Gedächtnisprotokoll Praktische Informatik

Prüfer: Prof. Jarke
Grundlage: Vorlesung Einführung in Datenbanken
Vorlesung Implementierung von Datenbanken
Vorlesung Requirements Engineering (K. Pohl)
Vorlesung Verteilte Systeme (Popien, i4)
Dauer: 45 Minuten
Datum: 29.9.95
Note: 1.3

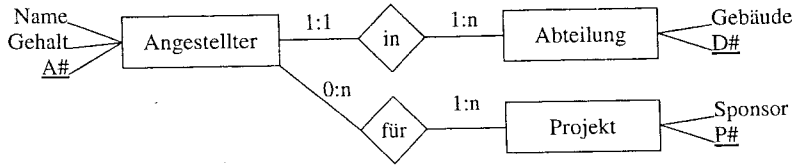
Die Prüfung

Requirements Engineering

Jarke: Gegeben folgendes ER-Diagramm… Hmm, nein, Sie hatten ja auch RE, also erstellen sie mal ein ER-Diagramm zu folgender Situation:

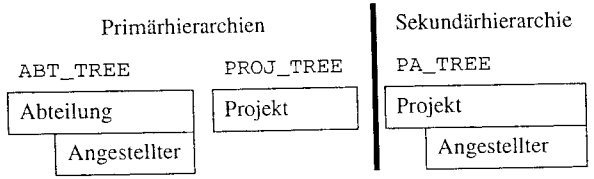
- Angestellte mit Nummer, Name und Gehalt
- Abteilungen mit Nummer (D#), Titel und Gebäude
- Ein Angestellter ist genau einer Abteilung zugeordnet, in einer Abteilung arbeiten mehrere Angestelle
- Projekte mit Nummern und Geldgebern
- Ein Angestellter kann in mehreren Projekten arbeiten, ein Projekt beschäftigt mehrere Angestellte

Ich: Hingemalt, Schlüssel festgelegt.

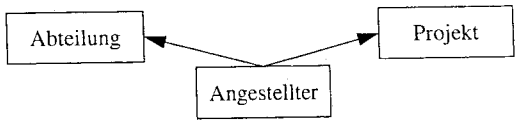


Einführung in Datenbanken

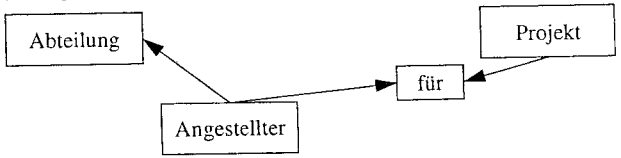
Jarke: Übertragen sie das mal ins Hierarchische, Netzwerk- und Datenmodell.
Ich: *(Schluck)* Im Hierarchischen Modell müßten die Angestellten ja wohl entweder redundant abgelegt werden, oder eine Relation wird mit einer Sekundärhierarchie dargestellt. Also:



Jarke: Hmm. Jaja. Und wie wird das so auf Platte abgelegt?
Ich: Ähm, hintereinander. *(Nach einigen Versuchen:)* Ein Abteilungsdatensatz, dann Angestellte der Abteilung. In der Sekundärhierarchie nur Pointer.
Jarke: Na gut.
Ich: Und im Netzwerk-Modell haben wir auch nur 1:n – Beziehungen. Das Schema würde so notiert *(um erstmal Luft zu bekommen)*:



Eine Instanz davon sähe so aus: *(gezeichnet: 1 Projekt, 3 Angestellte, ein paar Zeiger.)*
Jarke: Aber so geht das ja noch nicht. Schaun Sie sich doch mal die 'für' – Relation an.
Ich: *(Rätsel...)* Ach ja, die cat Records fehlen. Also *(nach Korrekturen, Instanz ergänzt)*:



Jarke: Jaaa…
Ich: *(Tief durchatmen, da wären wir doch noch irgendwie durchgekommen)* Übersetzung RE → Relational: Entities nach Tabellen, Relationships nach Tabellen oder verschmelzen. Beide Optionen notiert.
Jarke: Ja, dann stellen wir doch mal eine Anfrage. Sagen wir mal, es gäbe da die Vermutung, daß die Leute in dem einen Gebäude mehr verdienen als die in dem anderen. *(Andeutungen über die Zustände in der Informatik...)* Also hätten wir mal gerne die Durchschnittsgehälter pro Gebäude.
Ich: Das dann in SQL also? *(Dumme Frage eigentlich...)* Eher explorativ notiert. Kommentare über den hübschen `JOIN` in SQL2, was aber nicht interessierte…

```
SELECT    AVG(Gehalt)
FROM      Angestellter JOIN Abteilung JOIN Projekt
WHERE
GROUP BY  Gebäude
```

Jarke: Warum haben sie denn die Projektrelation dabei?
Ich: Ähm, ich schreibe immer erstmal alle Relationen auf und streiche dann die unnötigen. *(Allgemeines Grinsen. Außerdem fehlte die 'für'-Relation.)*
Jarke: Und da haben wir jetzt eine höllisch interessante Zahlenreihe…
Ich: Oh, die Gebäude sollten wirklich dabei. Ergebnis:

```
SELECT    AVG(Gehalt), Gebäude
FROM      Angestellter JOIN Abteilung ~~JOIN Projekt~~
~~WHERE~~
GROUP BY  Gebäude
```

Jarke: *(Murmelt)* War auch wegen der Mengeneigenschaft nötig. *(Verstehbarer)* Damit haben wir auch die Duplikate, wenn zwei Leute das gleiche verdienen.
Ich: *(Verstehe nur: das gibt aber Ärger mit den Duplikaten)* Das kommt so aber doch hin, wenn zwei Leute das gleiche verdienen! *(Das mußte natürlich auch geklärt werden... Großmaul, ich.)*

Implementierung von Datenbanken

Jarke: Da gibt es ja so eine Relationale Algebra, erzählen Sie da doch mal was drüber…
Ich: *(Zuerst recht durcheinander, dann geordneter):*

- Strukturen: Relationen als Teilmengen Kreuzprodukte der Komponentendomänen
- Operationen: σ, π, $\otimes$ *(Hier: Join)* und bei kompatiblen Mengen: $\cap$, $\cup$, —
- Integrität: FD, JD, Zeitabhängige *(ID fiel mir nicht ein. Hat er aber nicht drauf bestanden)*

Jarke: Nun hat die Relationale Algebra eine wichtige Eigenschaft… Was kommt denn als Ergebnis

der Operationen heraus?

Ich: Äh, Relationen: Abgeschlossenheit!

Jarke: Ja, aber dafür brauchen wir noch eine Bedingung…

Ich: *(???? Die kompatiblen Mengen hatte ich extra mit erwähnt. Also Raten:)* Passende Spalten beim Join?

Jarke: Dann kommt ja einfach das Kreuzprodukt heraus. *(Genau meine Meinung…)* Was anderes.

Ich: *(Verzweifelt)* Bei den Mengenoperationen müssen die Mengen kompatibel sein, aber…

Jarke: *(Eher zu sich selber)* Ja. Bei der Vereinigung und Differenz kommt das auch so hin, aber beim Schnitt gibt das echten Ärger.

Ich: …das habe ich doch gleich am Anfang gesagt?

Jarke: *(Soso)* Und dann gibt es noch ein Problem bei Implementierungen…

Ich: *(Rätsel)* Duplikate…

Jarke: Jaa. Wo wir grade dabei sind, wie sieht's denn mit der Komplexität der Operationen aus.

Ich: *(Fluch! Was mache ich hier?)* Öh, σ und π sind wohl linear…

Jarke: Nicht so voreilig… Denken Sie mal nach. Geht das nicht schneller?

Ich: *(Also nicht linear… Er muß wohl Speicherstrukturen meinen.)* Logarithmisch? Wenn die Tabelle als Binärbaum vorliegt? *(Nicht daß ein unoptimierter σ_{true} da nicht ganz drüber müßte…)*

Jarke: Schon besser. Aber warum nimmt man denn normalerweise keine Binärbäume?

Ich: Die B-Bäume und B*-Bäume sind so viel praktischer. Da kann man die Knotengröße genau auf Blockgröße des Dateisystems austreiben.

Jarke: Und wo liegt da der Vorteil? *(Oder sowas. War jedenfalls noch nicht zufrieden)*

Ich: Locking? Weniger unvorhersehbare Sperrprobleme? *(Rat, rat.)* Durchsatz?

Jarke: Da gibt es aber noch einen anderen Vorteil.

Ich: Ach ja, Ausgeglichenheit. B[*]-Bäume sind immer ausgeglichen.

Jarke: Jaaa. Und damit haben wir welche Komplexität bei `INSERT` und `DELETE`?

Ich: $\log_{\text{Knotengröße}}(n)$, weil Verschmelzen und Aufspalten schlimmstenfalls bis zur Wurzel oder einem Blatt laufen.

Jarke: OK. Aber geht das nicht noch schneller?

Ich: Hash. Da ist die Zugriffszeit auf jedes Element konstant.

Jarke: Gut. Und wie sieht das jetzt bei π aus?

Ich: *(Wie implementiere ich eine Projektion, wenn nicht als Filter?)* Eigentlich doch kein Mehraufwand: in einem Ausdruck nur Teile weitergeben, sonst erst recht. Also wohl wieder linear oder sonstwas bei den entsprechenden Strukturen. Oder nur Verwaltungsinformation modifizieren oder?

Jarke: Aber die Mengeneigenschaft?

Ich: Ach ja, dafür müßten die Duplikate entfernt werden. Da kommt ein Sortieraufwand dazu. Aber die Basisrelation kann ja schon sortiert vorliegen *(wenn das bei den Joins geht…)*

Jarke: Also ist π sogar noch teurer als σ. Aber die Duplikate bleiben ja auch oft noch drin. Und die Vorsortierung … Hm. Aber der Join? Wie wird der implementiert?

Ich: 3 übliche Methoden:

- Hash-Join: Wenn die Basistabellen über die Joinspalten gehasht sind, Join in den Hashbuckets ausführen.
- Sort-Merge: Wenn die Basisrelationen passend sortiert sind, kommen die Tupel gleich in der passenden Reihenfolge.
- Nested Loop: Wenn's nichts besseres gibt.

Jarke: Dann gibt es ja noch eine Spezialform, dem Semijoin *(Hier: $\oplus$)*

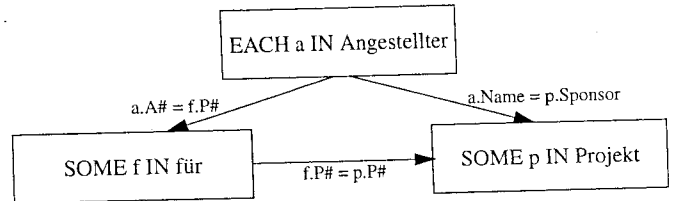
Ich: Definition $R \oplus S = \pi_R(R \otimes S)$, Viieel günstiger: Nur Existenz zu testen, keine Vergrößerung der Antwortmenge. *(Schwärm…)*

Jarke: Wenn der so toll ist: können alle Anfragen nur mit Semijoins beantwortet werden?

Ich: Nein, das war ein Ergebnis der Strukturierten Anfrageklassifikation: Hartnäckig bösartige Teilausdrücke können nicht Semijoin-ausgewertet werden. Zyklen im Quantgraph.

Jarke: Können Sie dafür ein Beispiel geben?

Ich: Die waren immer so kompliziert… Professoren, Gastprofessoren, Städte… Mehrstufige Prädikate… Oder mal mit dem Schema von oben… *Bastel…* So vielleicht:



Jarke: Ich dachte eher an einen Datenbankzustand…

Ich: *(Totalblockade)* Da fällt mir so schnell nichts ein.

Jarke: War vielleicht auch was viel verlangt.

