3 Implementierung von Datenbanken

- **Na gut, dann fragen wir noch mal schnell was zur Implementierung. (Es waren schon 35 Minuten rum).**
  Erzählen sie mir doch mal was zur Joinreigenfolgenoptimierung.

  Ich erzähle ihm also die Story von den möglichen Permutationen (wann will Joins nicht zum kartesischem Produkt entarten lassen) und von Kostenmodellen für Zwischenergebnisgrößen. Optimale Reihenfolge der Zwischenergebnisgrößen. Parametersysteme. Gültige Parametersysteme erklärt.

- **OK OK. Was gibt es da denn noch ?**

  Besonders gut ist wenn man Semijoins erhält. Viel günstigere Auswertung etc.

- **Wann kann man eine Anfrage durch Semijoins auswerten ?**

  Die Geschichte der guten und der bösen Ausdrucke. Hinreichend für böse: Zykeln im Quantgraph. Notwendig: Nicht als Genereller Semijoinausdruck darstellbar. Noch ein paar Schachteln aufgemalt, einstufige Prädikate daran erklärt.

- **Ja. Da gibt es ja noch so was wie Transaktionsverwaltung.**

  Ich erzähle von Serialisierbarkeit. Definition hiervon. Die Klassen FSR $\subset$ VSR $\subset$ CSR. Herbrand–Semantik. LRF, RF Mengen.

- **Mit welchem Aufwand ist FSR also zu testen ?**

  Exponentiell. Es sind n! Permutationen (im Worst-case) bei n zu betrachtenden Transaktionen zu behandeln.

- **Wie sieht das noch mit Fehlertoleranz aus ?**

  RC $\subset$ ACA $\subset$ ST $\subset$ RG erklärt und definiert. Da RC orthogonal zu FSR ist, muss man beides fordern.

- **Wie wird das in der Realität gemacht ?**

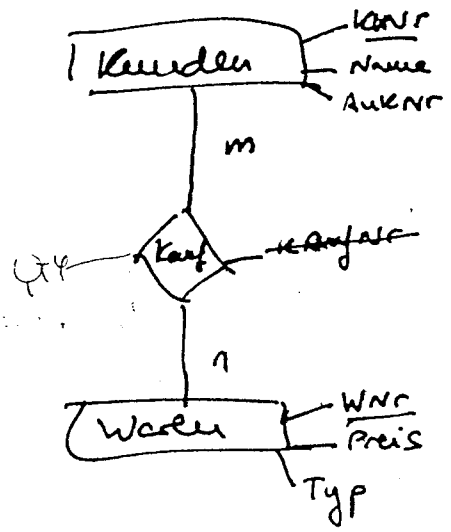
  Z.B. 2PL Scheduler. Besser S2PL oder SS2PL oder konservatives 2PL (geht meistens nicht). Und noch ein bischen dazu rumgelabert.

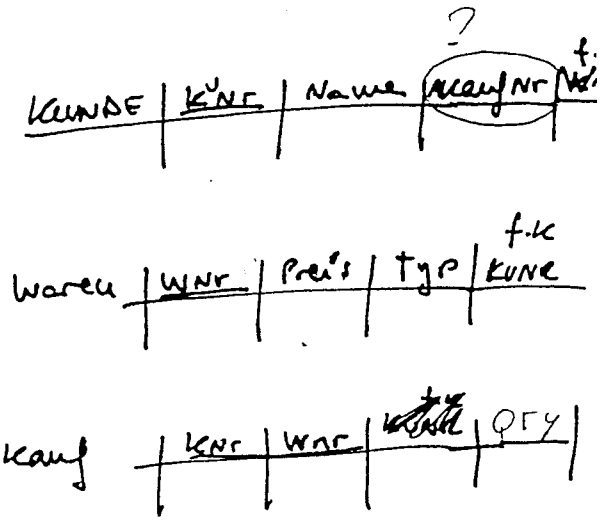
- **Gut. Wann haben wir angefangen ? Beisitzer: Vor genau 45min.**
  OK das war's.

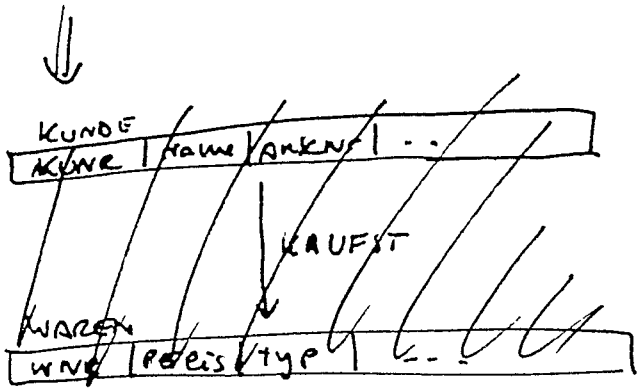
4 Fazit

Aus Protokollen zu Prüfungen über Expertensysteme war ja schon zu entnehmen, daß Jarke DATALOG liebt. Also ganz klar selbst Schuld, die Magic–Set Transformation nicht draufzuhaben. Das war auch die Begrüngung (+ der eine Hänger EER$\longrightarrow$RelDB) warum es nicht zur 1.0 reichte. Schade, in anderen Vertiefungsprüfungen fragt er halt weniger zum dritten Fach. Ich kann auf jeden Fall bestätigen, was auch schon in anderen Protokollen stand: Jarke ist ein sehr lockerer, angenehmer Prüfer. Kommt man nicht weiter, gibt er Tips. Ich hatte mich zweimal verschrieben (z.B. hatte ich zweimal Group by, statt einmal Group by und Order by in der SQL–Anfrage geschrieben). Jarke sagt beide Male „Äääh, ich kann nicht so genau lesen was da links unten steht". Also ein dezenter Hinweis, daß etwas nicht stimmt.

