- **Welche Eigenschaften gelten für solchermaßen erzeugte Schemata?**
  Dekomposition: BCNF und lossless join. Dekompositionsverfahren ist NP-vollständig. Synthese: 3NF und Erhalt funktionaler Abhängigkeiten.

- **Wie funktioniert der Bottom-Up-Algorithmus?**
  Zunächst CFD's (compound functional dependencies) erklärt. Aus jeder CFD des annular cover wird ein Relationenschema. Falls kein Schema einen candidate key der Universalrelation enthält, so kann noch ein entsprechendes Schema, bestehend aus einem

candidate key, hinzugenommen werden; auf diese Weise wird sogar noch der lossless join garantiert.

- **Wie wird denn der Abschluß einer Menge funktionaler Abhängigkeiten berechnet?**
  Der Abschluß der funktionalen Abhängigkeiten wird nicht direkt berechnet, allein schon wegen der exponentiell vielen trivialen Abhängigkeiten. Stattdessen werden Abschlüsse über Attributen betrachtet: $X \rightarrow Y \in F^+ \iff Y \subseteq X^+$ Diese Frage kann in $O(n)$, $n =$ textuelle Gesamtlänge der funktionalen Abhängigkeiten in F, entschieden werden.

- **Welche besondere Eigenschaft hat die relationale Algebra?**
  Abschlußeigenschaft: Das Ergebnis einer relationalen Operation ist stets wieder eine Relation. Dies ist ein Vorteil gegenüber objektorientierten Anfragesprachen.

- **Wie werden Anfragen in SQL gestellt?**
  Hier habe ich den ganzen Sermon aufgezählt: `SELECT` – `FROM` – `WHERE` – `GROUP BY` – `HAVING` – `ORDER BY`. Das `SELECT` steht für Projektion, `FROM` für kartesisches Produkt, `WHERE` für Selektion usw.
  Zuletzt erwähnte ich, daß `SELECT`-Ausdrücke geschachtelt werden können. Diese Bemerkung brachte Jarke auf die nächste Frage.

- **Kann zu jeder geschachtelten SQL-Anfrage eine äquivalente, nicht-geschachtelte Anfrage angegeben werden?**
  Ich bejahte diese Frage, ohne jedoch den Beweis hierfür anzutreten. Stattdessen erwähnte ich, daß geschachtelte Anfragen manchmal einfacher zu konstruieren sind (bottom-up), und daß es deshalb sowohl geschachtelte als auch nicht-geschachtelte Ausdrucksmöglichkeiten gibt.

- **Geben Sie doch mal ein Beispiel für eine geschachtelte SQL-Anfrage an!**
  Hier griff ich auf die berühmt-berüchtigte Supplier-Parts-Datenbank zurück (btw die findet sich schon in Codd's Original-Paper von 1970). Auf Jarkes Anregung formulierte ich eine Anfrage, die alle in Aachen befindlichen Supplier ausgibt, die rote Teile liefern. Das Ergebnis, mit einer kleinen Korrektur durch Jarke, war folgendes:

```
SELECT S#
FROM   S
WHERE  CITY = 'Aachen' AND S# IN
 (SELECT S#
  FROM   SP
  WHERE  P# IN
    (SELECT P#
     FROM   P
     WHERE  COLOR = 'red'))
```

- **Wie ändert sich das `SELECT`-Statement, wenn ich alle Aachener Supplier haben will, die *keine* roten Teile liefern?**
  Die Antwort war nicht weiter schwierig, es mußte nur heißen:

```
...
WHERE CITY = 'Aachen' AND S# NOT IN
...
```

- **Ja, und wie wird die Anfrage ohne Schachtelungen formuliert?**
  Ich formulierte die Anfrage um und beging einen Fehler dahingehend, daß alle Supplier ausgewählt wurden, die *unter anderem* ein nicht-rotes Teil liefern. Einerseits war mir

klar, daß meine 'Lösung' nicht richtig sein konnte, aber andererseits fiel mir zunächst nichts besseres ein. Aufgefordert, die Anfrage korrekt zu formulieren, kam ich jetzt doch sehr ins Stocken, bis Jarke einen hilfreichen Hinweis gab:

