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:

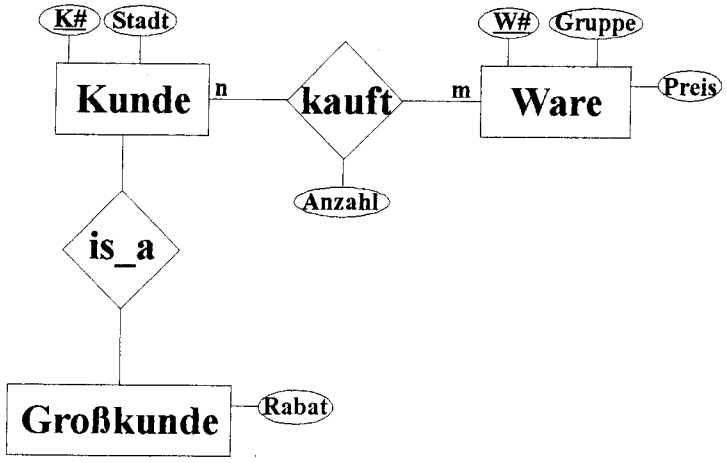
EDB: Folienkopien d. Vorlesung / Vossen "Datenmodelle ..." (zum Nachschlagen)
OODB: Kemper/Moerkotte "OODB" (Relevant: Kap. 8-15)
Kemper "Grundlagen OODB" (30 Seiten Super-Überblick)
RE1: Folienkopien d. Vorlesung
RE2: Oestereich "Objektorientierte Softwareentwicklung mit UML"

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.

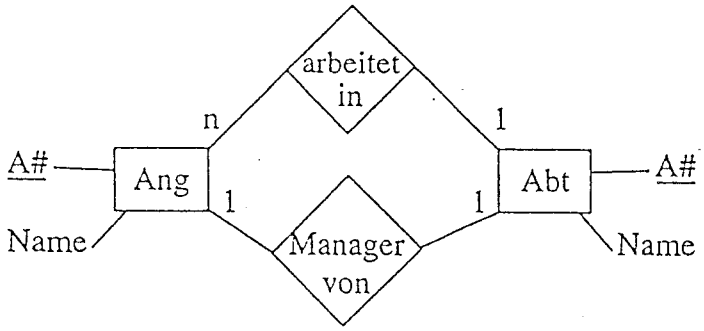
Datenbanken (15 min)

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
```

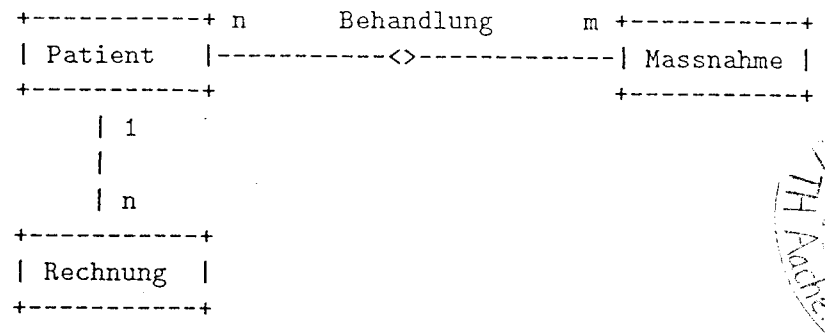
Er hat dann noch erklaert: die Rechnung-Nr ist eindeutig (also keine eigene Nummerierung pro Kunde), An einem Tag pro Patient nur eine Behandlung, P/T bedeutet irgendwas mit Persoenlich oder technische Behandlung und noch irgendwas.

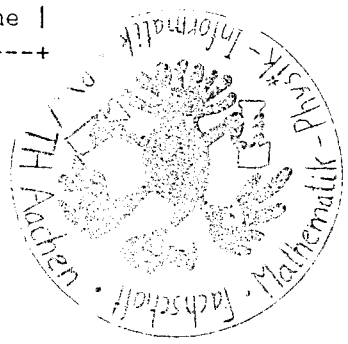
Nach ca. 3 Minuten dann die erste Frage: Der Arzt will diese Rechnungen mit dem Computer erstellen. Wie gehen sie denn so ran?

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.

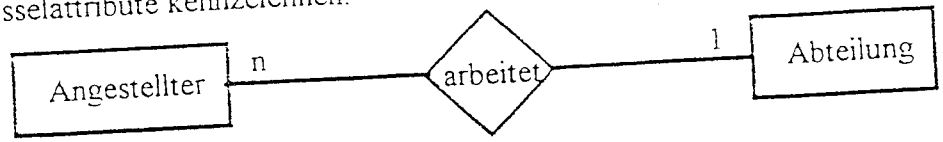
Welchen Operatoren der relationalen Algebra entsprechen die einzelnen Teile der SELECT... FROM...WHERE...-Anweisung?
SELECT entspricht Projektion (π), WHERE entspricht der Selektion (σ) und FROM der Auswahl der zu bearbeitenden Relationen (Argumente).

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?