SUPERMARKT







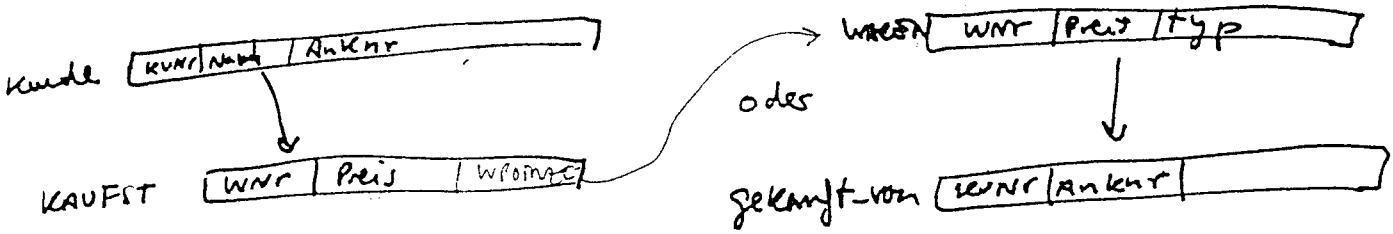
For each Entity ein Record

For each Relationship ein Set

1:1 1:N

Attribute der Relation in menge

- HIERARCHISCHE



oder VIRTUAL POINTER

Immer wenn ein Kunde was kauft, Replikation.

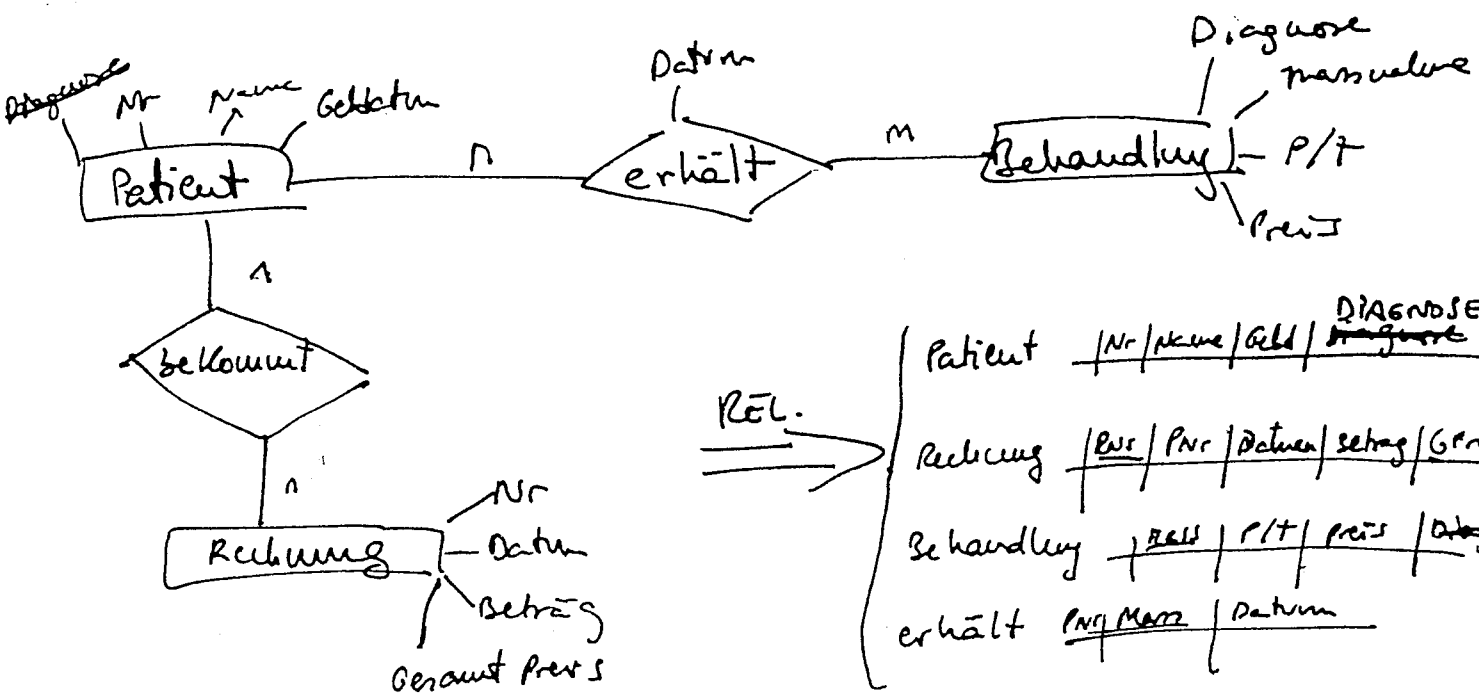
- WELCHE WAREN EIN KUNDE GEKAUFT HAT.

SELECT WNR, KNR FROM KAUF

- ALLE WAREN UND genaue verkaufte Menge

SELECT SUM(QTY) FROM KAUF
GROUP BY WNR

SELECT COUNT(*) FROM KAUF
GROUP BY WNR



REL.