- **Versuchen Sie es doch einmal mit relationaler Algebra! Was muß ich anstelle des negierten Existenzquantors nehmen?**
  In diesem Moment fiel es mir wie Schuppen aus den Haaren: Mengendifferenz! Noch unfähig, ein Wort auszusprechen, schrieb ich mit der Hand das Zeichen für Mengendifferenz in die Luft, und Jarke nickte wohlwollend. Daraufhin erstellte ich die Lösung:

$$\Pi_{\texttt{S\#}}(\sigma_{\texttt{CITY} = \text{'Aachen'}} S) \setminus \Pi_{\texttt{S\#}}((\sigma_{\texttt{COLOR} = \text{'red'}} P) \bowtie SP)$$

  In SQL geht das mit `EXCEPT`. Brauchte ich aber nicht mehr aufzuschreiben.

2 Implementierung von Datenbanken

- **Welchem algebraischen Konstrukt entsprechen die Schachtelungen, die in dem vorhin behandelten SQL-Beispiel angegeben sind?**
  Semi-Join. Bedeutet, daß besonders effiziente Auswertung möglich ist.

- **Welche Ausdrücke können nicht durch Semi-Joins ausgewertet werden?**
  Solche mit Zyklen im Quantgraphen.

- **Wie überprüfe ich dieses Kriterium, wenn eine geschachtelte SQL-Anfrage gegeben ist?**
  Ein Zykel ist dann vorhanden, wenn eine Tupelvariable nicht nur in ihrem definierenden Niveau auftaucht, sondern zusätzlich auch in einem niedrigeren Niveau verwendet werden muß.

- **Wie wird eine Anfrage durch ein RDBMS optimiert?**
  Zunächst High-Level-Optimierung, zum Beispiel Tableau-Methoden zur Vermeidung überflüssiger Teilanfragen. Weiterhin besonders wichtig: Heuristiken für relationale Algebra, z.B. Selektion und Projektion möglichst früh ausführen.

- **Was heißt eigentlich: 'eine Selektion ausführen' ? Wie wird eine solche Operation realisiert?**
  Dies hängt von den gegebenen Zugriffsstrukturen ab. Liegt ein Index, z.B. ein B^*-Baum, für das Selektionsattribut vor, so können die selektierten Tupel durch einen Index-Scan ermittelt werden. Für komplexere Anfragen muß ggf. eine geeignete Kombination von Zugriffsstrukturen gewählt werden.

- **Wie wird die Reihenfolge von Joins optimiert?**
  Einerseits gibt es Join-Kostenmodelle, die von bestimmten Voraussetzungen, z.B. Gleichverteilung von Attributen, ausgehen. Außerdem gibt es Heuristiken. Joins werden zunächst möglichst so gruppiert, daß keine kartesischen Produkte als (Zwischen-) Produkte entstehen. Weiterhin sollten Joins über Schlüsseln früh ausgeführt werden, da das Ergebnis nicht mehr Tupel als die jeweils andere Relation enthalten kann.

- **Sie haben Kostenmodelle für Join-Reihenfolgen erwähnt. Können Sie etwas über diese erzählen?**
  Hier fielen mir nur wilde Assoziationen ein: Beispiel-Anfragen über Leute in Computer-Departments, die in einem Büro im fünften Stock wohnen... irgendwas war da doch... spekulier...
  Schließlich mußte ich passen, nicht jedoch ohne den wertvollen Hinweis auf eine Diplomarbeit am Lehrstuhl, die sich genau mit diesem Thema beschäftige und wo der

betreffende Diplomand sicher gut Bescheid wisse. Diese Art von Meta-Wissen jedoch fand naturgemäß keine Gnade beim Prüfer ;->
Anmerkung: Nach der Prüfung sagte mir der Beisitzer, daß es bei Jarke in letzter Zeit kaum eine Prüfung gegeben habe, wo er *nicht* nach Kostenmodellen für Join-Reihenfolge-Optimierung gefragt habe. Hier sollte man vorbereitet sein, auch wenn Jarke früher 'nie' in diese Richtung gefragt hat.