Verteilte Datenbanken bzw. Verteilte Systeme

(Eigentlich ging die Prüfung über die Vorlesung "Verteilte Systeme", die mit Datenbanken außer einer Abhandlung über Transaktionen ("die sind sterilisierbar" anstelle von 'serialisierbar'!) nichts zu tun hatte. Jarke machte daraus kurzerhand verteilte Datenbanken. Obwohl er bei der Festlegung der Prüfungsthemen meinte, neben DBI&II könnte keine Vorlesung 'Wissensrepräsentation' rein, das würde zu einseitig…)

Jarke: Was mach den Semijoin denn für Verteilte Datenbanken besonders interessant?

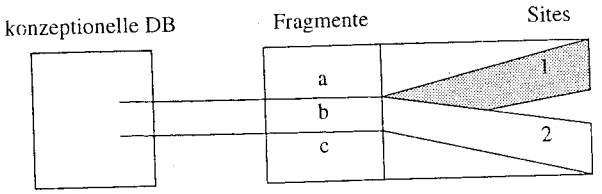
Ich: Man muß nicht komplette Relationen, sondern nur die Joinspalte verschicken… *(Grübel:)* Immer die aus der Zweiten Tabelle, oder auch mal anders?

Jarke: Ja. In der Regel allerdings immer dahin verschicken, wo das Ergebnis gebildet wird. Und was ist sonst noch zu beachten? *(Oder sowas, lief wohl hinaus auf:)*

Ich: Operationen zu den Daten oder umgekehrt.

Jarke: Wie wird denn eine Datenbank verteilt?

Ich: 2 Schritte: Fragmentation und Allokation. Für beides gibt es Heuristiken (irgendwas im Schema schreit einen an) und mathematische Modelle. Prinzip:



Jarke: Welche Möglichkeiten der Fragmentation gibt es denn?

Ich: Horizontal: Spalten oder Relationen abtrennen. Spalten werden aber abgetrennt, denn bei so einer losen Koppelung sollten im Design schon getrennte Tabellen entstanden sein…

Jarke: Nun, so selten ist das aber nicht. Geben Sie doch mal ein Beispiel.

Ich: Äh… Personaldatenbank: Angstellter(A# | Name, Adresse | Gehalt, Zulagen | Vorstrafen …)

Jarke: Hmm, na gut. Sonst in Produktionsumfeld: Ein Produkt wird von sehr vielen Daten beschrieben, die nicht überall relevant sind. Aber die Normalisierungsverfahren machen da oft schon kleinere Tabellen daraus. Und weiter?

Ich: Vertikal: Partitionierung des Wertebereichs einer (Schlüssel-) Komponente. Beispiel Kontonummern.

Jarke: Und die Allokation?

Ich: Entweder es gibt da klare Sachlagen (Kontonummern) oder recht detaillierte mathematische Modelle… *(Unbehagen)*

Jarke: Wie läuft das denn? Erklären Sie mal! Schaltkreistheorie könnte helfen. *(Heiterkeit)*

Ich: Man nimmt sich so einen Stapel üblicher Anfragen, zerlegt den irgendwie (Karnaugh, Quine-McClusky) in Minterme und zerhackt die Datenbank solange, bis irgendwie statistisch gleich große Stücke herauskommen. Wohl eher was für den Fall, wenn man bergeweise Statistiken hat, aber nicht weiß, was eigentlich in den Relationen drinsteht.

Jarke: Naja. Dann kommt die Allokation...

Ich: Nun, da werden die Fragmente den Rechnern zugeteilt, auch durchaus mehrfach (Replikation)...

Jarke: Warum denn dieses? Was für Vorteile?

Ich: Datensicherheit, Antwortzeit, Ausfallsicherheit... *(In einigen Anläufen. Eigentlich wollte ich auf Kosten/Nutzen raus, wo er doch heute so detailliert war, aber das ließ er nicht zu.)*

Jarke: Das hört sich ja nur nach Vorteilen an. Wieso verteilt man denn nicht immer komplett? Die DB einmal auf CD, dann wöchentlich neu verschicken? (Wie es ja heutzutage doch üblich wird...)

Ich: Die Kosten werden zu hoch ...

Jarke: So teuer ist das heuer auch nicht mehr...

Ich: Aber die Schreibkosten!

Jarke: Ah!

Ich: Wenn halt ständig überall geschrieben wird...

Jarke: Wie denn...

Ich: Da würde ich am liebsten ein Beispiel für nehmen... *(In der Vorlesung fand sich dazu eh kaum was)*: im Ingres-System wird der Verteilte Recovery-Manager eh auf den Anwendungsentwickler abgewälzt. Auf allen Rechnern einzelne Transaktionen, Log führen, `PREPARE TO COMMIT`, Log, `COMMIT`, Log. Evtl. Recovery.

Jarke: Also hauptsächlich Verwaltungskosten. Noch 3 Minuten, also noch eine Frage zu RE:

Requirements Engineering

Jarke: Da gab es doch diese SA-Diagramme...

Ich: *(male schon mal die Symbole auf, Top-Level-Diagramm, Schachtelung)*

Jarke: ...jaaa... Was für Konsistenzbedingungen gab es denn da bei der Schachtelung?

Ich: Balancierung: visuelle und Dictionary. Bei '+'–Datenströmen reicht entweder eine Komponente, oder die können aufgeteilt werden, weiß nicht.

Jarke: OK. Und wie kann man an die Erstellung eines solchen Diagramms herangehen?

Ich: In der Vorlesung 4 Methoden:

- Datenverfolgung (Zwischenspeicher+Prozesse als Nebenprodukt)
- Stimulus-Response (Ereignisverfolgung, nötige Daten)
- Puzzle/Bottom–Up, Benennungsprobleme
- (Eigentlich quer dazu:) Kontextabgrenzung. Top-Level-Diagramm.

Jarke: Danke, das war's dann.

Allgemein

Ich war insofern überrascht, als daß ich von älteren Protokollen den Eindruck (Wunschdenken?) hatte, daß es weniger um Details als um den Überblick ginge. Das Netzwerk- und Hierarchische Datenmodell hatte ich dann gedanklich schon mal beiseite gelegt und die mathematischen Modelle (Allokation) nur widerwillig überflogen. Falsch gedacht. Ihm ist anscheinend keine Ecke zu abstrus (bzw. die echt abstrusen Sachen kommen gar nicht erst in die Vorlesung).
Bei Fremdvorlesungen wie VSe empfiehlt es sich doch wohl, den Inhalt genauer zu bestimmen. Erstaunlich, wie oft doch noch Mißverständnisse auftreten. Wir hatten vorher Prüfungstraining gemacht, mit Uhr, Beisitzer und 'Sie', aber da haben wir viel eindeutigere Stichworte gegeben. Die Atmosphäre war ganz angenehm, wenn auch nicht so super wie bei anderen Prüfungen. Der arme Mann hatte auch elf Leute zu prüfen. Keine Ahnung, wie die letzte davon gelaufen ist.

Material

Zentral:

- Jarke, *Folienkopien zu Datenbanken I & II*, WiSe 94/95, SoSe 95
- K. Pohl, *Folienkopien zu Requirements Engineering*, SoSe 95
- Eigene Mitschrift zu RE, Datenbanken I & II
- G. Vossen: *Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme*, Addison-Wesley ^{2}1994, 1. korrigierter Nachdruck 1995

Nützlich:

- *Ingres Database Administrator's Guide*, Kapitel ?: "Ingres Locking", Kapitel ?: "2-Phase-Commit" (Verteiltes Schreiben), Ingres 1993 (ca.), zu Version 6.04

Interessant:

- *Ingres SQL Reference Manual*, Kapitel 6: "Ingres Features", Ingres 1993 (ca.), zu Version 6.04
- *Oracle SQL*Plus User's Guide*, Kapitel 20: "Data Sharing and Security" (Locks), Oracle ^{2}1987, zu Oracle Version 5.1, SQL*Plus Version 2.0

Eher überflüssig: (VSe war ja gar nicht dran)

- C. Popien: Skript *Verteilte Systeme*, SoSe 95
- Eigene Mitschrift zu Verteilte Systeme
- W.Rosenberger, D.Kenney, G.Fisher: *Understanding DCE*, O'Reilley&Associates ^{2}1992
- C.Hunt: *TCP/IP Network Administration*, Ein Nutshell Handbook, O'Reilley&Associates ^{5}1994

Und dann habe ich noch einiges in HTML-Dateien gesammelt und verknüpft. Im Wesentlichen die DB-Vorlesungen. Alles ohne jegliche Gewähr. Momentan im Netz unter `http://www.imib.rwth-aachen.de/www/mitarb/fxg/Definitionen` zu haben.

Prüfungsprotokoll
Praktische Informatik bei Prof. Jarke

EDB, IDB, BS I + II

Prüfungstermin: 30.9.1999
Note: 1.0

1 Material

- Navathe, Elmasri, Fundamentals of Database Systems
- Vossen, Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme
- Silberschatz, Galvin, Operating System Concepts
- Folienkopien EDB und IDB (nicht verzweifeln :-))

2 Die Prüfung

2.1 EDB

Lebenszyklus von Datenbanken

- Requirements Engeneering
- Entscheidung für ein Datenmodell
- Modellierung der Datenstrukturen
- Vergleich zur Vorgehensweise bei anderen Systemen (Höhere Bedeutung der Datenstrukturen)
- Rückgriffe
- (E)ER, UML, alternativ Universalrelation aufstellen und FDs suchen, Dekomposition oder Synthese (nur kurz angesprochen)

Vergleich (E)ER mit UML

- Entitäten vs. Objekte/Klassen
- Beziehungen
  - ER: Beziehungen mit Attributen, Generalisierung/Spezialisierung, Kategorien
  - UML: Assoziation, Generalisierung/Spezialisierung, Aggregation
  - Läßt sich Aggregation auch im ER-Modell ausdrücken? Ja, Simulation über Assoziationen, aber etwas andere Semantik, da in UML ein Objekt nur an einer Aggregation teilnehmen kann, dieser Constraint läßt sich in ER nicht ausdrücken.

ER-Modell basteln

- Entitäten: Firma, Stadt, Angestellter
- Beziehungen:
  - Firma-Stadt(Niederlassung, m:n)
  - Angestellter-Stadt(Wohnt, 1:n)
  - Firma-Angestellter(arbeitet, 1:n)
- Diagramm siehe Abbildung 1
- Überführen in Relationales Modell (Generelle Vorgehensweise erklärt, zwei Möglichkeiten für 1:n Beziehungen: extra Tabelle, oder Beziehung in Tabelle der "1-Seite´´ einbauen).
  - Angestellter (Name, FName, Gehalt, SName)
  - Firma (Name, Branche),
  - Stadt (Name)
- SQL-Anfrage: In welcher Stadt verdienen die Angestellten am meisten? Geht nicht, da nicht eindeutig ist, in welcher Niederlassung ein Angestellter arbeitet.
- Also: wo wohnen die reichsten Angestellten?

```
SELECT Stadt.Name, AVG (Gehalt) FROM Angestellter, Stadt
WHERE Angestellter.SName = Stadt.Name
GROUP BY Stadt.Name
ORDER BY AVG(Gehalt) DESC
```

- Frage von Prof. Jarke: ist diese Anfrage aus Implementierungssicht optimal formuliert? Dachte, er will auf Anfrageoptimierung hinaus, war aber nur überflüssiger Join (SName ist ja direkt das gesuchte Feld aus der Stadt-Relation)