Patient | Nr | Name | Geb | ~~Diagnose~~ (DIAGNOSE)

Rechnung | RNr | PNr | Datum | Betrag | GPreis

Behandlung | ~~RNr~~ | P/t | Preis | ~~Diagnose~~

erhält | PNr | MNr | Datum

FD

Rechnr RechDatum Diagnose PtNr Pname Gebdatum Bdatum Manualnr P/t Preis GPrei

~~PtNr~~ Pname Gebdatum Rechnr | Rechnr RechDat Diag PtNr Bdatum Manualnr P/t Preis GPrei

~~Ptnr Name Geb~~

Ptnr Bdatum Rechnung Rechnr RDatum Diag Ptnr Manual GPreis

Manualnr Bdatum P/t Preis

SQL

```
SELECT  SUM(B.PREIS)
FROM    Behandlung, RECHNUNG, MASSNAHME
WHERE   R.DATUM='1990' AND RNR = P.RNR AND P.PNR = M.MASS.
                                AND M.MASS = B.MASS
GROUP BY DIAGNOSE.
```

3NF

~~Patient~~

ReclNr
Diagnose
Pnr
Patname
Gebdatum
Gesamtkosten

→ 1 NF →

ReclNr
Diagnose
Gesamtkosten

ReclNr
Pnr
Patname
Gebdatum

→ 2NF →

ReclNr
Patnr

Pnr
Patname
Gebdatum

Recl → Diag, Ges.

→ 2 NF →

Patient PNr
Pname
Gebdatum

Rechng ReclNr
Diagnose
Pnr
Gesamtkosten

SELECT P.NAME PNR ~~R.Diagnose~~ R.Diagnose
FROM P, R
WHERE P.GEB = '1930' AND P.~~NAME~~ PNR = R.PNR

RA : $\Pi_{P.NAME,\ P.PNR,\ R.DIAG}\left(\sigma_{GEB='1930'}(PATIENT) \bowtie_{P.NR = R.PNR} RECHNUNG\right)$

$\{P.NAME, P.PNR, R.DIAG \mid \exists p \in Patient(p) \wedge \exists r \in Rechnung(r) \wedge p.geb = '1930' \wedge p.pnr = r.pnr\}$

$\{vut \mid patient(v, u, '1930') \wedge rechnung(x, t, u, q)\}$

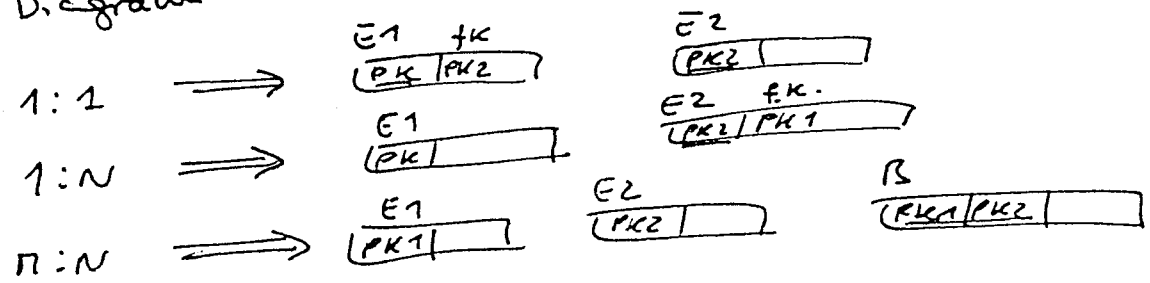
- Constraints on ER DIAGRAM

  Cardinality ratio: 1:N, 1:1, N:M.

  total participation: $\forall E$ belongs to a Beziehung

  partial " $\nexists E$

- ER Diagram $\Longrightarrow$ relationalen Modell



1:N ALS RELATION

Wenn viele $\in R$ auf N seite, die keine Relation zu 1 seite haben gibt $\rightarrow$ NULL-Werte

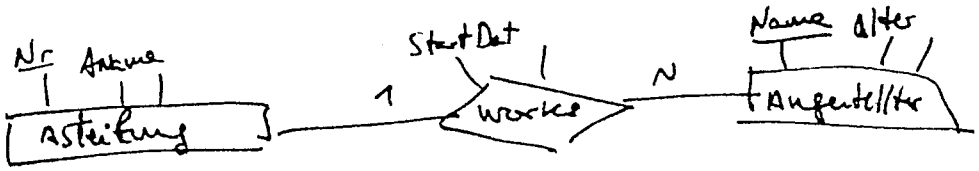
Bücher

BNr	Name	PNr	
123	—	0	→ von Keiner gelesen.
456	—	0	
		0	

```
SELECT AVG (ALTER)
  FROM GEFANGEN
  ~~WHERE~~
  GROUP BY G.NR
```

SPRACHEN RELATIONALE MODELL

- THEORETISCH
  - rel. Algebra
  - rel. Kalkül
- PRAKTISCH
  - SQL – entw. von IBM
  - QBE
  - QUEL – für INGRES



