Grundlagen: Datenbanken

Zentralübung / Wiederholung / Fragestunde

Linnea Passing   Harald Lang

gdb@in.tum.de

WiSe 2017 / 2018

Diese Folien finden Sie online.

Die Mitschrift stellen wir im Anschluss online.

Agenda

- Hinweise zur Klausur
- Stoffübersicht/-Diskussion
- Wiederholung + Übung
  - Mehrbenutzersynchronisation
  - Erweiterbares Hashing
  - Anfragebearbeitung/-optimierung
  - Datenbankentwurf
  - Relationale Algebra
  - Relationale Entwurfstheorie

Hinweise zur Klausur

Termine

- 1. Klausurtermin - **Mi. 28.02.2018, 8:00 bis 9:30 Uhr**
- Notenbekanntgabe / Anmeldung zu Einsicht - **Mi. 7.03.2018 (ab Mittag)**
- Einsicht - **Do. 8.03.2018 (Nachmittags)**
- Anmeldung zur 2. Klausur von - **ab 10.03.2018, bis ... ? (TBA)**
- 2. Klausurtermin - **TBA**

Verschiedenes

- **Raumbekanntgabe**, via TUMonline sowie in Moodle
- 90 Minuten / 90 Punkte
- Sitzplatzvergabe (Aushang: $MatrNr \mapsto Sitzplatz$, KEINE Namensnennung)
- Betrugsfälle
- Notenbekanntgabe
- Einsichtnahme (Instruktionen in Moodle, nach Notenbekanntgabe)
- Bonus: Gilt für beide Klausuren.

Stoffübersicht (1)

Datenbankentwurf / ER-Modellierung

- ER-Diagramme, Funktionalitäten, Min-Max, Übersetzung ER $\leftrightarrow$ Relational, Schemavereinfachung/-verfeinerung

Das Relational Modell

- Stichworte: Schema, Instanz/Ausprägung, Tupel, Attribute,...
- Anfragesprachen
  - Relationale Algebra
    - RA-Operatoren: Projektion, Selektion, Join (Theta, Natural, Outer, Semi, Anti), Kreuzprodukt, Mengendifferenz/-vereinigung/-schnitt, Division
  - Tupelkalkül, ~~Domänenkalkül~~ nicht prüfungsrelevant

Stoffübersicht (2)

SQL

- ...

Relationale Entwurfstheorie

- Definitionen:
  - Funktionale Abhängigkeiten (FDs), Armstrong-Axiome (+Regeln), FD-Hülle, Kanonische Überdeckung, Attribut-Hülle, Kandidaten-/Superschlüssel, Mehrwertige Abhängigkeiten (MVDs), Komplementregel, Triviale FDs/MVDs,...
- Normalformen: 1., 2., 3.NF, BCNF und 4. NF
- Zerlegung von Relationen
  - in 3.NF mit dem Synthesealgorithmus
  - in BCNF/4.NF (zwei Varianten des Dekompositionsalgorithmus)
  - Stichworte: Verlustlos, Abhängigkeitsbewahrend

Stoffübersicht (3)

- **Physische Datenorganisation**
  - Speicherhierarchie
  - HDD/RAID
  - TID-Konzept
  - Indexstrukturen (Bäume, Hashing)

- **Anfragebearbeitung**
  - Kanonische Übersetzung (SQL $\rightarrow$ Relationale Algebra)
  - Logische Optimierung (in relationaler Algebra)
    - Frühzeitige Selektion, Kreuzprodukte durch Joins ersetzen, Joinreihenfolge
  - Implementierung relationaler Operatoren
    - ...
    - Nested-Loop-Join
    - Sort-Merge-Join
    - Hash-Join
    - Index-Join

Stoffübersicht (4)

- **Transaktionsverwaltung**
  - BOT, read, write, commit, abort
  - Rollback (R1 Recovery)
  - ACID-Eigenschaften
- **Fehlerbehandlung (Recovery)**
  - Fehlerklassifikation (R1 - R4)
  - Protokollierung: Redo/Undo, physisch/logisch, Before/After-Image, WAL, LSN
  - Pufferverwaltung: Seite, FIX, Ersetzungsstrategie steal/¬steal, Einbringstrategie force/¬force
  - Wiederanlauf nach Fehler, Fehlertoleranz des Wiederanlaufs, Sicherungspunkte
- **Mehrbenutzersynchronisation**
  - Formale Definition einer Transaktion (TA)
  - Historien (Schedules)
    - Konfliktoperationen, (Konflikt-)Äquivalenz, Eigenschaften von Historien
  - Datenbank-Scheduler
    - pessimistisch (sperrbasiert, zeitstempelbasiert), optimistisch

Mehrbenutzersynchronisation

Transaktionen (High-Level)

- Ein Programm, das auf einem Datenbestand arbeitet.
  - Beispiele: Banküberweisung, Online-Bestellung, Ausleihe (Bib.)
- Daten werden gelesen, verarbeitet (Programmlogik) und geschrieben.

Atomarität

- Eine Transaktion überführt eine Datenbank von einem konsistenten Zustand in einen wiederum konsistenten Zustand.
- Zwischenzeitlich kann die Datenbank in einem inkonsistenten Zustand sein.
- Atomarität (*Alles-oder-nichts-Eigenschaft*):
  - Es werden entweder alle Änderungen übernommen, oder keine.
  - Schlägt während der Ausführung eine Operation fehl, werden alle bisherigen Änderungen in den Ausgangszustand zurück gesetzt.