- **Zur Recovery. Welche Arten von Recovery gibt es?**
  Hier erwähnte ich drei Arten: Recovery für Transaktionsfehler, einen System-Absturz oder den Verlust der Datenbank (Headcreash).

- **Den letzten Punkt wollen wir mal vergessen, schließlich wissen wir ja, daß Festplatten ein absolut zuverlässiges Medium sind *(lacht)*.**
  Wie läuft denn die Recovery nach einem System-Crash?
  Ich antwortete mit einer kurzen Erklärung des allgemeinsten Protokolls, also UNDO-REDO.

- **Welche Informationen brauche ich für UNDO und REDO?**
  UNDO: Before-Image geänderter Datenobjekte. REDO: After-Image geänderter Datenobjekte. Dazu jeweils die Angabe, welche Transaktion für die Änderung verantwortlich war.

- **Wann sind UNDOs oder REDOs überflüssig?**
  UNDO ist überflüssig, wenn geänderte Seiten erst zum Commit-Zeitpunkt in die stabile Datenbank geschrieben werden. REDO ist unnötig, sofern geänderte Seiten spätestens mit dem Commit in die stabile Datenbank geschrieben werden. Ein Schattenspeicher kommt z.B. ohne UNDO oder REDO aus.

- **Wenn Schattenspeicher solche Vorteile haben, warum konnte sich dieses Konzept nicht durchsetzen?**
  Dies liegt weniger an dem zusätzlichem Speicherbedarf als vielmehr daran, daß Schattenspeicher einem effizienten Clustering von Daten entgegenstehen.

- **Welche Besonderheit gilt für das Schreiben des Logs?**
  Log-Einträge werden streng sequentiell angelegt. Daher lautet die Empfehlung, das Log auf einer eigenen Platte zu führen. Diese Vorgehensweise ist nicht nur sicherer, sondern vermeidet auch unnötige Platten-Suchzeiten und erhöht so den Durchsatz.

3 Objektorientierte Datenbanken

Hier blieb nicht mehr sehr viel Zeit (ca. 5 min), so daß Jarke nur einige Punkte gestreift hat.

- **Was ist bei Mehrfachvererbung zu beachten?**
  Mehrfachvererbung kann zu Konflikten führen, wenn Merkmale gleichen Namens von verschiedenen Oberklassen vererbt werden. Die Auflösung von Konflikten kann auf verschiedene Weisen erfolgen (z.B. Zwang zur Umbenennung). Eine allgemein akzeptierte Vorgehensweise gibt es nicht.

- **Wie können Operationen verfeinert werden?**
  Der Rückgabewert kann spezialisiert, die Parameter können generalisiert werden (zumindest in GOM). Auf diese Weise bleibt die Substituierbarkeit gewahrt.

- **Was heißt Substituierbarkeit?**
  Jede Instanz einer speziellen Klasse kann überall auch dort verwendet werden, wo Instanzen einer allgemeineren Klasse gefragt sind.

4 Fazit

Die Prüfungsatmosphäre war entspannt und angenehm. Mit der Benotung war ich zufrieden. Was an der 1.0 gefehlt hatte, waren der eine SQL-Aussetzer zwischendurch und meine Unkenntnis von Join-Kostenmodellen. Aber wie heißt es so schön: Verschnitt ist überall ;-)

Prüfungsprotokoll Praktische Informatik bei **Prof. Jarke**

Einf. DB, Implementierung DB, Betriebssysteme
Note: 1.7
Dauer: 60 Min.