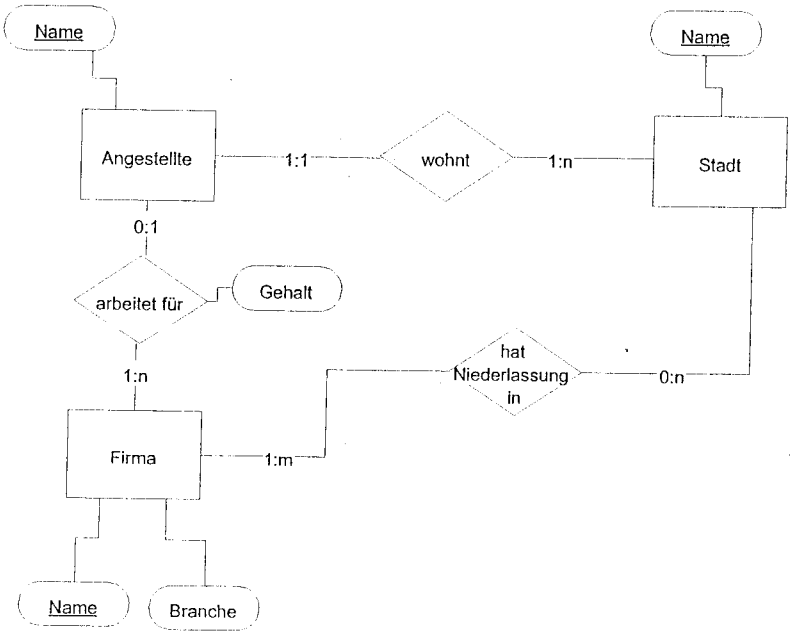


Abbildung 1: ER-Diagramm

2.2 IDB

Anfrageoptimierung

- Zerlegung in gutartige und bösartige Komponenten, Quantgraphen
- gutartig: Auswertbar mit Semijoin (Generalisierte Semijoinausdrücke)
- Vorteil von Semijoin gegenüber Join: Ergebnis wird kleiner (worst case: bleibt gleich, Join: worst case ist kartesisches Produkt), Vorteile bei der Implementierung

Transaktionen

- Read-Write-Modell
- Grund für verzahnte Ausführung von Transaktionen (Performance, Auslastung ...)
- Scheduler: Beeinflußt eintreffenden Strom von Aktionen, so daß ausgehender Strom äquivalent zu seriellem unter gewährleistung der Fehlersicherheitskriterien
- FSR: Final State Serialisierbarkeit: Herbrandsemantik der letzten Schreib-OP je Objekt / Life-Reads-From-Relation; nicht effizient überprüfbar, Phantom-Probleme können auftreten
- VSR: View Serialisierbarkeit: Herbrandsemantik aller OPs / Reads-From-Relation; nicht effizient prüfbar, nicht PCA
- CSR: Konfliktserialisierbarkeit: Konflikt-Relation/Graph; PCA, effizient prüfbar, Problem: Dirty Read
- Fehlersicherheit: RC, ACA, ST, RG (kurz angesprochen)
- Scheduler
  - sperrend vs. nicht-sperrend
  - 2PL
  - TL, MGL (nur kurz erwähnt)
  - strenges 2PL: (CSR und ST)
  - konservatives 2PL: Deadlockfrei, aber nicht praxistauglich (woher soll der Scheduler am Anfang wissen, was er sperren muß)

2.3 BS I + II

Deadlocks

- Deadlock-Bedingungen: Mutual Exclusion, No Preemption, Hold and Wait, Circular Wait
- Prevention:
  - versuchen, eine der Bedingungen auszuschließen
  - Mutual Exclusion: schwierig
  - No Preemption: Verfahren, bei denen Prozessen Resourcen abgenommen werden, ebenfalls schwierig
  - Hold and Wait: Konservatives 2PL
  - Circular Wait: Ordnung der Resourcen, die beim sperren eingehalten werden muß: TL
- Avoidance: Zusatzinformationen über zukünfitges Programmverhalten ermöglichen Überprüfung, ob System in sicherem Zustand, z.B. Bankers Algorithmus
- Detection und Recovery (nur kurz)

Speicherhierarchie

- Prof. Jarke fragte nach den Übergängen in der Speicherhierarchie, habe ihm erstmal was über Caching erzählt, er wollte aber eigentlich auf Paging hinaus.
- Demand Paging, Virtual Memory erläutert

2nd level Speicher

- was macht man, wenn man viele Platten hat und sequentiell lesen will? Disk Striping erläutert.
- RAID

3 Und sonst?

Die Atmosphäre war (für eine Prüfung) sehr angenehm, Prof. Jarke hat bei Bedarf hier und da auch schonmal ein bißchen geholfen. Während ich das Relationenschema gebastelt habe, habe ich mir den Mund fusselig geredet und jeden Schritt den ich gemacht habe erklärt, das hat ihn aber offensichtlich nicht interessiert, jedenfalls hat er in der Zeit in den Unterlagen der andern Prüfungen herumgeblättert und -gekritzelt.

Verglichen mit den (relativ alten) Prüfungsprotokollen, die ich gelesen habe, hatte ich in meiner Prüfung den Eindruck, daß er nicht mehr so viel Wert auf Formalismen und runtergeschriebene Details legt. Kann aber auch daran liegen, daß wir z.B. die Theorie zu relationalen Datenbanken (Schemasysnthese etc.) nur relativ kurz angesprochen haben.

Man sollte sich von den Jahrzehnte alten Vorlesungsunterlagen nicht dazu verleiten lassen, neuere Aspekte zu vernachlässigen, ich war einigermaßen verblüfft, als er doch recht genau auf UML einging (ich war nicht in der Vorlesung, und den UML-Abschnitt im Skript bildeten ein paar Folien aus einem anderen Vortrag).

Bis auf wenige Ausnahmen (Anfrageoptimierung und Transktionsverarbeitung) sind die Folienkopien die schlechtesten, die ich je gesehen habe!

Viel Erfolg!

Gedächtnisprotokoll
Diplomprüfung Informatik
Vertiefungsgebiet Informationssysteme

Prüfer:	Prof. Jarke
Datum:	25.11.96
Dauer:	ca. 45 Minuten
Fächer:	Einführung in Datenbanken
	Implementierung von Datenbanken
	Konzeptuelle Modellierung (Jeusfeld)
	Wissenbasierte Informationssysteme (Jeusfeld)

Note: 1.3

1 Einführung in Datenbanken/Konzeptuelle Modellierung

- Jarke beschreibt das Szenario einer Bibliothek, zunächst sehr allgemein mit den Begriffen *Mensch* und *Veröffentlichung* bzw. *Buch*. Die Beziehungen sind: Menschen können Bücher lesen oder schreiben. Dazu soll ich ein ER-Modell aufstellen.
- Die Kardinalitäten sollen auch in (min,max)-Notation angegeben werden.
- Die Entität Mensch soll spezialisiert werden in Leser und Autor[1]. Die Beziehungen gehen nun zu Autor und Leser.
- Das ER-Modell soll ins Relationale Datenmodell übertragen werden. Ich bilde für jede Entität und Relationship eine Relation. Die Relation *Autor* ist offensichtlich eine Teilmenge von der Relation *Mensch*. Jarke will wissen, wie man verhindert, das Autoren doppelt gespeichert werden müssen. Dazu kann man in SQL *Sichten* definieren (Autoren sind Mensch, die schon mal ein Buch geschrieben haben). Ich habe also eine Query dafür hingeschrieben.
- Die gleiche Query soll ich auch noch in relationale Algebra übertragen. Die Query kann durch einen *Semi-Join* ausgedrückt werden.
- Jarke will wissen, wie man herausfinden kann, ob ein Attribut ein Schlüssel ist (d.h. jeder Attributwert tritt nur einmal in der Relation auf). Meine Lösung sieht so aus (es ging hier konkret um das Attribut Lesernummer, der Relation Leser):

```
SELECT LNr, COUNT(*)
FROM Leser
GROUP BY LNr
HAVING COUNT(*) > 1
```

[1] Hier kam also EER ins Spiel, die einzige Frage die sich auf die beiden Jeusfeld-Vorlesungen bezog

- Jarke fragt, wie man Anfragen denn optimiert. Ich erzähle was über heuristische Anfrageoptimierung und Joinreihenfolgeoptimierung. Jarke fragt dann noch Parametersystemen, will die Voraussetzungen dafür wissen usw.

2 Implementierung von Datenbanken

- Hier findet dann auch der gleitende Übergang in die Implementierung statt. Anhand des 5-Schichtenmodells soll ich erklären, wo die Anfrageoptimierung stattfindet. Dazu muß ich zunächst das Schichtenmodell aufzeichnen und die einzelnen Schichten (oder besser gesagt: die Schnittstellen) erklären.

- Jarke geht zur Segmentschnittstelle und will was über Blöcke, Dateien, Records, Blockgröße und Blockfaktor hören. Er will wissen auf will Blöcke man best/average/worst case zugreifen muß, wenn man k von insgesamt N Records lesen will, die auf b Blöcke verteilt sind (Das steht im 2. Kapitel der Einführung, aber das ist auch schon alles was ich darüber weiß).

- Dann war noch Recovery gefragt: Wann ist Redo, wann ist Undo nötig?

- Zum Schluß noch die verschieden Klassen für Schedules und deren Zusammenhang erklären (insbesondere RG und CSR, aber auch VSR, FSR, ST, ACA und RC wichtig).

3 Fazit

Die Prüfungsatmosphäre war entspannt und angenehm. Mit der Benotung war ich zufrieden. Was an der 1.0 gefehlt hatte, waren ein paar kleinere Hänger zwischendurch und die Unkenntnis bei Worst/Average/Best Case für den Blockzugriff.

Datum:	29. September 1995
Prüfer:	Prof. Dr. M. Jarke
Stoffumfang:	Einführung in Datenbanken, Implementierung von Datenbanken, Betriebssysteme
Literatur:	Kopien aus der Vorlesung (DB), Operating System Concepts (Fourth Edition) von Silberschatz und Galvin
Note:	1.7

Betriebssysteme:

Jarke: In Betriebssystemen gibt es so eine Sache, die heißt Zuteilung von Betriebsmitteln. Das wichtigste Betriebsmittel ist die CPU. Wie wird denn die CPU zugeteilt?

ich: CPU-Scheduling, aber nur bei Multitasking-Systemen.

Jarke: Es gibt da so ein Diagram...

ich *[habe das typische Statusübergangsdiagramm hingemalt und erklärt.]*

Jarke: Wir haben ja auch andere Betriebsmittel, die nicht so gesondert behandelt werden wie die CPU, und die vielleicht auch mehrfach vorhanden sind — da kann es zu Problemen kommen. Machen Sie das mal an einem Beispiel deutlich, dem Consumer–Producer Problem.

ich: Das Problem ist, wenn zwei Prozesse auf eine Resource gleichzeitig zugreifen wollen. Beim Consumer–Producer Problem ist das ein Puffer, in den der Produzent schreibt und aus dem der Konsument liest. Da muß man darauf achten, daß der Puffer nicht überläuft und daß er beim Lesen nicht leer ist. Man kann das mit Monitoren lösen.

Jarke: Und wenn man so etwas edeles wie Monitore nicht zur Verfügung hat?

ich: Man kann auch Semaphoren benutzen.

Jarke: Schreiben Sie doch mal Code auf für einen Produzenten und einen Konsumenten.

ich: *[Irgendwas gefaselt, neue Interpretationen für signal() und wait() erfunden, war aber falsch.]*

Jarke: Einer Ihrer Vorgänger hatte da so einen schönen Fehler gemacht, den haben Sie jetzt nicht gemacht. Wenn Sie diese beiden wait() vertauschen, dann kann es Probleme geben...

ich: Ja, einen Deadlock. *[Dann die vier Bedingungen aufgezählt und etwas zur Behandlung gesagt.]*

Implementierung von Datenbanken:

Jarke: Wo können denn in Datenbanken Deadlocks auftreten?

ich: Beim Zugriff auf die Daten, wenn Transaktionen parallel abgearbeitet werden und man einen sperrenden Scheduler verwendet. Man kann mit dem konservativen 2PL deadlockfreie Schedules erzeugen, wenn man vorher weiß, welche Resourcen benötigt werden.

Jarke: Wie behandeln denn moderne Datenbanken den Deadlock?

ich: Wie viele Betriebssysteme auch: don't care. Wenn es mal zu einem Deadlock kommen sollte, dann kann man ja die Transaktion abbrechen.

Jarke: Wir hatten da in der Vorlesung ein Bild zu den Problemen, die beim Zugriff auf Daten auftreten können.

ich *[hatte keine Ahnung von was er redet.]* Es gibt Probleme wie dirty–read und lost–update...