A tomicity
C onsistency
I solation
D urability

Transaktionen (aus Sicht des Datenbanksystems)

Eine Transaktion T_i besteht aus folgenden **elementaren Operationen**:

$r_i(A)$	-	**Lesen** des Datenobjekts A
$w_i(A)$	-	**Schreiben** des Datenobjekts A
a_i	-	**Abort** (alle Änderungen rückgängig machen)
c_i	-	**Commit** (alle Änderungen festschreiben)

Die letzte Operation ist entweder ein **commit** oder ein **abort.**

Die dahinterliegende Programmlogik ist hier nebensächlich.

Historie (Schedule)

Eine Historie spezifiziert eine **zeitliche Abfolge von Elementaroperationen** mehrerer **parallel laufender Transaktionen** (*verzahnte Ausführung*).

$$H = r_1(A), r_2(C), w_1(A), w_2(C), r_1(B), w_1(B), r_2(A), w_2(A), c_1, c_2$$

T_1 T_2

*Eine Historie umfasst nicht zwangsläufig eine totale Ordnung ALLER Operationen, aber mindestens die der **Konfliktoperationen**.*

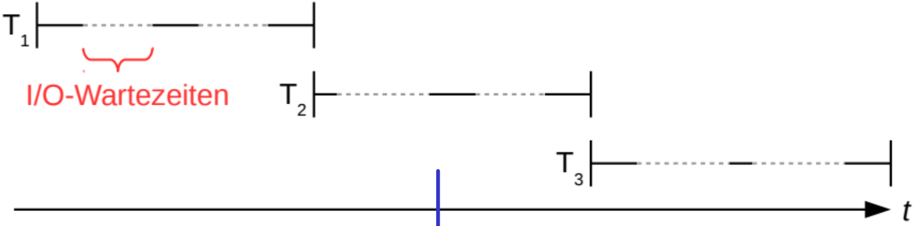
Konfliktoperationen

Zwei Operationen (verschiedener aktiver Transaktionen) auf dem selben Datum stehen zueinander in Konflikt, gdw. mindestens eine Operation schreibend ist.

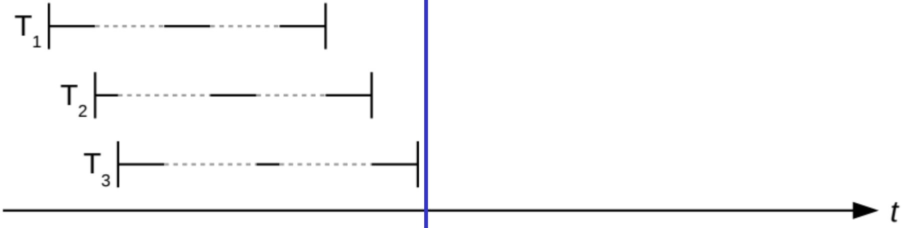
- **Unkontrollierte Nebenläufigkeit** kann zu Inkonsistenzen führen:
  - *lost update*
  - *dirty read*
  - *non-repeatable read*
  - *phantom problem*

Serielle vs. Parallele Ausführung

Eine **serielle Ausführung** verhindert all diese Probleme, da zu jedem Zeitpunkt maximal eine Transaktion aktiv ist und somit keine Konflikte auftreten können.



Eine verzahnte **parallele Ausführung** (im Mehrbenutzerbetrieb) ist effizienter.



Serialisierbarkeit (Konzept)

... soll die Vorzüge der seriellen Ausführung (**Isolation**) mit den Vorteilen des Mehrbenutzerbetriebs (**höherer Durchsatz**) kombinieren.

Serialisierbarkeit

Beispiel (Überweisung von $\overbrace{A \text{ nach } B}^{T_1}$ und von $\overbrace{C \text{ nach } A}^{T_2}$):

$$H = r_1(A), r_2(C), w_1(A), w_2(C), r_1(B), w_1(B), r_2(A), w_2(A), c_1, c_2$$

H:

	T_1	T_2
	$r_1(A)$	
$A := A - 10$		$r_2(C)$
	$w_1(A)$	
		$w_2(C)$
	$r_1(B)$	
$B := B + 10$	$w_1(B)$	
		$r_2(A)$
		$w_2(A)$
	c_1	
t		c_2

Konfliktop.

$\equiv$

H':

T_1	T_2
$r_1(A)$	
$w_1(A)$	
$r_1(B)$	
$w_1(B)$	
c_1	
	$r_2(C)$
	$w_2(C)$
	$r_2(A)$
	$w_2(A)$
	c_2

$H' = T_1 \mid T_2$

Serialisierbarkeitsgraph SG(H):

$$T_1 \rightarrow T_2$$

kein Zyklus → Serialisierbar

$H \equiv H'$ gdw. Konfliktoperationen in der gleichen Reihenfolge.