8. Oktober 1996

1 Material

- Vorlesungsfolien Einführung in Datenbanken
- Vorlesungsfolien Implementierung von Datenbanken
- Vossen, G.: "Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme", 2. Auflage, 1994
- Lockemann, P.C. und Schmidt, J.W.: "Datenbank-Handbuch", 1987 (Nur ergänzend)
- Silberschatz, A. und Galvin, P.B.: "Operating System Concepts", 1994
- Diverse Prüfungsprotokolle

2 Die Prüfung

Jarke: Hmm, Betriebssysteme ... Erläutern Sie doch mal die Speicherhierarchie und geben Sie die historische Entwicklung an.

Ich: Schicke erst mal voraus, daß ich zuerst die aktuelle Speicherhierarchie erläutern werde und anschließend die historische Entwicklung umreißen werde. Allgemein: Hierarchie nach Geschwindigkeit. Fange an zu erzählen: Register, Cache (mit Hinweis auf 1st und 2nd Level) Hauptspeicher, Festplatte, optische Medien, Bandlaufwerke. Dann Historie (voll improvisiert): Erzähle was über Trommelspeicher als Hauptspeicher, Bandlaufwerke als einziger Massenspeicher, noch weiter zurück: Lochkarten...

Jarke: (Scheint zufrieden zu sein) Sie erwähnten gerade die Register. Wie ist denn das mit den Registern beim Prozeßwechsel?

Ich: (Aha!) Fange an zu erzählen: Müssen natürlich gesichert werden. Insbesondere PC, Flags und SP. Macht Betriebssystem nach Interrupt, wobei der Prozessor beim Interrupt eh den PC und die Flags auf den Stack legt. Register werden dann mit diesen Im PCB (Process Control Block) gespeichert, wo ja auch alle anderen für den Prozeß relevanten Daten (Page Table, ID, evtl. Priorität) gespeichert werden.

Jarke: Und wie geht das dann genau wenn der Prozeß wieder gestartet wird?

Ich: Nachdem vom laufenden alles gesichert ist, werden alle Register wieder restauriert ...

Jarke: ... wie genau?

Ich: Erst werden alle Register und die Page Table restauriert, dann die Flags und als letztes der PC gesetzt, so daß der Prozeß dann wieder läuft.

Jarke: (zufrieden) Sie erwähnten die Page Table. Erzählen Sie doch mal etwas über Speicherverwaltung.

Ich: Hole etwas weiter aus und umreiße grob die gesamte Problematik, um dann das Paging genauer zu erläutern. Erwähne dabei auch das Auslagern von Seiten und den Begriff "Demand Paging" und Seitenersgetzungsstrategien. An diesem Punkt hätte ich noch eine Stunde erzählen können, durfte ich aber nicht...

Jarke: Sie kennen doch die Segmentschnittstelle im Schichtenmodell von Datenbanken... Warum verwendet man denn da nicht einfach die BS-Routinen?

Ich: Ja weil doch das DBMS viel besser weiß welche Seiten wie verwendet werden und ob diese noch mal gtebraucht werden usw. (ewähne auch den Spezialfall des Joins wo eine Relation wirklich nur einmal gebraucht wird).

Jarke: (hakt weiter nach, irgendwie hatte ich nicht getroffen was er hören wollte) Erläutert mir dann den LRU-k-Algorithmus (Wenn eine Seite k-mal benutzt wurde, bleibt sie drin) und meint das der ja auch noch recht neu und man müte das vielleicht auch nicht wissen. Er hätte ihn in der letzten Vorlesung erläutert (die ich natürlich nicht gehört hatte ...). Da ich auch den Spezialfall des sequentiellen Lesens erwähnt hatte fragt er da weiter: Wo man denn gerade in letzter Zeit häufig sequentiell Lesen würde und wo das Problem sei...

Ich: Multimedia-Datenbanken. Filme! Problem: Durchsatz, soll schließlich nicht ruckeln! Lösung: read-ahead!

Jarke: (zufrieden) Fragt ziemlich unvermittelt nach dem Bankers-Algorithmus!