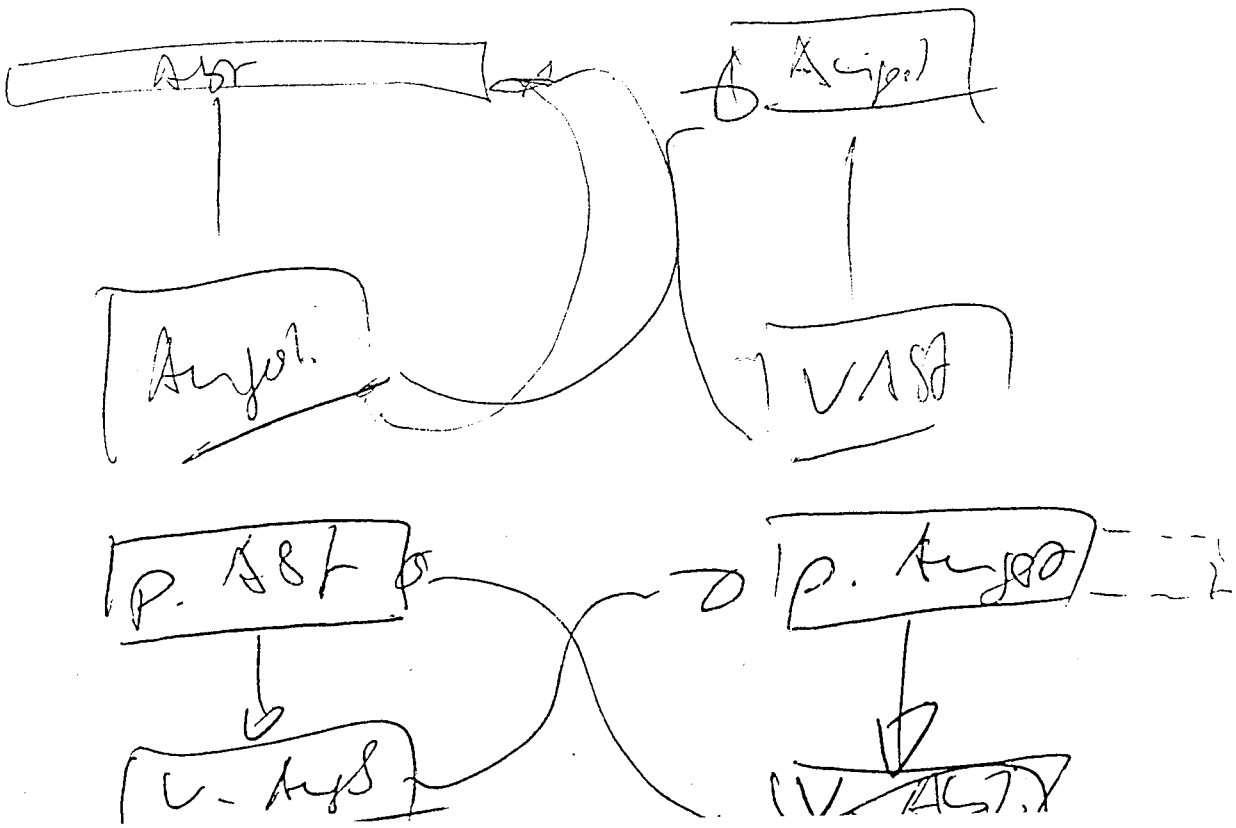
Abteilung | Nr | AName

Angestellter | Name | ANr | StartDat / Alter

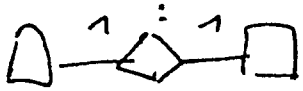
f.k

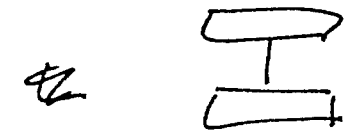
```
SELECT ANG.NAME
  FROM ANGESTELLTE, ABTEILUNG
 WHERE A.ALTER > 56 AND AB.NAME = 'F & E'
       AND A.ANR = AB.NR
```

Hier

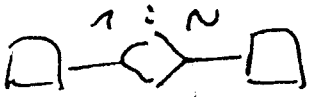


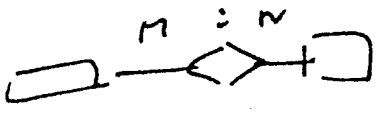
NETZWERK MODEL : Composite Attribute erlaubt!



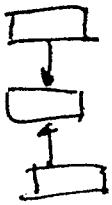


1 member Kt





linking record



Prüfungsprotokoll Praktische Informatik
Prof. Jarke

Datum: 7.12.95
Themen: Einf. in Datenbanken
Implementierung von Datenbanken
Betriebssysteme (Silberschatz/Galvin: Operating System Concepts)

Begonnen hat Jarke mit **Betriebsystemen**: „Wollen wir doch mal einige Zusammenhänge zwischen Betriebssystemen und Datenbanken diskutieren. Der gleichzeitige Zugriff von Transaktionen auf Datenobjekte muß ja synchronisiert werden, welche Methoden gibts es denn dazu?"

Als Softwarelösung gibts es zum Beispiel den Bakery-Algorithmus...

Wie sieht der denn aus? Schreiben Sie mal hin!

Bakery-Algorithmus wie im Buch hingeschrieben.

OK. Welche Konzepte kennen Sie noch ?

Semaphoren. Definition von Wait und Signal hingeschrieben und die Realisierung mit einer FIFO-Warteschlange (um Busy-Waiting zu vermeiden!) erkärt.

Da gibt es ja das Consumer-Producer Problem. Erklären Sie das mal und schreiben Sie mal eine Lösung dafür mit Semaphoren hin.