H ist **(konflikt-) serialisierbar.**

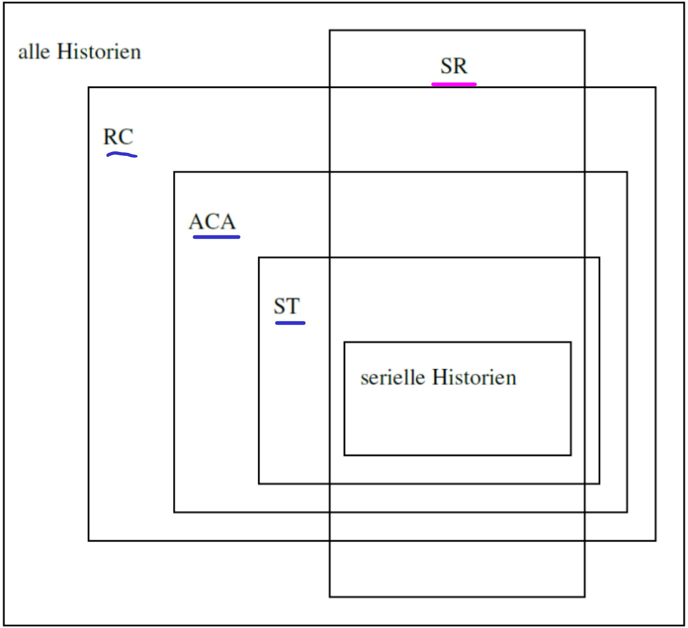
Serialisierbarkeitstheorem

H ist **serialisierbar**, gdw. SG(H) azyklisch ist.

Weitere Eigenschaften von Historien

- **Rücksetzbar (RC)**
  - Commit der schreibenden Transaktion T_j muss vor dem Commit der lesenden Transaktion T_i durchgeführt werden.
  - $c_j <_H c_i$
- **Vermeidet kaskadierendes Rücksetzen (ACA)**
  - Es wird erst gelesen, wenn die Änderungen der schreibenden Transaktion T_j festgeschrieben wurden (Commit).
  - $c_j <_H r_i$
- **Strikt (ST)**
  - Wie ACA, verhindert aber zusätzlich blindes Schreiben (ohne vorheriges Lesen).
  - $a_j <_H o_i$ oder $c_j <_H o_i$ (Operation o= r oder w)

Eigenschaften von Historien (Zusammenhang)



Eigenschaften von Historien: Übung

RC: T_3 liest von T_2 und T_4. $c_2, c_4 <_H c_2$ ✓

ACA: $c_2, c_4 <_H r_3(B)$ ✓

ST:

H:

Schritt	T_1	T_2	T_3	T_4
1	$w(A)$			
2				$r(B)$
3	$\neg$ ST	$w(A)$		
4				$w(B)$
5	c			
6				c
7		$w(B)$		
8		c		
9			$r(B)$	
10			$w(C)$	
11			c	

wahr	falsch	Aussage
✓		$H \in SR$
✓		$H \in RC$
✓		$H \in ACA$
	✓	$H \in ST$

SG(H):

$$\begin{tikzcd}
 & T_1 \arrow[d, "1"] \\
T_4 \arrow[r, "2"] \arrow[dr, "3"'] & T_2 \arrow[d, "4"] \\
 & T_3
\end{tikzcd}$$

Zyklenfrei

$H \equiv T_1 | T_4 | T_2 | T_3 \equiv T_4 | T_1 | T_2 | T_3$

Eigenschaften von Historien: Übung (2)

H:

Schritt	T_1	T_2	T_3
1	$r(A)$		
2		$w(A)$	
3		$r(B)$	
4	$w(B)$		
5	c		
6		c	
7			$r(A)$
8			$w(A)$
11			c

wahr	falsch	Aussage
X		$H \in RC$
X		$H \in ACA$
X		$H \in ST$
	X	$H \in SR$

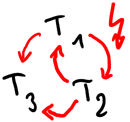
RC: T_3 liest von T_2. $c_2 < c_3$ ✓

ACA: —— —— $c_2 < r_3(A)$ ✓

ST: $w_2(A) < w_3(A)$

aber $c_2 < w_3(A)$

SG(H):



Datenbank-Scheduler

Der Datenbank-Scheduler "**ordnet**" die (eingehenden) **Elementaroperationen** der Transaktionen so, dass die resultierende Historie bestimmte Eigenschaften hat.

Mittel: -> kann Tx abbrechen

Er implementiert ein Synchronisationsverfahren und sorgt so für **kontrollierte Nebenläufigkeit**.

Synchronisationsverfahren

- Pessimistisch
  - Sperrbasiert lockS(x) vor r(x), lockX(x) vor w(x), unlock(x)
    - **2PL** Two Phase Locking
    - **Strenges 2PL**
  - Zeitstempel-basiert jede Tx bekommt einen (eindeutigen) Zeitstempel
- Optimistisch
  - inkl. abgeschwächter Form: Snapshot Isolation $\text{WriteSet}(T_{\text{andere}}) \cap \text{WriteSet}(T_i) = \emptyset$

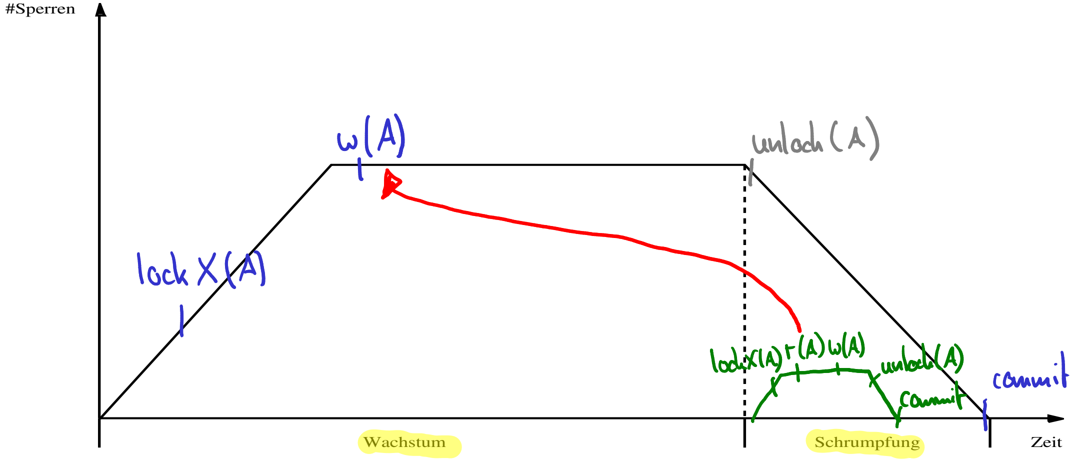
Lesephase (Änderungen nur lokal, keine Änderung an der Datenbasis)

