# *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

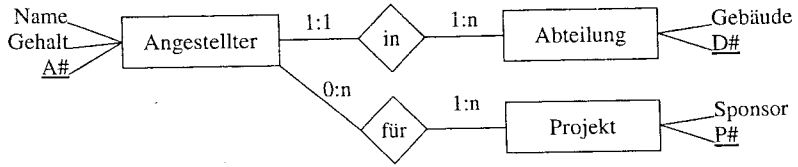
## *Die Prüfung*

### Requirements Engineering

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

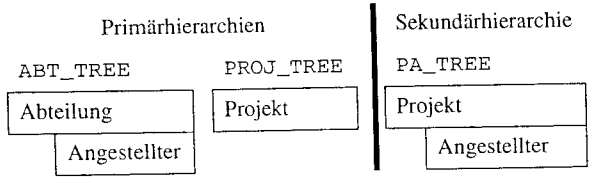
- Angestellte mit Nummer, Name und Gehalt
- Abteilungen mit Nummer (D#), Titel und Gebäude
- Ein Angestellter ist genau einer Abteilung zugeordnet, in einer Abteilung arbeiten mehrere Angestelle
- Projekte mit Nummern und Geldgebern
- Ein Angestellter kann in mehreren Projekten arbeiten, ein Projekt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

*Ich:* Hingemalt, Schlüssel festgelegt.

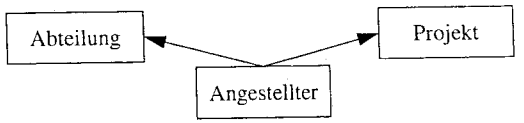



### Einführung in Datenbanken

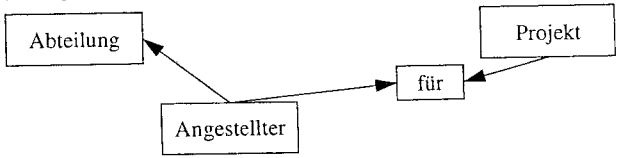
*Jarke:* Übertragen sie das mal ins Hierarchische, Netzwerk- und Datenmodell.
*Ich:* *(Schluck)* Im Hierarchischen Modell müßten die Angestellten ja wohl entweder redundant abgelegt werden, oder eine Relation wird mit einer Sekundärhierarchie dargestellt. Also:




*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: $\sigma$, $\pi$, $\otimes$ *(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, $\sigma$ und $\pi$ 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 $\sigma_{true}$ 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 $\pi$ 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 $\pi$ sogar noch teurer als $\sigma$. 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: $\oplus$)*

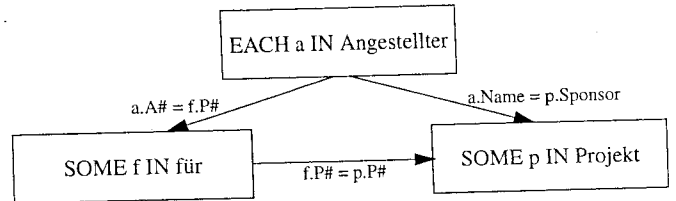
*Ich:* Definition $R \oplus S = \pi_R(R \otimes S)$, Viieel günstiger: Nur Existenz zu testen, keine Vergrößerung der Antwortmenge. *(Schwärm…)*

*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?

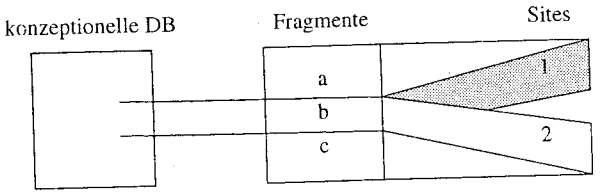
*Ich:* Man muß nicht komplette Relationen, sondern nur die Joinspalte verschicken… *(Grübel:)* Immer die aus der Zweiten Tabelle, oder auch mal anders?

*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)

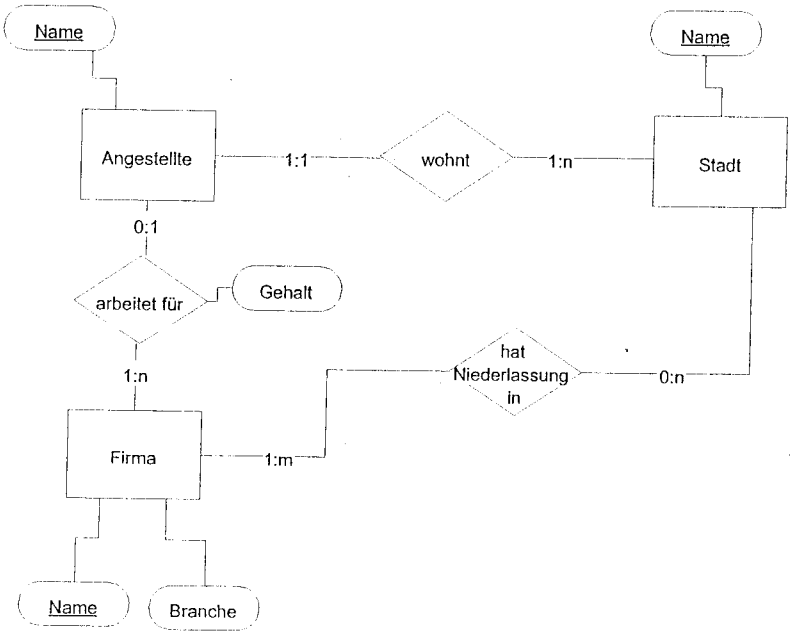



Abbildung 1: ER-Diagramm

## 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?

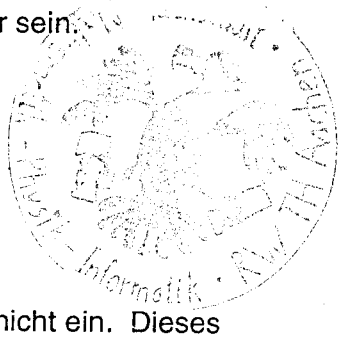
ich: Exists — oder auch count(), avg() und so.

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.

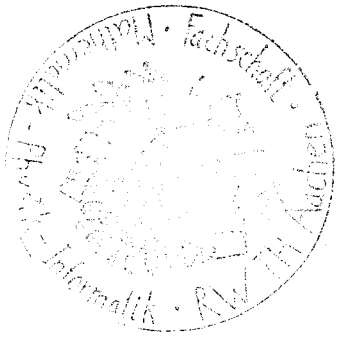
**Einführung in Datenbanken:**

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.