Jarke: Das meine ich nicht. *[Malt ein paar Linien auf sein Papier.]*

ich *[erkenne den Systemabsturz während Transaktionen wieder.]* Aha! *[Dazu was erzählt. So sind wir auf Recovery mit Log gekommen.]*

Jarke: Wie geht das denn genau mit dem Log und Recovery und wo liegt das Log und wann schreibt man was und wie funktioniert das alles und wann braucht man es und überhaupt? Sagen Sie mal!

ich: *[irgendwas gesagt.]*

Jarke: Warum machen heutige Datenbanksysteme dieses Log? Das ist doch mehr Aufwand als

wenn man den geänderten Puffer beim Commit einfach auf die Platte schreibt.

ich: Wenn man mehrere Puffer zusammen zurückschreibt, dann hat das Geschwindigkeitsvorteile bei den neuen tollen Platten.

Jarke: Fallen Ihnen sonst noch Gründe ein?

ich: Wenn die Platte kaputtgeht, kann man Daten aus dem Log rekonstruieren. Und bei virtuellem Memory (des Betriebssystems) hat man gar nicht die Möglichkeit, genau zu überprüfen, wann der Puffer geschrieben wird. Da wird dann wieder eine Recovery nötig.

Jarke wollte dann was zu virtuellen Maschinen wissen, das konnte ich ihm aber nicht sagen.

Jarke: Welche Operationen bietet die relationale Algebra?

ich: select, project und join.

Jarke: Sind das alle?

ich: Glaube schon. Ach ja, es gibt noch Schnitt und Vereinigung. Aber die liefern komische Ergebnisse, wenn man die falschen Relationen reintut.

Jarke: Was sind denn richtige Relationen?

ich: Gleiche Attribute.

Jarke: Welche Besonderheit hat denn die Algebra?

ich *[habe keine Ahnung.]*

Jarke: Sie haben eben schon erwähnt, daß bei Schnitt und Vereinigung 'komische' Ergebnisse rauskommen können...

ich: Ach ja, Abgeschlossenheit. Da kommen immer wieder Relationen raus.

Jarke: Ja, genau. Dadurch ist es möglich, Operatoren zu schachteln. Wenn Sie jetzt an die Implementierung von SQL denken, welcher Operator liefert nicht immer eine Relation?

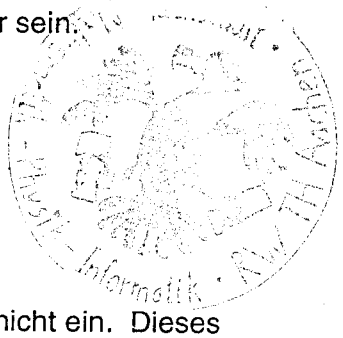
ich: Exists — oder auch count(), avg() und so.

Jarke: Nein, diese Operatoren geben wieder eine Relation zurück. Sie können auf ein count() wieder ein count() anwenden. Welche Operation kann denn teuer werden?

ich: Join.

Jarke: Falsch. Einer aus drei. Daß der Join teuer ist, wissen wir alle. Ich meine einen anderen Operator, bei dem man nicht unbedingt damit rechnet.

ich: Projektion. Wegen der doppelten Elemente. Die zu entfernen, kann teuer sein.



Jarke: Ja, wie teuer denn nun?

ich: Wenn's sortiert ist, dann ist das einfacher.

Jarke: Die Kosten.

ich: Linear, wenn die Relationen sortiert sind.

Jarke: Und sonst?

ich: Sonst muß ich's vorher noch sortieren, also n * log n.

Jarke: Gut. Welche Arten gibt es, den join zu implementieren?

ich: Nested Loop, Hashjoin, den zweiten weiß ich nicht, da fällt mir der Name nicht ein. Dieses Verfahren geht davon aus, daß die Listen sortiert sind.

Jarke: Gut, das hatten Sie eben schon beschrieben *[er meint vermutlich beim Project]*. Erklären Sie doch mal, wie der Hashjoin funktioniert.

ich: Die beiden Relationen werden mit einer Hashfunktion auf Buckets verteilt und dann stehen die zusammengehörenden Tupel in den Buckets drin und man ist fertig. Wenn die Hashfunktion nicht so toll ist, dann muß man noch in den Buckets nach den richtigen Tupeln suchen.

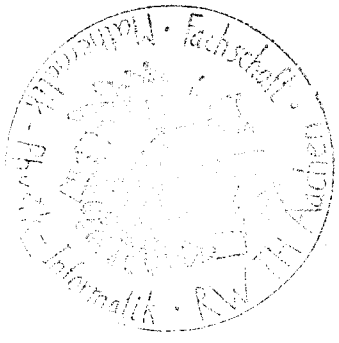
Einführung in Datenbanken:

Jarke: Da sind auch schon fast fertig. Noch eine Frage aus der Einführungsvorlesung. Nennen Sie die Armstrong-Axiome!

ich: Reflexivität, Erweiterung, Transitivität.
Jarke: Von was denn?
ich: Von funktionalen Abhängigkeiten (FDs). Also: A bestimmt B und B bestimmt C, dann bestimmt A auch C, das ist die Transitivität; A, B und C sind Mengen von Attributen. Die Erweiterung: A bestimmt B dann auch AE bestimmt BE, auch hier A, B und E Mengen von Attributen; Reflexivität: A Teilmenge von B bedeutet B bestimmt A.
Jarke: Was ist denn toll an den Axiomen?
ich: Abgeschlossenheit.
Jarke: Häh?
ich: Vollständigkeit.
Jarke: Häh? Naja, sind die Axiome denn wirklich Axiome?
ich: Ja.
Jarke: Nein. Man kann sie beweisen.
ich: Aha.
Jarke: Und für was braucht man denn nun die Armstrong'schen Axiome?
ich: Verlustfreier Join.
Jarke: Nein.
ich: Anfrageoptimierung.
Jarke: Nein, nein, der verlustfreie Join war schon nah dran. Da in der Nähe war das. Mit Normalformen und so.
ich: Erhaltung der funktionalen Abhängigkeiten.
Jarke: Richtig.
ich: Man kann bei den Normalformen überprüfen, ob die Menge der Funktionalen Abhängigkeiten äquivalent bleibt.
Jarke: Ja, das war's dann. Wenn Sie draußen warten würden...

PS: Jarke macht in der Prüfung nicht ganz den konfusen Eindruck, den er in der Vorlesung vermittelt. Trotzdem hampelt er auch beim Fragenstellen ziemlich rum und spielt ständig an seiner Uhr rum. (Vielleicht geht die deshalb ab und an kaputt :-)
Naja. Die Atmosphere war dennoch gut, und Jarke bohrt nicht lange in Wunden herum, wenn man sie zu erkennen gibt (inwieweit das für die Note abträglich ist, kann ich nicht beurteilen). Für eine gute Note scheint es auszureichen, über den Stoff Bescheid zu wissen. Man muß nicht unbedingt die Fragen sofort verstehen.
PPS: Obwohl viele der Prüfungsprotokolle, die ich mir vor der Prüfung angesehen habe, sehr rudimentäre Fragen und Antworten vorgaukeln und scheinbar jeder mit mindestens 2.0 besteht, war vor mir einer in der Prüfung, der nicht bestanden hat. Jarke scheint auch (inzwischen) sehr auf das Detailwissen abzufahren (wenn das in vielen Protokollen und in meinem vermutlich auch nicht so aussieht).

Und: Viel Erfolg für Deine Prüfung.



Gedächtnisprotokoll Diplomprüfung Informatik

Prüfer : Prof.Jarke
Fach : Vertiefungsgebiet Informationssysteme/Wissensbasierte Systeme
Inhalt : Einführung in Datenbanken (Vorlesung Jarke)
Implementierung von Datenbanken (Vorlesung Jarke)
Verteilte Datenbanken (Vorlesung Jeusfeld)
Konzeptuelle Modellierung (Vorlesung Jeusfeld)
Termin: April 1995
Dauer : 55 Minuten

Einführung in Datenbanken:

Was ist ein Relationenmodell? /
Structures,Operations,Constraints
Welche Operationen gibt es beim relationalen Datenmodell? /
Vereinigung,Durchschnitt,Restmenge,Kreuzprodukt,
Selektion,Projektion,Join
Welche Komplexität haben Join,Projektion und Selektion? /
Welche Join-Verfahren gibt es? /

Implementierung von Datenbanken:

Wie optimiert man Join-Anfragen?
Mit Quantgraphen gutartige und bösartige Ausdrücke ermitteln
Welches sind die Hauptkomponenten bei der Transaktions-Verwaltung?
Serialisierung,Fehlerrecovery
Worum geht es bei der Serialisierung und welche Arten gibt es?
Final-State-, View-,...
Welche Eigenschaften sollte ein Schedule zusätzlich noch haben?
Rücksetzbarkeit, kaskadierende Aborts vermeidend, strikt, rigoros
Was ist 2-Phasen-Sperren, welche Schedule-Eigenschaften erreicht man dadurch und welche Varianten gibt es?
Konservatives und strenges 2-Phasen-Sperren
Schichtenmodell:Schnittstellen erklären.
Wo ist die Tranaktionsverarbeitung im Schichtenmodell einzugliedern?
Parallel zu den einzelnen Schichten
Angenommen, man möchte die Tranaktionsverarbeitung wegen des großen Aufwandes bei der Implementierung parallel zu allen Schichten stattdessen nur in einer Schicht realisieren.Welches Problem kann auftreten, wenn zwei Transaktionen auf zwei verschiedene Sätze zugreifen?
Sätze können auf derselben Seite liegen

III. Expertensysteme

Frage: *Was ist der Unterschied zwischen Hornklausen und EFRS ?*

EFRS enthalten spezielle Hornklausen, insbesondere solche ohne Funktionssymbole. Die Menge der Fakten, die aus einem EFRS logisch folgen, ist endlich groß im Unterschied zu Hornklausenmenge.

Frage: *Schreiben Sie mal eine Hornklausenmenge auf, aus der unendlich viele Fakten folgen*!

{P(A), P(x) => P(f(x))}

Frage: *Ist Prolog nichtmonoton ?*

Da die Negation als "negation as failure" bearbeitet wird, ist sie bzgl. der Negation nichtmonoton.
Bsp.: Ist P(A) nicht in der Datenbasis, so ist not(P(A)) TRUE, fügt man P(A) ein, so ist not(P(A)) FALSE.

Frage: *Welchem nichtmonotonem Schließen entspricht das ?*

CWA.

Frage: *Wie kann man in Prolog die Negation simulieren, ohne dabei "not" zu benutzen ?*

Mit tatkräftiger Unterstützung von Prof. Nejdl leiten wir folgendes her:

```
not(A) :- A & ! & fail.
not(A):- true.
```

(Ich wollte es auch nicht glauben, aber es funktioniert tatsächlich !)

Allg. Bemerkungen zur Prüfung:

Die Prüfung begann etwa 20 min später als vereinbart, was allerdings bei ihm nichts besonderes ist. Die Prüfungsatmosphere empfand ich als angenehm und freundlich. Prof. Nejdl fragte nicht in die Tiefe, sondern ging in den Büchern nur nach deren Inhaltsverzeichnis vor und fragte grob die Inhalte mancher Kapitel ab. Sollte man einmal nicht weiter wissen, so läßt Prof. Nejdl einem einerseits Zeit zum Überlegen, hilft einem andererseits dann aber auch weiter. Prof. Nejdl fragte auch nach Inhalten, die er gemäß Vereinbarung nicht abfragen wollte. Das sollte man ihm dann auch mitteilen.

Gedächtnisprotokoll Diplomprüfung "Praktische Informatik", Prof. Jarke

---

Fächer : **Einführung in Datenbanken (Jarke)**
(+ Unterlagen) seine Folienkopien ...

Betriebssysteme
Silberschatz "Operating System Concepts" : §1-12 + §19 (Unix)

Expertensysteme (Motschnig)
Vorlesung vom WS 95/96, Folienkopien als Unterlagen

Datum : **22.1.1996**

Note : **2.3**

Betriebssysteme :

"Wie hat das denn so angefangen mit der Speicherverwaltung ? Da gibt's doch eine historische Entwicklung ?"