Validierungsphase: Konflikte mit anderen Tx? $\text{WriteSet}(T_{\text{andere}}) \cap \text{ReadSet}(T_i) = \emptyset$

Schreibphase: Änderungen werden in die DB eingebracht.

zeitlich überlappende Txen

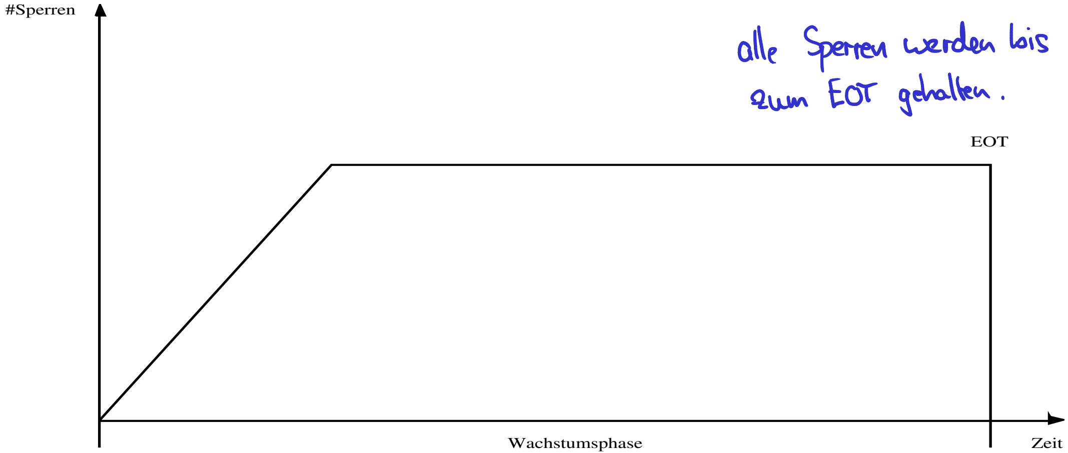
Zwei-Phasen-Sperrprotokoll (2PL)



$T_{grün}$ liest von T_{blau}

$c_{grün} <_H c_{blau}$ $\Rightarrow$ nicht RC!

Strenges 2PL



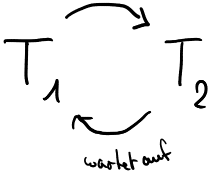
alle Sperren werden bis zum EOT gehalten.

Verklemmung (Deadlock)

Ein Ablauf zweier parallel laufender TAs:

Schritt	T_1	T_2	Bemerkung
1.	BOT		
2.		BOT	
3.	lockX(A)		
4.		lockX(B)	
5.	w(A)		
6.		w(B)	
7.	lockS(B)		Wartet auf T_2...
8.		lockS(A)	Wartet auf T_1...

Wartegraph:



Zyklus -> Deadlock

Zwei-Phasen-Sperrprotokoll (2PL)

Deadlockbehandlung

- **Vermeidung** durch preclaiming
- **Vermeidung** durch Zeitstempel an Tx
  - wound-wait
  - wait-die
- **Erkennung** durch Wartegraph

$(T_{alt} \rightarrow T_{jung}) - (T_{jung} \rightarrow T_{alt})$

wound - wait

wait - die

Erweiterbares Hashing

Erweiterbares Hashing

Hashfunktion $\boldsymbol{h}: \boldsymbol{S} \rightarrow \boldsymbol{B}$

Schlüssel — Bucket

wir betrachten die **Binärdarstellung** des Hashwerts

$h(x) = dp$

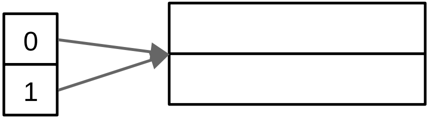
unbenutzte Bits

Anzahl betrachteter Bits
= globale Tiefe des Dictionaries

Initialer Zustand: (leere HT)