Ich hab folgendes hingeschrieben:

Producer

```
Wait(S)
IF counter < n
        THEN BEGIN
                add_element_to_buffer
                counter := counter +1
                END
Signal(S)
Remainder Section (Produzieren)
```

Consumer

```
Wait(S)
IF counter > 0
        THEN BEGIN
                get_element_from_buffer
                counter := counter -1
        END
Signal(S)
Remainder Section (Konsumieren)
```

und erklärt, daß der counter die Füllung des begrenzten (!!) Puffers angibt. Jarke war damit nicht so ganz einverstanden und sagte, daß es bei dieser Lösung noch ein Problem gäbe...

Wahrscheinlich ist entweder Starvation oder Deadlock nicht verhindert...

Was passiert denn, wenn der Producer ein neues Element produziert hat und dann merkt, daß der Buffer voll ist? Besser wäre doch, wenn er nur *dann* ein neues Element produziert, wenn auch Platz im Puffer ist...

Ich gebe Ihm recht aber komme nicht ganz auf die richtige Idee, meine Lösung dahingehend zu verändern. Jarke meint daraufhin, ich sollte einen zweiten Semaphor einführen. Ich komme aber immer noch nicht auf die rechte Lösung. Schließlich sagt der Beisitzer, daß es auch mit nur einem Semaphor geht! Ich bin inzwischen völlig durcheinander und überlasse die Diskussion Hernn Jarke und seinem Assi...
Das Problem sollte man sich also vor der Prüfung mal klarmachen!

Schließlich: Was das Sperren von Datenobjekten angeht, da gibt es ja das Multigranulare Sperrprotokoll. Malen Sie mal diesen Baum auf und erklären Sie das Ganze...

Gesagt getan. (Locks werden Top-Down gesetzt und BottomUp aufgehoben)

Schreiben Sie mal die Kompatabilitätstabelle für die Intension-Locks auf!

Aufgeschrieben und erläutert.

Was bedeutet *riwl* und wofür braucht man das?

riwl = Read-Intension-Write-Lock, diesen Lock setzt man z.B. auf ein File, wenn man das ganze File lesen und vielleicht einige wenige Tupel im File ändern möchte.

OK. Um nochmal einen Bezug zu Betriebssystemen herzustellen betrachten wir mal die 5 Schichten-Architektur. Wieso kann es sinnvoll sein, die Systempufferverwaltung vom DBMS statt vom Betriebssystem durchführen zu lassen?

1. Sicherheitsgründe bzw. Unterstützung des Recovery. Wenn man z.B. nur REDO-Recovery vorsieht, dürfen vor dem COMMIT einer Transaktion keine DB-Seiten in die stabile DB geschrieben werden. Ferner müssen vor dem Auslagern von DB-Seiten immer zuerts die dazugehörigen Log-Einträge auf Platte geschrieben werden. Das erkennt das Betriebsystem nicht.
2.Performance: DBMS kann andere Seitenaustauschstratgie bevorzugen (z.B. LRU-K statt LRU) und hat auch mehr Möglichkeiten, Prefetching zu betreiben.

Im Mitnehmen auch Zusammenfassung BS

Prüfungsprotokoll : DB + BS

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

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

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

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

- Seitenersetzungsstrategie
  'Pre Petching', ...

Prüfungsprotokoll Praktische Informatik
Prof. Jarke

Datum: 7.12.95
Themen: Einf. in Datenbanken
Implementierung von Datenbanken
Betriebssysteme (Silberschatz/Galvin: Operating System Concepts)

Begonnen hat Jarke mit **Betriebsystemen**: „Wollen wir doch mal einige Zusammenhänge zwischen Betriebssystemen und Datenbanken diskutieren. Der gleichzeitige Zugriff von Transaktionen auf Datenobjekte muß ja synchronisiert werden, welche Methoden gibts es denn dazu?"

Als Softwarelösung gibts es zum Beispiel den Bakery-Algorithmus...

Wie sieht der denn aus? Schreiben Sie mal hin!

Bakery-Algorithmus wie im Buch hingeschrieben.

OK. Welche Konzepte kennen Sie noch ?

Semaphoren. Definition von Wait und Signal hingeschrieben und die Realisierung mit einer FIFO-Warteschlange (um Busy-Waiting zu vermeiden!) erkärt.

Da gibt es ja das Consumer-Producer Problem. Erklären Sie das mal und schreiben Sie mal eine Lösung dafür mit Semaphoren hin.

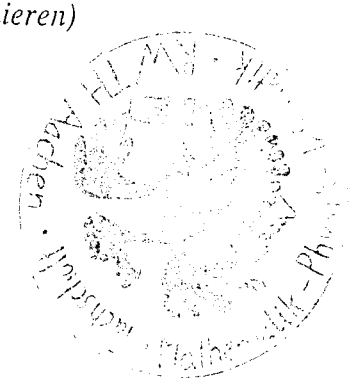
Ich hab folgendes hingeschrieben:

Producer

```
Wait(S)
IF counter < n
        THEN BEGIN
                add_element_to_buffer
                counter := counter +1
                END
Signal(S)
Remainder Section (Produzieren)
```

Consumer

```
Wait(S)
IF counter > 0
        THEN BEGIN
                get_element_from_buffer
                counter := counter -1
        END
Signal(S)
Remainder Section (Konsumieren)
```



und erklärt, daß der counter die Füllung des begrenzten (!!) Puffers angibt. Jarke war damit nicht so ganz einverstanden und sagte, daß es bei dieser Lösung noch ein Problem gäbe...

Wahrscheinlich ist entweder Starvation oder Deadlock nicht verhindert...