Single Partition Allocation, Multiple Partition Allocation (-> Problem, freien Speicherbereich zu finden, externe Fragmentierung), schließlich deshalb Paging (Aufteilen des Speichers in gleich große Teile (Frames), log. Adressraum wird ebenfalls in gleich große Teile (Pages) aufgeteilt

"Wie funktioniert denn das so mit dem Paging ?. Malen sie doch mal ein Bildchen !"

CPU generiert log. Adresse, PTBR zeigt auf PageTable, mit deren Hilfe wird der Frame im MainMemory ermittelt, in dem die Seite abgelegt ist.

"Stichwort : PageTable. Was ist das ? Wie funktioniert das genau?"

Zuerst behauptet, in jedem Eintrag der Table liegt die PID und PageNr, die in dem zug. Frame (Nr in der PageTable) eingetragen ist. Auf Jarkes Äußerung, man könne doch nicht für jeden Frame einen freien Eintrag halten (da sonst PageTable zu groß), habe ich schließlich die PageTable korrigiert : Also zu jeder Page des Prozesses ist die Frame-Nr eingetragen, in dem sich die zug. Page befindet.

"Es gibt doch nicht nur eine PageTable ?"

Jeder Prozeß hat eigene PageTable. Damit wird dann vom Dispatcher die Hardware-PageTable belegt.

"Wie läuft denn so ein PageFault ab ?"

Zuerst überprüfe ich (*"Sie?"* Nein, das OS :-)), ob ein Adressierungsfehler vorliegt. Wenn ja, Fehler durch OS. Ansonsten Memory-Management auffordern, Seite einzulagern.

"Was passiert mit einem Prozess, wenn eine Seite eingelagert werden muß ?"

... er wird interrupted und ... wartet ? (Das war das Richtige !) Prozess-Status Bildchen erläutert.

"Warum wartet er ? Warum geht er nicht direkt wieder in den Zustand 'ready' ?"

Um anderen Prozessen die Möglichkeit zu geben, während der Zeit des Einlagerns der Seite die CPU zu nutzten. Jetzt kam eine Frage, die mich ein bißchen verunsicherte :

"Haben Sie eine Ahnung, um welchen Faktor (1, 5, 100, 1000000) sich der Zugriff auf die Seite aufgrund des PageFaults verzögert ?"

Hm, da geht's um Millisekunden ... Hatte einen Faktor 10 im Kopf (irgendwoher aus dem Bereich Stackalgorithmus). Das war's aber nicht. 100-1000 meine Jarke.

sonderlich relevant. Als es Prof. Jarke zu langweilig wurde meinen krampfhaften Versuchen länger zuzuschauen, hat er netter Weise das Thema gewechselt.

Betriebssystem und Implementierung

Wie läuft die Prozeßverwaltung so ab?

Prozeßzustandsdiagramm aufgemalt, mehrere Warteschlangen erläutert, CPU-Scheduler wählt Prozesse aus der ready-Queue.

Was ist beim Prozeßwechsel zu tun?

context-switch: Register und Programmzähler retten, evtl. page table sichern

Sie erwähnten gerade die virtuelle Speicherverwaltung und Paging, erklären Sie bitte!

homogener, linearer Adreßraum; Abbildung logischer auf physische Adressen durch page table; page fault

Welche Seitenersetzungsstrategien gibt es?

FIFO, OPT, LRU, SECOND-CHANCE

Wo findet diese virtuelle Speicherverwaltung in der Fünf-Schichten-Architektur ihre Entsprechung?

Segmentschnittstelle

Warum nützt man für das DBMS nicht die virtuelle Speicherverwaltung des BS?

Portabilität, Seitenersetzungsstrategie des BS evtl. nicht adäquat (recovery), Schattenspeicher

Jetzt zur Transaktionsverwaltung, was ist eine Transaktion?

read-write-Modell, ACID-Prinzip

Was ist die Aufgabe eines Schedulers?

verzahnte Ausführung von Transaktionen, Korrektheit

Wie ist die Korrektheit definiert?

Äquivalenz zu seriellem Schedule

Warum führt man die Transaktionen verzahnt aus?

Lange TA können sonst das ganze System blockieren; er wollte hören: Wartezeiten von TA (oder Prozessen in BS) können von anderen TA genutzt werden. Also: Insgesamt wird die Verarbeitung gegenüber dem sequentiellen Fall beschleunigt.

Welche Serialisierbarkeitsbegriffe gibt es?

FSR, VSR, CSR, OCSR, CO

Wie erreicht ein Scheduler Konfliktserialisierbarkeit?

kurze Einführung: sperrende vs. nichtsperrende Scheduler; 2PL-Protokoll

Was ist das Problem dabei?

Deadlocks; Vermeidung durch konservatives 2PL; vier Bedingungen für Deadlocks genannt

Welche Ansätze zur Behandlung von Deadlocks gibt es?

Protokoll zur Diplomprüfung am 8.11.91 5597

bei Prof. Dr. M. Jarke über

(1) Informationssysteme von Herrn Jarke
(2) Implementierung von DB — " —
(3) Wissensbasierte Info.-syst. von Herrn Udo Hahn
(4) Objektorientierte DB von Herrn Unland

zu (3) – Aufbau eines EXP? ✓
- Aufbau von MYCIN?
[Konsultationsteil = Problemlösungsteil + Dialogkomp., Rest wie EXP]
- wie ist hier das Wissen gespeichert?
[med. Wissen in Konzeptklassen/-typen + Prod.-regeln]
- wie werden Regeln ausgeführt? ✓
- Erklärung des Backward-Chaining? ✓
- Arten der Wissensrepräsentation?
[Logik, Sem. Netze, Frames, Prod.-systeme]
- Unterschied Frames – Objektmodellierung (in DB)??
["fast gleich", nur Frames unterstützen prototypische Erwartungsw.]
- Hinzeichnen eines sem. Netzes mit 'is_a' Bez. u. Eigenschaftskanten
- Was ist die Semantik der 'is-a'-Beziehung??
[sagte was über 'Vererbung', 'Unterklasse', 'Teilmenge', ganz zufrieden war er damit aber nicht, ferner 'Spezialisierung' o.ä.?]

(4) – Wandeln Sie dieses Netz in ein relat. DB-Schema um??!
[je Konzeptklasse 1 Relation, Schlüssel jew. die gleichen?!, Eigenschaftskanten als Attribute]
- 'Hauptgrund' für OODB?
[Zersplitterung von Objekten in rel. DB ⇒ viele Joins notwendig!!]
- Objektidentität in rel.- OODB?
[in rel. DB keine, in OODB Surrogate?]
- Was wissen Sie über O_2?
["wenig, das haben wir nur als Bsp. gemacht", dann gabs Regeln]
- Was ist eine Transaktion? [→ Def. in DB-Handbuch]
- Welche Besonderheiten diesbzgl. in OODB?
[lange und geschwätzig]

- Was ist geschedult. T_i, Bsp? ✓
- Warum – " – ?
  [wg. Komplexen abgebrochen?, reichte ihm]

(2) – Was ist das 2 PL? ✓
- Nachtal?
  [pessimistisch + Deadlocks mögl.]
- Def. von Serialisierbarkeit? ✓
- Wann ist T_i serialisierbar? – Zeichnung eines Präzedenzgraphen?
  [kein Zyklus im P-Graph, Totalordnung überlege von Logs]
- Auswertungsverfahren für JOIN-Operator? Anz. # Lesezugriffe? ✓
- Def. von Semijoin?
  [Proj. des Join auf Attr. der "1. Rel.", sodass r_1 in R_1, some r_2 in R_2 : $(r_1.a = r_2.b)$]
- Vorteile der Semijoinauswertung? ✓

(1) – Def. von 1., (2., 3., BC-NF? ✓
- Probleme bei BCNF ✓
- Ist 3. NF auch abhängigkeitserhaltend ??? [→ Nein!]
- Test auf Verlustfreiheit?
  [Tableaux-Test, "chase-Alg."]
- Machen Sie doch mal ein Bsp? [→ Stimpf]
- Was für "bessere" Methoden gibt es? [Syntheseverf.]
- Erklären Sie das Syntheseverfahren (theoretisches)??
  ["ah, es gibt Äquivalenzklassen für Attr., und dann...", "das habe ich mir nicht mehr so genau angesehen"?]
  (Gabe erklärte das Verf. dann kurz)
- Vorteile dieses?
  [nur polynom. Zeitkompl., abhängigkeitserhaltend]

Bem.: angenehme, ruhige Gesprächsatmosphäre (ohne Hektik),
✓ breites Stoffgebiet, aber auch mit Detailfragen,
sehr gute Benotung?

Prüfungsprotokoll vom:	20. 1. 1994
Prüfer:	Jarke/Nejdl
Fächer:	Einführung DB (Jarke-Skript, Ullmann, etc.), Expertensysteme (Gottlob/Frühwirth), Wissensrepräsentation (Genesereth/Nilsson)
Dauer :	ca. 50-55 Minuten
Note :	1.7

Meine Prüfung war insofern etwas ungewöhnlich, als Jarkes erste Frage lautete, ob ich etwas dagegen hätte, wenn nicht er sondern Nejdl mir die Fragen zur Wissensrepräsentation und zu Expertensysteme stellen würde. Mir war's prinzipiell egal, und Nejdl wusste natürlich besser, was in der Vorlesung tatsächlich dran war, also war ich einverstanden (im Nachhinein wahrscheinlich eine gute Idee).

1 Datenbanken

Jarke: beschreibt Kunde-Waren-Einkauf- Zusammenhänge, möchte ER-Diagramm.
ich: male selbiges hin
Jarke: läßt Relationen aufstellen
ich: Relationen *Kunde*: (Kundennr), *Ware*: (Warennr, Preis, Gruppe) und *Einkauf*: (KNR, WNR, Menge)
Jarke: beschreibt Supermarkt-Szenario und will SQL-Anfrage, die die Anzahl der gekauften Produkte mit Warengruppe für einen Kunden auflistet
ich: (kriege die Anfrage mit Müh und Not hin)
Jarke: weitere SQL-Anfrage: Aufsummieren der Gesamtmengen, die von jedem Produkt gekauft wurden
ich: (habe mir SQL so genau nicht angeguckt, kann mich daher nur noch an die *sum*-Funktion erinnern, auf das *group by* bin ich nicht mehr gekommen)
Jarke: malt mir ein Rechnungsformular hin und möchte die entsprechenden Relationen in 1NF und 3NF (dabei soll ich die Normalformen erklären)
ich: erkläre NF, male 3NF hin (er will aber erst die 1NF, na gut). Leider macht das wohl keinen so souveränen Eindruck - er hat mich aber auch ziemlich durcheinander gebracht mit seiner SQL und diesem Beispielformular (das übrigens genau so im Skript steht, und das ich immer überschlagen hatte).
Jarke: Danke, das war's zu Datenbanken
ich: bin etwas platt, normalerweise fragt er da VIEL mehr zu - und darauf wäre ich auch vorbereitet gewesen

2 Expertensysteme

Nejdl: Welche Formalismen wurden zur Wissensrepräsentation vorgestellt ?
ich: PL1 und Untermenge davon, EFRS
Nejdl: Was ist der Unterschied ?