globale Tiefe $t = 1$

entspricht der Anzahl der betrachteten Bits

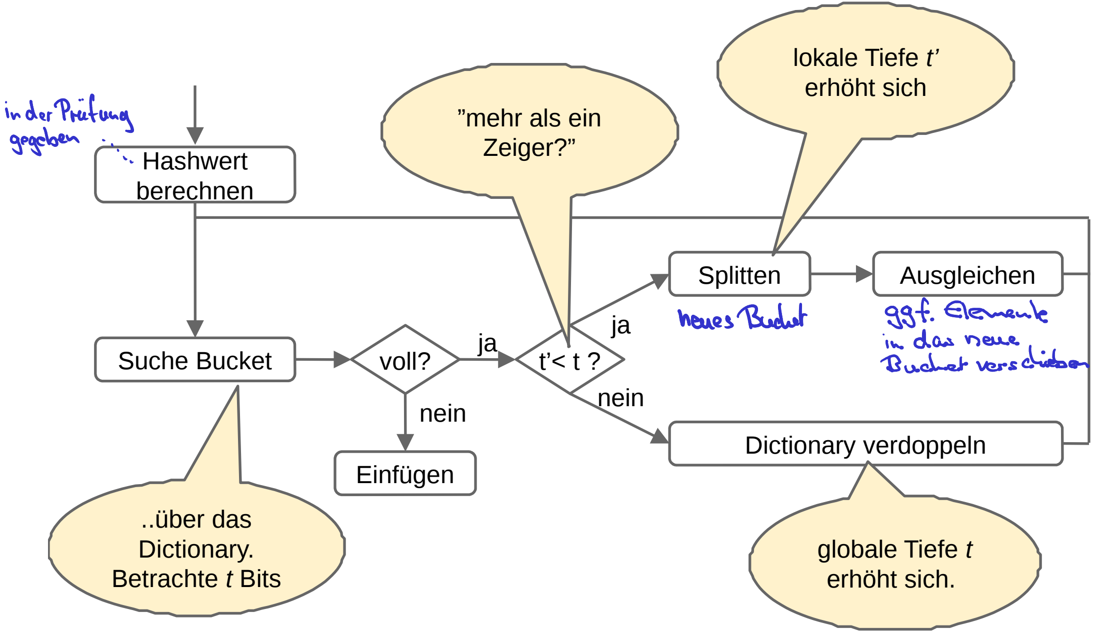


lokale Tiefe $t' = 0$

Dictionary (Verzeichnis)

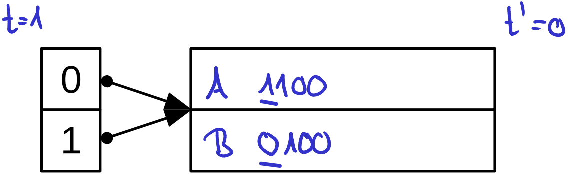
Bucket (Behälter)
Platz für n Einträge
hier n=2