Was passiert denn, wenn der Producer ein neues Element produziert hat und dann merkt, daß der Buffer voll ist? Besser wäre doch, wenn er nur *dann* ein neues Element produziert, wenn auch Platz im Puffer ist...

Ich gebe Ihm recht aber komme nicht ganz auf die richtige Idee, meine Lösung dahingehend zu verändern. Jarke meint daraufhin, ich sollte einen zweiten Semaphor einführen. Ich komme aber immer noch nicht auf die rechte Lösung. Schließlich sagt der Beisitzer, daß es auch mit nur einem Semaphor geht! Ich bin inzwischen völlig durcheinander und überlasse die Diskussion Hernn Jarke und seinem Assi...
Das Problem sollte man sich also vor der Prüfung mal klarmachen!

Schließlich: Was das Sperren von Datenobjekten angeht, da gibt es ja das Multigranulare Sperrprotokoll. Malen Sie mal diesen Baum auf und erklären Sie das Ganze...

Gesagt getan. (Locks werden Top-Down gesetzt und BottomUp aufgehoben)

Schreiben Sie mal die Kompatabilitätstabelle für die Intension-Locks auf!

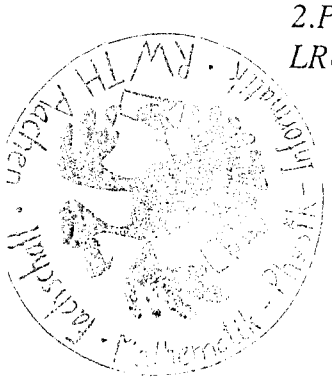
Aufgeschrieben und erläutert.

Was bedeutet *riwl* und wofür braucht man das?

riwl = Read-Intension-Write-Lock, diesen Lock setzt man z.B. auf ein File, wenn man das ganze File lesen und vielleicht einige wenige Tupel im File ändern möchte.

OK. Um nochmal einen Bezug zu Betriebssystemen herzustellen betrachten wir mal die 5 Schichten-Architektur. Wieso kann es sinnvoll sein, die Systempufferverwaltung vom DBMS statt vom Betriebssystem durchführen zu lassen?

1. Sicherheitsgründe bzw. Unterstützung des Recovery. Wenn man z.B. nur REDO-Recovery vorsieht, dürfen vor dem COMMIT einer Transaktion keine DB-Seiten in die stabile DB geschrieben werden. Ferner müssen vor dem Auslagern von DB-Seiten immer zuerts die dazugehörigen Log-Einträge auf Platte geschrieben werden. Das erkennt das Betriebsystem nicht.
2.Performance: DBMS kann andere Seitenaustauschstratgie bevorzugen (z.B. LRU-K statt LRU) und hat auch mehr Möglichkeiten, Prefetching zu betreiben.



Erklären Sie bitte, was Monitore sind, am besten an einem konkreten Beispiel.
Also erklärt, daß es ein Konstrukt ist, mit dem Prozesse synchronisiert werden, die auf gemeinsame Datenstrukturen zugreifen wollen. Im Monitor darf sich immer nur ein Prozess zur gleichen Zeit aufhalten, womit Mutual exclusion garantiert ist. Ein Beispiel wäre das Producer-Consumer-Problem, wobei ein Monitor dazu genutzt werden könnte, den Zugriff auf den gemeinsamen Buffer zu regeln. (Er wollte wohl noch, daß ich dieses Beispiel konkret als Monitor hinschreibe, aber irgendwie habe ich es geschafft, mich darum zu drücken...)

Dies ist ja nun ein Synchronisationsmechanismus für zwei Prozesse. Kennen Sie noch einen für mehrere Prozesse?
Den Bakery-Algorithmus. Sollte ihn dann Punkt für Punkt aufschreiben und erklären.

Können dort Deadlocks auftreten?
Nein, denn es geht immer nur um den kritischen Bereich, also um ein Betriebsmittel.

Wenn man so schöne Softwarekonstrukte für die Prozessynchronisation hat, warum braucht man dann überhaupt noch Hardwarekonstrukte wie die Semaphoren?
Um busy waiting zu vermeiden. Bei den Semaphoren gibt es die Möglichkeit, wartende Prozesse in eine Warteschlange einzureihen.

Bei mehreren Prozessen können ja nun Deadlocks auftreten. Nennen Sie doch mal die vier Bedingungen für Deadlocks.
Mutual exclusion, no preemption, hold and wait, circular wait.

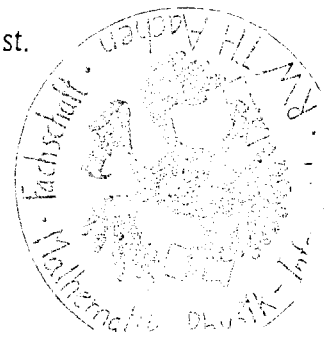
Expertensysteme

Das sollte reichen zu Betriebssysteme.
Wenn Sie nun einmal Datenbanksprachen wie SQL oder Relationenkalkül und EFRS vergleichen, was können Sie zu deren Mächtigkeit sagen?
SQL und Relationenkalkül sind insofern mächtiger, als daß sie auch Negation und Disjunktion ausdrücken können. Auf der anderen Seite ist EFRS angenehmer, weil entscheidbar ist, ob eine Formel allgemeingültig ist. Beim Relationenkalkül, das ja auf der Prädikationlogik erster Srufe basiert, ist dies nicht der Fall.