ich: (denk) EFRS haben keine Negation, keine Funktionensymbole (da war noch mehr, bloss was ?)
Nejdl: Was kann man in EFRS nicht ? ich: (überlege krampfhaft, mit einigen Tips von Nejdl) Disjunktion von pos. Literalen in einer Klausel nicht darstellbar, nur Hornklauseln möglich (?) Nejdl: Wie ist in Prolog die Negation realisiert ?
ich: erkläre *negation as failure*
Nejdl: will *Frames* erklärt haben
ich: male ihm einen hin, erzähle was von *AKO* und Vererbung, Möglichkeit, Vererbung zu blockieren
Nedjl: will Mehrfachvererbung erklärt haben
ich: male einen Frame *eierlegendes Tier* und einen *Säugetier*, darunter ein Schnabeltier: Eigenschaften beider Frames werden auf das AKO-Frame vererbt
Nejdl: fragt, was bei widersprüchlichen vererbten Eigenschaften (wie im Beispiel denkbar) passiert
ich: (verteidige erstmal die Existenz eines solchen Tiers ;) man muß sich überlegen, welche Eigenschaften man denn nun haben will (er sagt was von Strategie zur Auswahl der entsprechenden Eigenschaften, na ja, das meinte ich doch)
Nejdl: Was ist ein Meta-Interpreter ?
ich: erkläre das
Nejdl: Wie sieht der aus ?
ich: (bin langsam beeindruckt, wie gut sowohl Nejdl als auch Jarke genau die Themengebiete herausfinden, die ich nicht so gut drauf habe) male erstmal den Aufruf und die Regeln für Konjunktion, Disjunktion und Negation hin
Nejdl: und was noch ?
ich: *seufz* male noch die fehlenden hin, allerdings mit etwas Nachdenken.
Nejdl: Was für Inferenzstrategien gibt es ? ich: (höre Inferenz, bin momentan auf dem völlig falschen Dampfer, weil ich an Inferenz*netze* denke) ???
Nejdl: (legt mir das Inhaltsverzeichnis seines Buches hin) Haben sie da schon mal reingeguckt ?
ich: (werde langsam sauer) Ja, hab ich. (Später ist mir aufgefallen, daß die Frage ihre Berechtigung hatte — die EFRS, mit denen ich auch etwas Probleme hatte, kamen kaum auf den Vorlesungsfolien vor, waren aber ziemlich ausführlich im Buch beschrieben.)
Nejdl: Also, dann beschreiben Sie mal die Verfahren.
ich: beschreibe Vorwärtsverkettung
Nejdl: Ist die Menge *Cons*(S) beim EFRS immer endlich ?
ich: Ja, da nur endlich viele Ausgangsvoraussetzungen existieren und keine Rekursion (er guckt komisch), ich meine, Funktionssymbole treten nicht auf und können daher auch nicht geschachtelt werden, was zu unendlichen *Cons*-Mengen führen würde.
Ich erkläre auch noch die Rückwärtsverkettung (Stichwort Prolog), Resolution.
Nejdl: (zeigt auf das Beispiel, das mir bei diesen EFRS Probleme gemacht hatte) Resolvieren Sie das mal - genau wie Prolog das macht.
ich: erwähne beim Resolvieren, daß Prolog Tiefensuche durchführt und daß man

sonderlich relevant. Als es Prof. Jarke zu langweilig wurde meinen krampfhaften Versuchen länger zuzuschauen, hat er netter Weise das Thema gewechselt.

Betriebssystem und Implementierung

Wie läuft die Prozeßverwaltung so ab?

Prozeßzustandsdiagramm aufgemalt, mehrere Warteschlangen erläutert, CPU-Scheduler wählt Prozesse aus der ready-Queue.

Was ist beim Prozeßwechsel zu tun?

context-switch: Register und Programmzähler retten, evtl. page table sichern

Sie erwähnten gerade die virtuelle Speicherverwaltung und Paging, erklären Sie bitte!

homogener, linearer Adreßraum; Abbildung logischer auf physische Adressen durch page table; page fault

Welche Seitenersetzungsstrategien gibt es?

FIFO, OPT, LRU, SECOND-CHANCE

Wo findet diese virtuelle Speicherverwaltung in der Fünf-Schichten-Architektur ihre Entsprechung?

Segmentschnittstelle

Warum nützt man für das DBMS nicht die virtuelle Speicherverwaltung des BS?

Portabilität, Seitenersetzungsstrategie des BS evtl. nicht adäquat (recovery), Schattenspeicher

Jetzt zur Transaktionsverwaltung, was ist eine Transaktion?

read-write-Modell, ACID-Prinzip

Was ist die Aufgabe eines Schedulers?

verzahnte Ausführung von Transaktionen, Korrektheit

Wie ist die Korrektheit definiert?

Äquivalenz zu seriellem Schedule

Warum führt man die Transaktionen verzahnt aus?

Lange TA können sonst das ganze System blockieren; er wollte hören: Wartezeiten von TA (oder Prozessen in BS) können von anderen TA genutzt werden. Also: Insgesamt wird die Verarbeitung gegenüber dem sequentiellen Fall beschleunigt.

Welche Serialisierbarkeitsbegriffe gibt es?

FSR, VSR, CSR, OCSR, CO

Wie erreicht ein Scheduler Konfliktserialisierbarkeit?

kurze Einführung: sperrende vs. nichtsperrende Scheduler; 2PL-Protokoll

Was ist das Problem dabei?

Deadlocks; Vermeidung durch konservatives 2PL; vier Bedingungen für Deadlocks genannt

Welche Ansätze zur Behandlung von Deadlocks gibt es?

Gedächtnisprotokoll einer Diplomprüfung
Praktische Informatik

Prüfer:	Prof. Jarke
Beisitzer:	Hans Nissen
Fächer:	Grundlagen von Datenbanken Betriebssysteme (Silberschatz/Galvin, 4.Auflage, Kap. 1-9 + UNIX) Expertensysteme (G.Gottlob et al.)
Datum:	4.8.1995
Dauer:	13.55-14.40 (45 min)
Note:	1,3

Betriebssysteme (15 min)

Was passiert bei einem context-switch?

Register und PC in PCB „retten", PCB in entsprechende Warteschlange einhängen (kann CPU-, I/O- oder andere Schlange sein), den PCB des „neuen", nach bestimmten Scheduling-Verfahren ausgewählten Prozesses einlagern, dessen Register und PC aus PCB laden, Ausführung starten.

Was kann man tun, um mehrere Prozesse auf einem *Prozessor laufen zu lassen?*

Multiprogramming und Multitasking (=Timesharing) erklärt. Zwischenfrage: *Wozu dient Multiprogramming?* Zur Erhöhung des Durchsatzes!

Welche Scheduling-Verfahren kennen Sie?

Das war das Stichwort, auf das ich gewartet hatte. Habe also begonnen, Jarke einen Vortrag über die ganzen Verfahren zu geben und ihre Vor- und Nachteile beschrieben. Zwischenfrage: *Sie sagten, SJF sei optimal. Bzgl. welchem Kriterium?* Durchschnittliche Wartezeit. Vortrag fortgesetzt. Nach dem priority-scheduling hatte Jarke keine Lust mehr und unterbrach:

Welches Verfahren wird bei UNIX angewandt?

Mischform aus Round-Robin, Priority-Verfahren, Multilevel-Feedback mit aging.

Bei mehreren Prozessen kann es zu Problemen kommen. Wozu braucht man da Prozeßsynchronisation?

Zum kontrollierten Zugriff mehrerer Prozesse auf gleiche Daten.

Und wie kann das realisiert werden?

Semaphore erklärt. Bevor ich zu Konstrukten höherer Ebene (z.B. Monitore) kam:

Was für Probleme können im Zusammenhang mit Semaphoren auftreten?

Falsche Schachtelung der Semaphore möglich.

Was passiert denn dann?

Ähh... Deadlocks?! (stimmt nicht - Deadlocks haben mit der falschen Schachtelung nichts zu tun, daher die folgende Frage)

Wie kann man das in ein Relationenschema übertragen?

Mensch: MName, Stadt
Krankheit: KName, Sterbl.-rate
hat: MName, KName, Zeitpunkt

Schreiben sie mal eine SQL-Anfrage, die die Städte ausgibt und wie viele dort wohnende Menschen im Jahre 1995 an Cholera gestorben sind!

SELECT Stadt, Count(M.Mname)
FROM Mensch M, Krankheit K, hat H
WHERE M.MName = H.MName and K.KName = H.KName and Zeitpunkt = 1995
GROUP BY Stadt
Dabei erklärt, daß SELECT einer Projektion entspricht usw. Hatte zuerst einen kleinen Fehler drin: Statt Stadt hatte ich (oben und unten) Zeitpunkt geschrieben. War aber nicht weiter schlimm.

Expertensysteme (20 min)

Definition von EFRS?

PL1 mit Einschränkungen: Nur Hornklauseln, keine Disjunktion, Negation, Existenzquantifizierung, allquantifizierte Fakten.

Was gibt es denn dann noch für Formeln?

Fakten und Regeln kurz beschrieben

Wie ist die Semantik von EFRS erklärt?

Bin auf folgende Punkte eingegangen: modelltheoretische Semantik, Modellbegriff, Herbrand-Modell (allg. für PL1: Semantik der Funktionssymbole gleich deren Syntax), prozedurale Semantik, Fixpunktsemantik. Alles nur informal beschrieben. Keine Formeln.

Erklären sie den Unterschied zwischen Vorwärts- und Rückwärtsverkettung.

Gesagt, getan.

Und wie funktioniert die Auswertung bei PROLOG?

Unifikation und Resolution beschrieben.

Welche Probleme gibt es denn im Zusammenhang mit der Negation bei EFRS?

Closed World Assumption erklärt und kurzes Beispiel gegeben. Negation as finite failure beschrieben und auf die Problematik möglicher unendlicher Berechnungen eingegangen. Außerdem werden bei dieser Realisierung keine Variablenbindungen aufgebaut.

Jarke noch nicht zufrieden. Gut, jetzt haben Sie gezeigt, was bei Rückwärtsverkettung alles schiefgehen kann und wie das in PROLOG gelöst wird. Gibt es die Probleme bei Vorwärtsverkettung auch?

Erklärt, wie man Negation bei Vorwärtsverkettung realisiert: Vorwärtsverkette so lange bis entweder P hergeleitet wurde (dann ist not P = false) oder bis Fixpunkt erreicht und P nicht hergeleitet (dann ist not P = true).

Jarke noch immer nicht zufrieden.

Letzter Versuch: Da gibt es „stratifizierte Negation". Kann das aber auch nicht mehr genau erklären. Ist aber auch wirklich nicht mehr Stoff des Buches. (Genau das habe ich ihm auch gesagt)

(Dieser Bereich hat wesentlich länger gedauert als es hier scheint, da ich nicht so recht wußte, worauf Jarke eigentlich hinauswollte. Das gilt sowohl für die Frage nach der Semantik von EFRS — da wollte er wohl die Herbrand-Semantik — als auch für den Bereich „Probleme bei der Negation".)

Prevention, Avoidance, Detection + Recovery

Wieso keine Deadlocks bei konservativem 2PL?

Bedingung "hold-and-wait" nicht erfüllt

Andere Möglichkeiten zur Deadlockvermeidung?

Tree Locking Protokoll

Die Prüfungsatmosphäre ist sehr ruhig. Prof. Jarke erwies sich als geduldiger Zuhörer und hilft auch weiter, wenn man mal festhängt. Die Vorbereitung dieser Prüfung habe ich persönlich allerdings als besonders unangenehm empfunden, da man nur sehr wenige Anhaltspunkte hat, was man eigentlich lernen soll. Die Bücher zum Thema sind meist entweder schlecht zu lesen oder sehr umfangreich (in der Regel aber beides).

Material:

Folienskripten zu DB-Vorlesungen

Datenmodelle, Datenbanksprachen und DBMS-Systeme, Vossen, 2. Auflage, 1994

Datenbankhandbuch, Kap. zu Schichtenarchitektur, 1987

Fundamentals of DB Systems, Elmasri, Navathe, 1989, Kap. zu Anfrageoptimierung

Principles of Database and Knowledge-Base Systems, Ullman, 1989, Bd.1 Kap.3 zu Datalog, Bd. 2 Kap. 12+13 zu Magic Sets

Query Optimization in Database Systens, Jarke, Koch, ACM Computing Surveys, Vol. 16, No. 2, pp.111-143, 1984

Operating System Concepts, Silberschatz, Galvin, 4.Auflage, 1994

II. Compilerbau

Frage: *Wie kann man einen Parser realisieren ?*

Ich erkläre zunächst, daß ein Parser zur syntaktischen Analyse gehört, er als Eingabe vom Scanner Token erhält und als Ausgabe einen Parse-bzw. Ableitungsbaum ausgibt. Dann erwähne ich als grundsätzliche Möglichkeiten das Top-Down- und das Bottom-Up-Vorgehen. Für das Top-Down-Vorgehen beschreibe ich als erste Alternative die Verwendung eines Kellers und als zweite die Möglichkeit der Verwendung des rekursiven Abstiegscompilers, bei dem für jedes Nichtterminalsymbol eine Prozedur geschrieben wird.
Schließlich erwähne ich die LL(k)-Grammatiken.