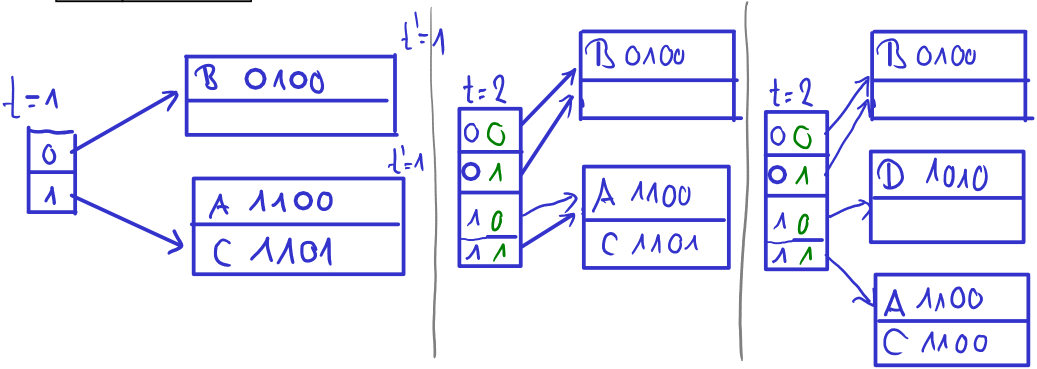
Erweiterbares Hashing / Einfügen



Übung: Erweiterbares Hashing / Einfügen

x	h(x)
A	1100 ✓
B	0100 ✓
C	1101 ✓
D	1010





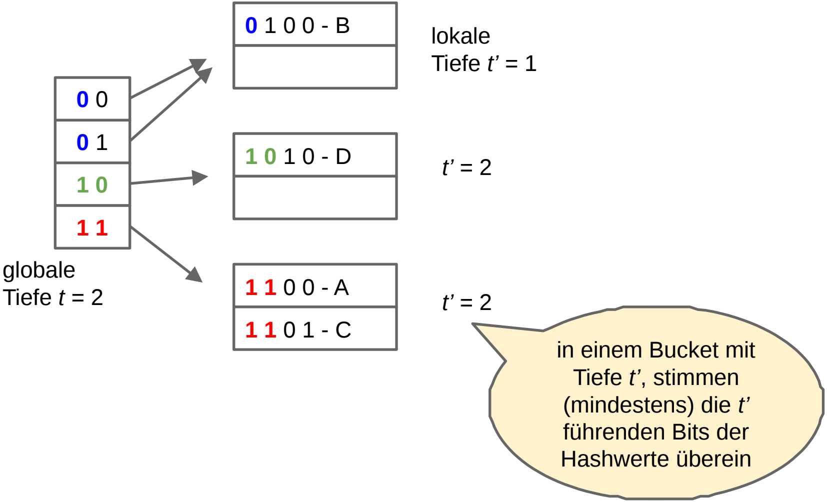
Übung: Erweiterbares Hashing / Einfügen

x	*h(x)*
A	1100
B	0100
C	1101
D	1010

Erweiterbares Hashing / Lösung

x	*h(x)*
A	1100
B	0100
C	1101
D	1010

d *p*

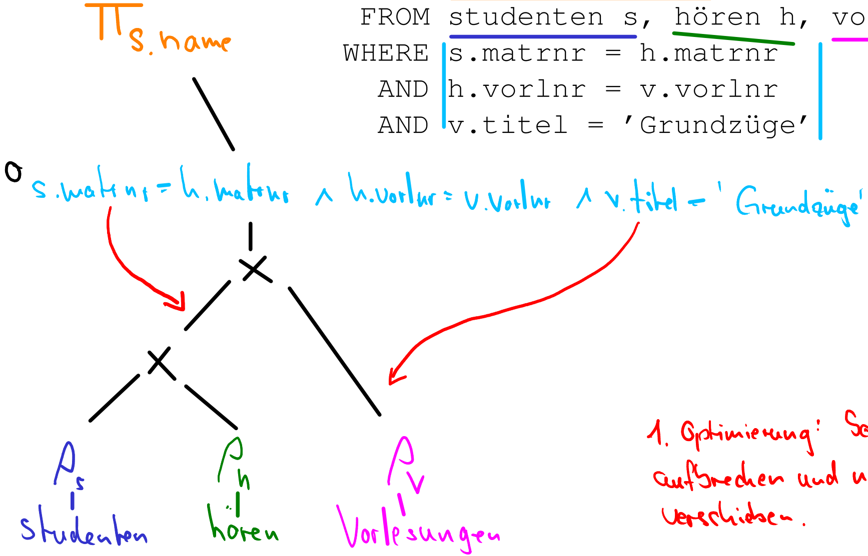


Anfrageoptimierung

Übung: Anfrageoptimierung

Geben Sie die kanonische Übersetzung der folgenden SQL-Anfrage an und optimieren Sie diese logisch:

```
SELECT DISTINCT s.name
  FROM studenten s, hören h, vorlesungen v
 WHERE s.matrnr = h.matrnr
   AND h.vorlnr = v.vorlnr
   AND v.titel = 'Grundzüge'
```



1. Optimierung: Selektionen aufbrechen und nach unten verschieben.

Übung: Anfrageoptimierung (2)

Angenommen

- $|s| = 10000$
- $|h| = 20 * |s| = 200000$
- $|v| = 1000$
- 10% der Studenten haben 'Grundzüge' gehört

Dann ergeben sich

- $|s \times h \times v| = 10000 \cdot 20 \cdot 10000 \cdot 1000 = 2 \cdot 10^{12}$

Nach der Selektion verbleiben noch

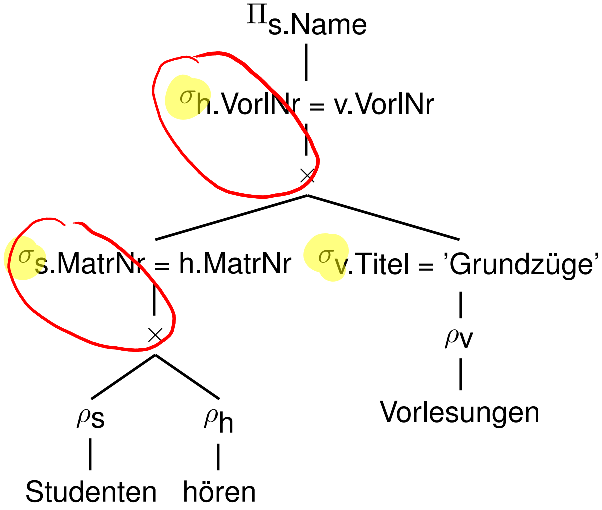
- $|\sigma_p(s \times h \times v)| = 0,1 \cdot |s| = 1000$

Übung: Anfrageoptimierung (3)

nach **Optimierung 1**: Selektionen frühzeitig ausführen (*push selections*):

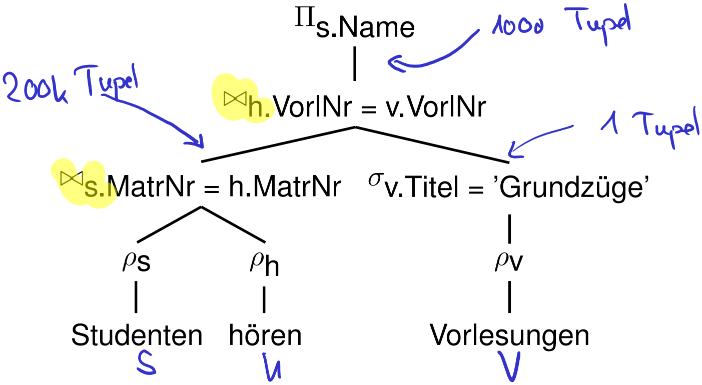
$\sigma_p(R \times S)$

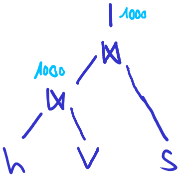
$= R \bowtie_p S$



Übung: Anfrageoptimierung (4)

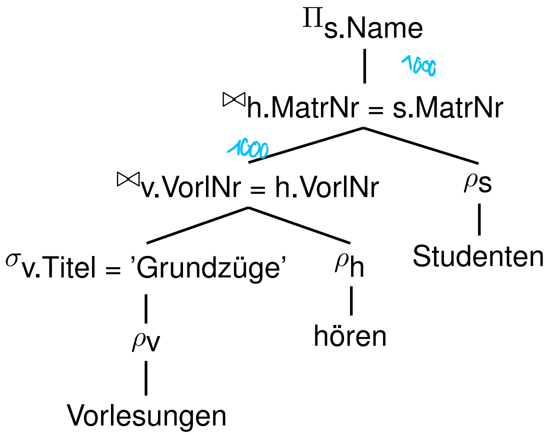
nach **Optimierung 2**: Kreuzprodukte durch Joins ersetzen (*introduce joins*):