(sieht mich etwas verdutzt an) Was soll denn das bedeuten (kannte er wohl wirklich nicht)?
Das bedeutet, daß für EFRS ein Algorithmus existiert, der entscheiden kann, ob eine vorliegende Regel allgemeingültig ist.

Achso. Na gut, aber was können Sie denn mit EFRS erreichen, was bei den relationalen Sprachen unmöglich ist?
Man kann neue Fakten ableiten aus den bestehenden Regeln und Fakten. Bei Datenbanken kann man nur bereits existierende Fakten abfragen.

Naja, aber bei Datenbanken erhält man ja auch neue Informationen, wenn man z. B. einen Join macht. Da gibt es aber noch etwas anderes...
(Hm, was will er denn nur hören? Ein letzter Versuch:)
Man kann bei EFRS garantieren, daß alle Anfragen endlich sind, da Cons(S) endlich ist.



Gedächtnisprotokoll Diplomprüfung in Praktischer Informatik

bei:	Prof. Jarke
am:	21. April 1993
Dauer:	exakt 3/4 Std.
Fächer:	Einführung in Datenbanken Implementierung von Datenbanken (Vorlesung Kemper) Betriebssysteme (Buch Silberschatz ...)

Fragen:

Erklärung des Entity-Relationship-Modells.
Beispiel aufmalen für:

Datei Angestellte mit Name und Adresse,
Datei Abteilung mit Nummer und Titel
und Beziehung gehört-zu mit Einstellungsdatum
Jeder Angestellte gehört eindeutig zu einer Abteilung

Umsetzung dieses Beispiels in das relationale Datenbankmodell.

Wie kann man dies noch vereinfachen?

Relationen mit gleichen Schlüsseln zusammenfassen.

Jetzt betrachten wir das Modell mal als n:m - Beziehung.
Welche Probleme treten auf?

Anomalien, Redundanzen

Welche Normalform haben wir hier vorliegen?

Warum? Welche Abhängigkeit stört?
Was kann man nun dagegen machen?

Was ist Dekomposition?

Was soll denn bei der Normalisierung erhalten bleiben?

Erklärung von *verlustlos* und *abhängigkeitserhaltend* .

Wie kann man dies beweisen?

Tableaumethode für Verlustlosigkeit (mit Erklärung)
Abschlußberechnung für Abhängigkeitserhaltung

Wie berechnet man den Abschluß?

mit den Armstrong Axiomen

Synthese-Algorithmus kurz erklären.

Folgende Anfrage in SQL hinschreiben:

Alle Angestellten, die in Aachen wohnen und in der Forschung arbeiten.

Welche Arten von Anfragesprachen gibt es?

Welche SQL-Statements entsprechen welchen algebraischen Operatoren?

Die Anfrage in relationaler Algebra hinschreiben.

Wie kann man diese Anfrage optimieren?

Überleitung zur Implementierung:

Wie wird der Join implementiert?

Welche Probleme treten bei parallel laufenden Transaktionen auf?

Was ist *serialisierbar* ?

gleiches Ergebnis wie bei serieller Abfolge.

Womit erreicht man das?

Sperren setzten und 2-Phasen-Sperrprotokoll.

Wieso werden Probleme dadurch gelöst?

Was ist wenn nach der Freigabe von Sperren wieder Sperren angefordert werden können?

Welches sind die vier Bedingungen für einen Deadlock? Erläuterung.

Was kann man beim 2-Phasen-Sperrprotokoll tun, um Deadlocks zu vermeiden?

Bei welchen Speicherungsstrukturen wird das gemacht?

B-Bäume

Wie werden Sperren gesetzt, damit nicht der gesamte Baum blockiert wird?

Noch was zu Betriebssysteme:

Was ist Virtual Memory?

Wie wird es realisiert?

Demand Paging

Welche Seitenersetzungsstrategien gibt es? Erklärung.

Kennen Sie ein Betriebssystem, wo ein bestimmtes (welches?) angewandt wird?

In welche Schicht des 5-Schichtenmodells gehört das?

Systempufferschnittstelle.

Gedächtnisprotokoll
Diplomprüfung Vertiefungsgebiet Informationssysteme
Professor Jarke
Dezember 1996
Dauer: 60 Minuten

Prüfungsvorlesungen:
Einführung DB (Jarke)
Implementierung DB (Jarke)
Verteilte DB (nie gehört, aber Mitschrift der Vorlesung von Jeusfeld)
Requirements Engineering (Pohl)

1) Requirements Engineering (ca. 25 Minuten)