Frage: *Wie kann ich einer LL(0)- bzw. einer LL(1)-Grammatik ansehen, daß sie eine solche ist ?*

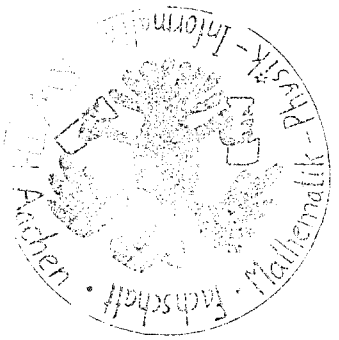
Ich erkläre, daß es bei einer LL(0)-Grammatik für jedesNichtterminalsymbol höchstens eine Regel geben darf und bei einer LL(1)-Grammatik die la-Mengen verwendet werden.

Frage: *Wie können Sie zur bottom-up-Analyse sagen ?*

Ich erkläre LR(0)- und LR(1)-Grammatiken und beschreibe, wie ein Kellerautomat mit LR(0)- bzw- LR(1)-Informationen arbeitet. In diesem Zusammenhang soll ich die Ermittlung der LR(0)-Informationen erläutern (NFA in einen DFA mittels Potenzmengenkonstruktion umwandeln).

Frage: *Welche Phase kommt denn nach der syntaktischen ?*

Ich erkläre die Aufgabe der semantischen Analyse, die einen Ableitungsbaum in einen attributierten Ableitungsbaum transformiert. Ich erkläre in diesem Zusammenhang, was eine attributierte Grammatik ist und was synthetische und inherite Attribute sind.



Prüfungsprotokoll Praktische Informatik

Professor Jarke

11.04.1995
16.00 Uhr

- **Datenbanken**
  (seine Vorlesung im SS91, Literatur: Korth, Silberschatz)
- **Betriebssysteme**
  (Silberschatz/Peterson, Kap. 1-9 + Kap 10)
- **Expertensysteme**
  (Gottlob/Frühwirth/Horn)

Dauer: 50 min
Note: 1.3

Expertensysteme:

- Jarke begann mit der Bemerkung, daß er bei meinem Kommilitonen vor 2 Stunden so wenig Zeit für XPS gehabt hat, daß er jetzt damit anfangen möchte. Also schnappt er sich das Buch, schlägt eine beliebige Seite auf und fragt: Was ist denn ein Meta-Interpreter? (Ich also erklärt, was dazu im Buch steht.)
- An so einem Meta-Interpreter kann man ja sehr schön erkennen, wie Prolog so funktioniert. Implementieren Sie doch mal einen Prolog-Metainterpreter in Prolog.
  (Tja, das war nun der Teil, der die Note runtergezogen hat. Ich hatte zwar zwei Regeln, die okay waren, aber die entscheidende Regel fehlte leider. Da mußte mir Jarke tierisch aufs Pferd helfen. Er hat's in Prosa vorgesagt und ich habe es in Prolog aufgeschrieben. Hiernach ging's aber nur noch aufwärts und ich habe mich bemüht, so schnell zu reden, daß Jarke nur noch Stichworte einwerfen konnte. Übrigens hat er sich für die besondere Regel, die man braucht, um Systemprädikate verwenden zu können, nicht mehr interessiert.)
- Wie würde man eine Regel zur Auswertung einer negierten Anfrage im Metainterpreter aufschreiben?
  (Entsprechende Regel aufgeschrieben. (Vgl. Buch, glaube ich).)
- Wie ist Negation in Prolog implementiert und welches Problem ergibt sich daraus?
  (Negation = negation as failure. Dann habe ich erläutert, daß keine Variablenbindung stattfindet und das gewisse Anfragen, dann eher merkwürdige Ergebnisse liefern können.)
- Das war aber nicht genau das, was er hören wollte. Er fragte dann, ob ich schon mal etwas von **** gehört habe. Ich: Nein. Er hat dann im Index des Buchs nachgeschaut und tatsächlich war's nicht drin (was auch immer es war). Welches andere Problem hat man denn bei Prolog?
  (Anfragen liefern ggf. kein Ergebnis, weil durch Tiefensuche in unendliche Tiefen vorgedrungen wird. (genau sowas wollte er wohl hören).)

  **** = Closed World Assumption
- Welches andere Verfahren gibt's denn?
  (Breitensuche erläutert. Irgendwie war auch noch zur Sprache gekommen, daß Prolog revidierendes Backward-Chaining macht.)
- Sprechen wir nun über Unsicherheit (und ich dachte schon jetzt wäre endlich Schluß mit XPS).

(Ich habe die vier verschiedenen Quellen von Unsicherheit dargelegt.)

- Welche Möglichkeiten gibt es denn, mit Unsicherheit fertig zu werden.
  (Ich habe Bayes erwähnt und habe dann auch die Formel hingeschrieben und dabei das ganze erklärt. (Er wollte, daß ich die Formel aufschreibe). Die Probleme von Bayes erläutert: Unabhängigkeit der Alternativen, Vollständigkeit (Summe W ergibt 1), etc.)
- Da gibt's noch eine andere Möglichkeit...
  (Certainty Factors. Prinzip erläutert, wie die Verknüpfung erfolgt etc. - vgl Buch)

Datenbanken:

- Irgendwie wollte er elegant zu Datenbanken überleiten und hat sich da noch mal auf Bayes berufen. Er meinte den hätte man ja wohl auch in Datenbanken mal gebraucht.
  (Ich habe was von Wert von Information erzählt. Und dann viel mir noch ein, daß er Bayes bei der Teamtheorie angewendet hat. Die hat er aber im SS91 gar nicht gemacht, sondern erst in folgenden Vorlesungen. Das habe ich ihm denn auch gesagt und er hat dann mit anderem weitergemacht.)
- Welche Möglichkeiten gibt's denn, wenn man eine Datenbank entwerfen will?
  (Entity-Relationship-Diagramm aufstellen oder direkt, etwa aus einem Formblatt, die Relationen aufstellen.)
- Machen wir doch das zweite, E-R-Diagramme habe ich schon den ganzen Tag gemacht.
  (Das ganze ging ohne konkretes Beispiel, d.h. er hat weder die Fluggesellschaft, noch den Supermarkt aus der Tasche gezaubert. Ich habe also erläutert, wie man - ganz nach Vorlesungsbeispiel - die Informationen in die 1. NF bringt, wie die Keys der Wiederholungsteile gebildet werden etc, dann die weitere Dekomposition zu 2. NF, 3. NF und BCNF erläutert. Dabei darauf hingewiesen welche Probleme jeweils beim Schritt in die nächste NF abgelegt werden.)
- Sie haben das jetzt für ein ganzes Unternehmen gemacht, welches Problem haben Sie?
  (Das ganze ist sehr unübersichtlich und wird optimiert, in dem man Relationen mit gleichem Schlüssel zusammenfaßt.)
- Welches Problem gibt es, bei diesem Vorgehen?
  (FDs können verloren gehen. Ich habe ein kleines Beispiel dafür gebracht, was passiert, d.h. wie man falsch zerlegt, wenn nicht alle FDs beachtet werden. 1.-3. NF erhalten FDs, lediglich bei der BCNF können FDs verloren gehen, auf das Teacher-Student-Subject-Beispiel aus der Vorlesung verwiesen.)

- Wie geht man denn sicher, daß man keine FDs übersieht?
  (Abschlußberechnung. habe den Algorithmus kurz erläutert (nicht aufgeschrieben).
- Was ist eine weitere Eigenschaft, die man für die Relationen haben möchte?
  (Lossless-Join! Das Problem hat man meistens dann, wenn man an Nicht-Schlüsseln zerlegt und kann vermieden werden durch das Syntheseverfahren.)
- Erläutern Sie das Verfahren.
  (Habe die vier Schritte des Verfahrens erläutert. Insbesondere das Herausfinden der CFDs hat ihn da interessiert. Die Punkte 1 und 3 (Minimalisierung & Verschieben) wollte er nicht genauer erklärt haben. Wobei ich bei Punkt 3 da auch ganz froh drüber war, das ist nämlich in den Vorlesungsunterlagen absolut mangelhaft dargestellt.)

II. Compilerbau

Frage: *Wie kann man einen Parser realisieren ?*

Ich erkläre zunächst, daß ein Parser zur syntaktischen Analyse gehört, er als Eingabe vom Scanner Token erhält und als Ausgabe einen Parse-bzw. Ableitungsbaum ausgibt. Dann erwähne ich als grundsätzliche Möglichkeiten das Top-Down- und das Bottom-Up-Vorgehen. Für das Top-Down-Vorgehen beschreibe ich als erste Alternative die Verwendung eines Kellers und als zweite die Möglichkeit der Verwendung des rekursiven Abstiegscompilers, bei dem für jedes Nichtterminalsymbol eine Prozedur geschrieben wird.
Schließlich erwähne ich die LL(k)-Grammatiken.

Frage: *Wie kann ich einer LL(0)- bzw. einer LL(1)-Grammatik ansehen, daß sie eine solche ist ?*

Ich erkläre, daß es bei einer LL(0)-Grammatik für jedesNichtterminalsymbol höchstens eine Regel geben darf und bei einer LL(1)-Grammatik die la-Mengen verwendet werden.

Frage: *Wie können Sie zur bottom-up-Analyse sagen ?*

Ich erkläre LR(0)- und LR(1)-Grammatiken und beschreibe, wie ein Kellerautomat mit LR(0)- bzw- LR(1)-Informationen arbeitet. In diesem Zusammenhang soll ich die Ermittlung der LR(0)-Informationen erläutern (NFA in einen DFA mittels Potenzmengenkonstruktion umwandeln).

Frage: *Welche Phase kommt denn nach der syntaktischen ?*

Ich erkläre die Aufgabe der semantischen Analyse, die einen Ableitungsbaum in einen attributierten Ableitungsbaum transformiert. Ich erkläre in diesem Zusammenhang, was eine attributierte Grammatik ist und was synthetische und inherite Attribute sind.

III. Betriebssyteme

Frage: *Was können Sie zur Hauptspeicherverwaltung sagen ?*

Ich erkläre ihm, daß es einmal möglich ist, den Speicher in nur zwei Teile einzuteilen, den Bereich für das Betriebssystem und den für einen User (single partition allocation), daß es aber bei mehreren Benutzern nötig ist, den Speicher auf mehrere Prozesse aufzuteilen (multiple partition allocation). Dazu zähle ich als Möglichkeiten Paging, Segmentation und Paged Segmentation auf.

Frage: *Wie funktioniert das denn beim Paging ?*

Ich erzähle ihm das Grundprinzip mit der Page Table usw. und daß die noch nicht belegten Seiten (Frames) vom Betriebssystem zu verwalten sind. Falls ein Prozeß in den Hauptspeicher will, dafür aber nicht mehr genug Platz ist, kann er entweder nicht aufgenommen werden oder ein anderer Prozeß muß herausgeswapt werden. Irgendwie kommen wir dann - von mir forciert - auf virtuellen Speicher zu sprechen.

Frage: *Was passiert bei einem Seitenfehler ?*

Nachdem mit Hilfe der Page Table festgestellt wurde, daß die angesprochenen Seite nicht im Hauptspeicher ist, wird sie aus dem Swap-Bereich der Platte in den Hauptspeicher geholt. Ist dort ein Frame frei, so wird dieser benutzt, ansonsten muß eine anderer Frame freigemacht werden.

Frage: *Was kann denn beim dauernden Paging passieren ?*

Hier ist das *Thrashing*-Phenomen angesagt. Ich erkläre ihm, was es ist .

Frage: *Wie kann man dem begegnen ?*

Ich bringe das Working-Set-Modell ins Spiel, womit er zufrieden ist.