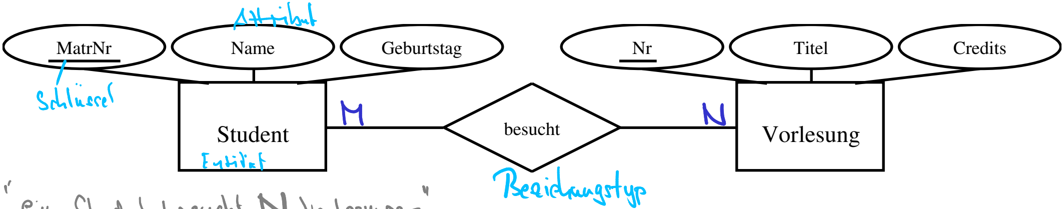
Übung: Anfrageoptimierung (5)

nach **Optimierung 3**: Joinreihenfolge optimieren (*join order optimization*), so dass die Zwischenergebnismengen möglichst klein sind:



Datenbankentwurf

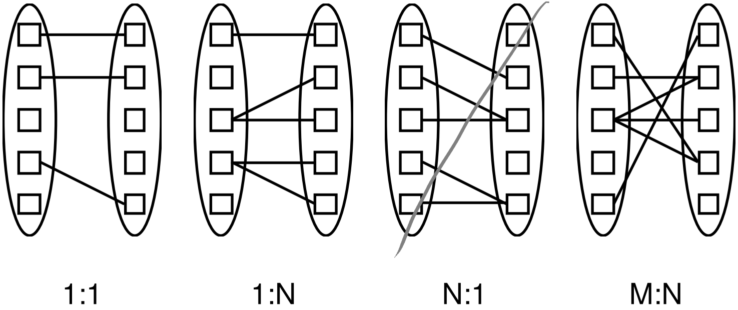
Datenbankentwurf



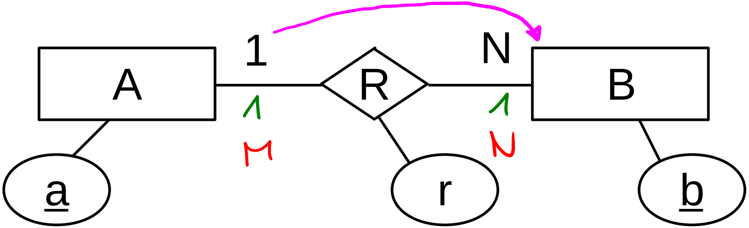
sprich: „ein Student besucht N Vorlesungen"

„eine Vorlesung wird besucht von M Studenten"

Funktionalitäten (Integritätsbedingungen)



ER-Modell in Schema überführen und verfeinern



Schema:

A: $\{\underline{a}\}$

B: $\{\underline{b}\}$

R: $\{a, \underline{b}, r\}$

Verfeinern: 1:N

A: $\{\underline{a}\}$

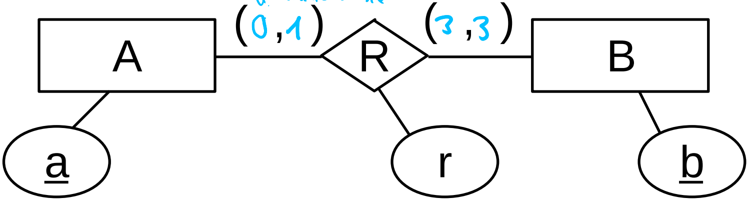
B: $\{\underline{b}, a, r\}$

(Min,Max) - Angaben

für Integritätsbedingungen

jedes A minimal 0x u. maximal 1x

jedes B genau 3x



R		
a	b	r
a1	b1	...
a2	b1	
a3	b1	
a4	b2	
a5	b2	
a6	b2	

1x

3x

Relationale Algebra

Algebraische Operatoren:

Projektion	$\Pi_{A_1,\ldots,A_n}$
Selektion	σ_p
Kreuzprodukt	$\times$
Verbund (Join)	$\bowtie_\theta$, $⟕_\theta$, $⟖_\theta$, $⟗_\theta$, $\ltimes_\theta$, $\rtimes_\theta$, $\rhd_\theta$, $\lhd_\theta$
Mengenoperationen	$\cup$, $\cap$, $\setminus$ *
Division !	$\div$ wichtig für All-quantifizierung
~~Gruppierung/Aggregation~~	$\Gamma_{\cancel{A_1,\ldots,A_n;a_1:f_1,\ldots,a_m:f_m}}$
Umbenennung	ρ_N, oder $\rho_{a_1 \leftarrow b_1,\ldots,a_n \leftarrow b_n}$

* '\' Differenz ≈ '–'
Minus

Anmerkung: Natural-Join vs. allgemeiner Theta-Join

$A: \{a, x\}$
$B: \{b, x\}$
$sch(A \bowtie B) = \{a, b, x\}$
$sch(A \bowtie_{A.x = B.x} B) = \{a, b, x, x\}$

	Natural	Theta
Inner	$\bowtie$	$\bowtie_\theta$
Outer	⟗, ⟕, ⟖	⟗$_\theta$, ⟕$_\theta$, ⟖$_\theta$
Semi	$\ltimes$, $\rtimes$	$\ltimes_\theta$, $\rtimes_\theta$
Anti	▷, ◁	▷$_\theta$, ◁$_\theta$

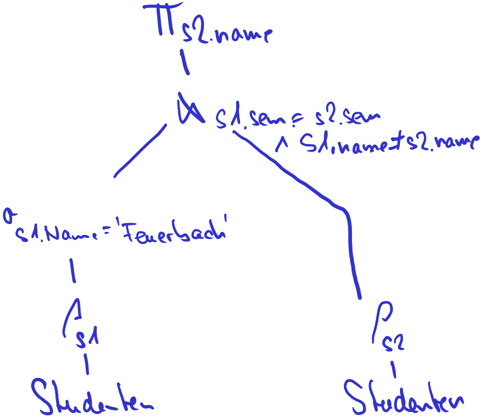
- Natural
  - Implizite Gleichheitsbedingung auf gleichnamigen Attributen
  - Die gleichnamigen Attribute tauchen im Ergebnis nur einmal auf (inner und outer).
- Theta
  - Explizite (beliebige) Joinbedingung: θ.