F: Hilfsmittel beim RE?
A: SA: DFDs, Minispezifikationen und DDs
F: Was steht in den DDs
A: Daten über Daten, d.h. Metadaten
F: DFD-Symbole erklären
A: 1) Prozesse/Knoten als Kreise darstellen
2) Datenspeicher als parallele Linien „
3) Datenflüsse als gerichtete Kanten
4) Datenquellen oder -senken als Rechtecke
F: Welche „Constraints" gibt es in DFDs, d.h. was sollte nicht entworfen werden ?
A: 1) Datenspeicher, der nur Daten aufnimmt, oder nur Daten abgibt (tritt relativ selten auf, kann es aber durchaus geben)
2) Datenflüsse/Kanten immer benennen
3) Prozesse: es muß i.a. Kanten geben, die herein laufen und welche, die herausgehen
F: Was macht man vor der Erstellung des DFD ?
A: Man geht zum Kunden und erstellt mit ihm ein Konzept
(Jetzt wurde es etwas unangenehm :)
F: Und was macht man, wenn das nicht geht ?
A: Man analysiert das derzeit vorhandene System
F: und wenn es das nicht gibt ?
A: (Der Gedanke: „dann entwickelt man einfach eines und stellt es ihm vor", war mir zu einfach).
Ich fing an etwas über die 4 Welten des RE (unterschiedlich Ansichten von einem System) und die 3 Dimensionen des RE-Prozesses (Spezifikation, Repräsentation und Agreement) zu erzählen.
(Was er wirklich wissen wollte, blieb mir verborgen - vielleicht wollte er nur einen allgemeinen Begriffe wie „Brain-Storming" hören ?)
F: Wie sollte ein gutes DFD aussehen ?
A: nicht zu groß /(ca. 6-7 Prozesse), sonst aufgliedern in mehrere Ebenen und Projektion eines Knotens (welcher ein Teilsystem darstellt) auf eine Ebene ...
F: Was ist denn hier unbedingt (constraints !) zu beachten ?
A: (nach mehrmaligem hin und her) ... Sie meinen bestimmt, daß die mit der Umgebung in Verbindung stehenden Kanten die gleichen sein müssen (*trivial*)

2) Einführung in Datenbanken (20-25 Minuten)

F: Genau, aber so trivial ist das gar nicht ... Gehen wir über zu DB. Welche Objekte gibt es im ER-Diagramm ?

A: Entitäten, Beziehungen, Attribute (auch mehrwertige), Schlüssel, ISA-Bez., ...

F: Constraints ?

A: Kardinalitäten aufgezählt (er wollte die 0:1 1:1 0:n 1:n m:n - Notation haben !)

F: Wie überführt man so etwas in Relationen ?

A: Entitäten -> Relationen; mehrwertige Attribute -> Relationen; Beziehungen -> Relationen (oder bei 1:1 oder 1:n die Attribute an die Relationen der Entitäten hängen); ISAs -> (mehrere Möglichkeiten, die offensichtlich alle OK waren)
(zwischendurch meinte er mal, daß wir nicht schnell genug voran kämen).

F: Machen Sie doch mal eine SQL-Anfrage: Sie suchen alle Buchnummern, die mehr als 1x in der Datenbank vorkommen.

A: select nr from relation where count(nr)>1 and nr in (select nr,count(nr) from relation group by nr) ... (oder so ähnlich, aber dann gemeint, daß das wohl zu komplex würde, abgesehen davon, ob es überhaupt richtig sei)

F: Sie können das erste select weglassen, wenn Sie noch ein having anfügen

A: select nr,count(nr) from relation group by nr having (count(nr)>1) -- oder ähnlich --

F: Was macht die Dekomposition ?

A: Überführt in BCNF, verlustfrei aber nicht immer abhängigkeitserhaltend. Synthese überführt in 3.NF (abhängigkeitserhaltend und verlustfrei). Verlustfreiheit erklärt. Abhängigkeitserhaltung erklärt. (Dekompositions- und Synthesealgorithmen hat er nicht abgefragt !)

F: Wie ist eine FD definiert ?

A: Verbal erklärt (A->B : wenn 2 Tupel auf A übereinstimmen, dann auch auf B)

F: Genauer ?

A: Habe gefragt, ob er es wirklich ganz formell haben möchte, denn dazu müßte ich zuerst mal die Mengen (Attributmengen) festlegen ?

3) Implementierung von Datenbanken (10-15 Minuten)

F: Na gut, wenn Sie das nicht können ... Kommen wir zur Impl.DB (es waren schon ca.45-50 Minuten vergangen) 6 Schnittstellen der 5 Schichtenarchitektur und eine SQL-Anfrage sind gegeben. Erklären sie die Schnittstellen mit Hilfe der SQL-Anfrage:

A: 6 Schnittstellen aufgeschrieben und angefangen zu reden über alles, was mir dazu einfällt ... dabei B/B* - Bäume erwähnt, Hash erklärt, Nested-Loop und Merge/Sort-Join, Komplexität des Nested-Loop = n*m, des Merge/Sort-Join n*logn (geschätzt und er hat genickt) ..

F: einige Fragen, wie z.B. „Wie implementiert man das XXX zwischen Schicht i und i+1"

Fazit: Note 2.7 - ich hatte mehr erwartet. Angenehm war, daß er nichts zu Quantgraphen, Einfügen/Löschen in B/B*-Bäumen, o.ä. insbesondere aus Kapitel 2 Implementierung fragte.

Empfehlung: Ich kann die Prüfung in DB / Impl. DB nicht unbedingt weiter empfehlen. In den Vorlesungen zu viele Themen nur angeschnitten und die Folienkopien sind schwer verständlich. Die Literatur (Vossen) umfaßt ca. 500 Seiten, deckt aber nur 80 % des Stoffes der Vorlesungen DB und Impl.DB ab. Seht Euch auf jedem Fall die Prüfungsprotokolle an, so könnt Ihr leichter abschätzen, was Ihr unbedingt lernen solltet und was nicht. Sammelt alles über das Schichtenmodell und versucht soviel zu verstehen, daß Ihr über jede einzelne Schnittstelle ein paar Minuten referieren könnt.

Empfehlung: Buch Vossen
Buch Silberschatz/Korth
möglichst viele Prüfungsprotokolle (ca. 30-50)