Frage: *Sie sprachen eben von mehreren Prozessen. Wie wird das denn geregelt ?*

Zunächst erkläre ich, was ein Prozeß ist und in welchem Zustand er sein kann (ready, running, waiting) bzw. worauf ein Prozeß alles warten kann (s. Abb. 4.8 im Buch). Dann erwähne ich die ready-Schlange, in der Prozesse auf die Zuteilung des Prozessors warten.

Frage: *Welcher Prozeß wird denn nun gewählt ?*

Ich erwähne Scheduler und einige Möglichkeiten des Schedulings.

Frage: *Welche Kriterien gibt es denn, diese zu beurteilen ?*

Ich fange an aufzuzählen: Wartezeit in der ready-queue, komme aber nur bis hier, da er wohl nur das hören wollte.

Frage: *Wie geschieht denn genau ein Prozeßwechsel ?*

Jetzt wird es etwas chaotisch, da Prof. Jarke seinen Spaß am Detail zu entdecken scheint. Was er hören will und wir dann "gemeinsam entwickeln", ist folgendes:
Nachdem ein Hardwareinterrupt (z.B. Timer- oder I/O-Interrupt) erfolgte - solche muß es geben, damit ein Prozeß nicht die ganze Zeit die CPU belegt - wird von einem Hardwaremechanismus der aktuelle Inhalt des Befehlszählerregisters gerettet, dann der Befehlszähler auf die Startadresse einer Interruptroutine gesetzt, die dafür sorgt, daß der gerettete Befehlszähler sowie die Registerinhalte bzgl. des unterbrochenen Prozesses in dessen PCB gespeichert werden. Dabei darf die Interrupt-Routine natürlich nicht die Register vor der Rettung deren Inhalte überschreiben. Schließlich bestimmt der Scheduler, welcher Prozeß als nächster die CPU erhält. Die Register werden mit den im PCB gespeicherten Werten geladen und der Befehlszähler mit der Adresse geladen, an der der Prozeß unterbrochen wurde (auch im PCB enthalten).
Prof. Jarke fragt, ob neben den Register- und dem Befehlszählerinhalt nicht noch etwas zu retten ist. Mir fällt dazu nichts ein, und er sagt schließlich: der Page-Table-Inhalt.
Prof. Jarke setzt nun zum Finale an und sagt:

Frage: *Mit welcher Scheduling-Politik erreicht man denn garantiert die kürzesten Wartezeiten in der ready-Schlange ?*

Glücklicherweise fällt mir hier Abb. 4.12 ein und ich sage: Shortest Job First.

Allg. Bemerkungen zur Prüfung:

Die Prüfung war ziemlich fair, wenn auch Prof. Jarke, nachdem er gemerkt haben muß, daß ich B-Stern-Bäume nicht besonders gut beherrschte, eher damit hätte Schluß machen können.
Die Prüfungsatmosphere war ruhig, Prof. Jarke wurde nie hektisch oder ungeduldig.
Die Benotung empfand ich als ok.

Ja ja, aber bei Datenbanken sind die Anfragen ja auch endlich, weil sie durch das FROM bereichsgeschränkt sind.
(Tja, das stimmt irgendwie. Wußte wirklich nicht, worauf er hinaus wollte, habe das auch gesagt)

Die Rekursion ist möglich in EFRS! In Datenbanksprachen unmöglich.
(Aha. Wäre ich nie drauf gekommen.)

Sie haben vorhin gesagt, Cons (S) wäre endlich. Ändert sich daran etwas, wenn man auch die Negation und Disjunktion zuläßt?
Wenn man bei der Negation fordert, daß dieselbe Variable auch nichtnegiert vorkommt, ändert sich nichts. Wenn man das nicht fordert, kann es auch unendlich werden (keine Ahnung, ob das stimmt, steht nicht im Buch, und Jarke ließ auch nicht durchblicken, ob ich nun Quatsch erzählt hatte oder nicht). Wenn man die Disjunktion zuläßt, hat man nicht mehr nur Fakten vorliegen, sondern Disjunktionen von positiven und negativen Fakten.

Er hat dann noch irgendwas zur Negation gefragt und meinte dann irgendwann, ob ich schonmal was von Stratifizierung gehört hätte. Hatte ich nicht, steht auch nicht im Buch, was ich ihm auch gesagt habe, und damit hat er dann auch aufgehört.

Beschreiben Sie doch mal, wie die Semantik von EFRS beschrieben ist.
Habe allgemeine Interpretation erklärt, dann H-Interpretation als Teilmenge von H-Basis und H-Modell. Cons (S) als Menge aller aus S folgenden Fakten erwähnt.

Wie kann man denn Cons(S) bestimmen?
Entweder modelltheoretisch als Durchschnitt aller Modelle von S (Cons(S) ist minimales Modell von S) oder beweistheoretisch, indem man aus den vorliegenden Fakten und Regeln mittels EP alle möglichen neuen Fakten ableitet.

O. k., das war genug zur Theorie, nun zur Praxis. Nehmen wir doch mal Prolog. Da erfolgt ja die Auswertung rückwärtsverkettet. Erklären Sie das doch mal genauer.
Erst Rückwärtsverkettung erklärt, er wollte aber ein auf Prolog zugeschnittenes Beispiel haben. Den Metainterpreter wollte er allerdings nicht. Ich habe dann als Beispiel

mutter(X,Y) := kind(Y,X) & weiblich(X).

angegeben und daran erklärt, wie bei der Anfrage das Ziel mutter(beate,hans)? rückwärtsverkettet durch die Subgoals kind und weiblich zu lösen versucht wird, indem in der Datenbasis Fakten gesucht werden, die diese Subgoals erfüllen. Das war aber wohl nicht das, was er hören wollte; was er nun hören wollte, weiß ich nicht, denn er hörte hier einfach auf mit dem Thema.

Datenbanken

Jetzt müssen wir aber auch noch Datenbanken machen. Viele Formalismen in XPS beruhen ja auf Logik. In Datenbanken haben wir mit den FDs auch einen sehr logikähnlichen Konstrukt. Schreiben Sie doch mal die Definition für FDs formal hin.
Hier wußte ich wieder nicht so ganz, was er wollte, und habe erstmal die Definition, wie sie im Skript steht, hingeschrieben, von wegen, B hängt funktional von A ab (A -> B), wenn für jeden gleichen Wert von A auch der Wert von B gleich ist.

Nein, aber ich meine formal, so richtig mit Formeln.
Ich habe es dann mit einer Formel versucht, die ein bißchen nach Relationentupelkalkül aussah, war aber auch nicht gemeint. Diese gesuchte Formel sollte angeblich im Skript stehen, konnte ich aber bis jetzt nicht finden.

Na gut, nehmen wir doch für eine SQL-Anfrage mal unser Standardbeispiel Angestellter(Eno, Name, Adresse, Dept). Hier besteht ja eine funktionale Abhängigkeit zwischen Eno und Dept. Was bedeutet das konkret?
Daß kein Angestellter in zwei verschiedenen Departments arbeiten kann.

Gut. Dann stellen sie doch mal eine SQL-Anfrage nach dieser funktionalen Abhängigkeit, d. h. fragen sie nach den Angestellten, die in zwei verschiedenen Abteilungen arbeiten.
Hier habe ich dann etwas länger gebraucht, Jarke hat mir auch irgendwie nicht mehr viel Zeit zum Überlegen gelassen. Meine Lösung sah nachher ungefähr so aus:

```
SELECT Eno FROM Angestellter x, Angestellter y
WHERE x.Eno = y.Eno
    AND x.Dept != y.Dept
```

Hieraus eine Anfrage in Relationenalgebra zu machen, wäre wahrscheinlich etwas zu kompliziert. Aber sagen Sie doch mal, welchen Konstrukten der Algebra die einzelnen Anfrageteile entsprechen!
SELECT entspricht der Projektion, FROM dem kartesischen Produkt, obwohl daraus nachher sowieso meistens ein Join wird, und WHERE der Selektion.

Tja, dann brauchen wir noch eine schöne Frage zum Schluß. Man will ja bei der Erstellung von Datenbanken möglichst alle Relationen in 3. NF haben. Geben Sie einen abhängigkeitserhaltenden Algorihmus zur Dekomposition in die 3. NF an.
Ich habe dann den Algorithmus angegeben, wie er in der Übung WS 95/96 beschrieben war, also erstmal minimale Überdeckung bestimmen, dann gleiche linke Seiten zusammenfassen.

Der Algorithmus, den Sie beschreiben, ist doch eher ein Synthesealgorithmus!
Habe dann gesagt, daß es genau so in der Übung beschrieben ist, und zwar geht man dabei von einer Universalrelation aus, die man durch diesen Algorithmus dekomponiert.

Viel Glück für Eure Prüfung!

Hier überlegte ich wieder, weil ich nicht sicher war, ob man die Negation in SQL so einfach hinschreiben kann. Ich dachte also etwas nach ...

"Wie sieht denn so eine SQL-Anfrage struktuell aus ?"

Ja doch, das weiß ich : SELECT ... FROM ... WHERE

"Und wie lautet deshalb die gesuchte Anfrage ?. Schreiben sie doch einfach mal was hin"

Ich schrieb also hin :

```
SELECT Supplier
FROM SIP
WHERE ... (tja ich vermutete so etwas wie...) Supplier NOT IN SADR
```

"Na, so können sie das aber nicht hinschreiben ."

Klar, also :

```
SELECT Supplier
FROM SIP
WHERE Supplier NOT IN (  SELECT Supplier
                         FROM SADR)
```

Ob das jetzt so richtig war (ist) weiß ich nicht. Jedenfalls war Jarke damit zufrieden.

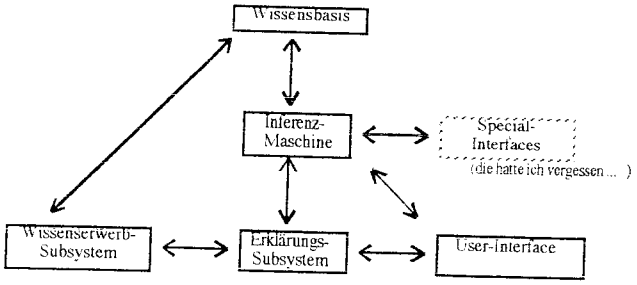
"Ok, dann kommen wir jetzt noch zu Expertensysteme"

Expertensysteme

Hinter diesem Bereich stand für mich ein großes Fragezeichen, da ich die Vorlesung von Frau Dr. Motschnig prüfen ließ. Bisher hatte Jarke wohl in diesem Bereich Bezug auf das Frühwirth-Buch genommen. Deshalb hatte ich da sicherheitshalber auch mal mehrere Blicke reingetan ... man weiß ja nie.

"Wie sieht denn die grundlegende Architektur eines Expertensystems so aus ?"

Schluck, ich hatte mit einen Einstieg zum Thema 'Intelligent Agents' gerechnet. Also mußte ich ein wenig nachdenken. Auf meine Bemerkung : 'Ich muß gerade mal nachdenken, in welchem Film ich bin', bemerkte Jarke sichtlich erheitert : "Nun, Sie sind hier in einer Prüfung in praktischer Informatik" (Gelächter !)
Schließlich konnte ich mich aber noch an das folgende Bildchen (die 'Architektur' ;-)) erinnern und malte daher auf :



Ich begann, die einzelnen Komponten zu erläutern ...

"Wie ist das Wissen in der Wissensbasis abgelegt ?"

Zum einen durch Fakten, die wahre Sachverhalte der modellierten Welt beschreiben, zum anderen durch Regeln, die beschreiben, wie man neue Fakten ableiten kann.
"In der Vorlesung wurde ein Beispiel für ein erfolgreiches Expertensystem vorgestellt : MYCIN. Wie sind denn die Regeln der Wissensbasis von MYCIN aufgebaut ?"

Daran konnte ich mich auch noch erinnern : Regeln sind in MYCIN aufgebaut in der Form
IF 'Bedingung' THEN 'Action' wobei 'Bedingung' eine boolsche Funktion von Funktions-Triplet Paaren ist (so wurde es doch genannt ?) : <fct> <obj> <attr> <value>