    $\bowtie_{s.MatrNr = h.MatrNr \wedge \ldots}$
  - Im Falle von Inner- und Outer-Join werden alle Attribute der beiden Eingaberelationen in das Ergebnis projiziert.

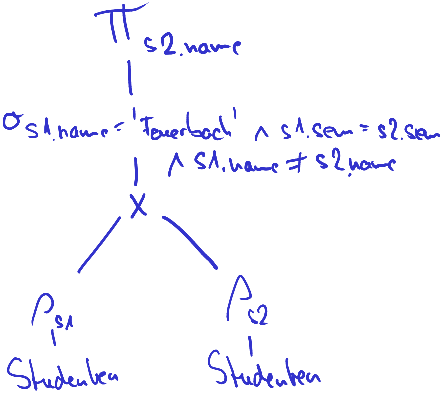
Übung: Relationale Algebra (1)

Finde Studenten (nur Namen ausgeben), die im gleichen Semester sind wie Feuerbach.

"auch okay" ;-)



logisch $\equiv$

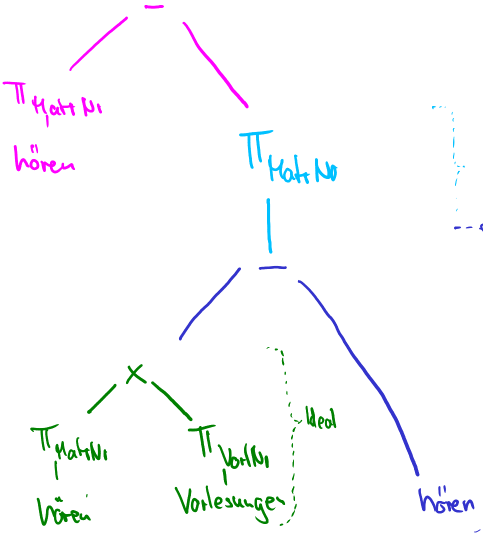


Übung: Relationale Algebra (2)

All-quantifizierung

Finde Studenten (nur MatrNr ausgeben), die alle Vorlesungen gehört haben.

über Negation



Stud. die nicht alles hören

Ideal

"Lüge"

nicht-hören

$$\text{hören} \div \Pi_{VorlNr}(\text{Vorlesungen})$$

$$\text{schema}(\{[\text{MatrNr}, \text{VorlNr}]\} \div \{[\text{VorlNr}]\}) = \{[\text{MatrNr}]\}$$

Relationale Entwurfstheorie

Relationale Entwurftheorie

Funktionale Abhängigkeiten (kurz FDs, für functional dependencies):

- Seien α und β Attributmengen eines Schemas $\mathcal{R}$.
- Wenn auf $\mathcal{R}$ die FD $\alpha \to \beta$ definiert ist, dann sind nur solche Ausprägungen R zulässig, für die folgendes gilt:
  - Für alle Paare von Tupeln $r, t \in R$ mit $r.\alpha = t.\alpha$ muss auch gelten $r.\beta = t.\beta$.

MatrNr → Name

Übung: Relationenausprägung vervollständigen

FD - Sudoku

Gegen seien die folgende Relationenausprägung und die funktionalen Abhängigkeiten. Bestimmen Sie zunächst x und danach y, sodass die FDs gelten.

1) $B \rightarrow A$

2) $AC \rightarrow D$

	A	B	C	D
0	7	3	5	8
1	x 1	4	2	8
2	7	3	6	9
3	1	4	2	y 8

Funktionale Abhängigkeiten

Seien $\alpha,\beta,\gamma,\delta \subseteq \mathcal{R}$

Axiome von Armstrong:

- *Reflexivität*: trivial
  Falls $\beta \subseteq \alpha$, dann gilt immer $\alpha \rightarrow \beta$
- *Verstärkung*:
  Falls $\alpha \rightarrow \beta$ gilt, dann gilt auch $\alpha\gamma \rightarrow \beta\gamma$
- *Transitivität*:
  Falls $\alpha \rightarrow \beta$ und $\beta \rightarrow \gamma$ gelten, dann gilt auch $\alpha \rightarrow \gamma$

i.d.R. sind nicht alle geltenden FD angegeben; insbesondere nicht die trivialen

Mithilfe dieser Axiome können alle *geltenden* FDs hergeleitet werden.

Sei F eine FD-Menge. Dann ist F^+ die Menge aller geltenden FDs (Hülle von F)

Funktionale Abhängigkeiten

Nützliche und vereinfachende Regeln:

- *Vereinigungsregel*:
  Falls $\alpha \to \beta$ und $\alpha \to \gamma$ gelten, dann gilt auch $\alpha \to \beta\gamma$
- *Dekompositionsregel*:
  Falls $\alpha \to \beta\gamma$ gilt, dann gilt auch $\alpha \to \beta$ und $\alpha \to \gamma$

  und umgekehrt gilt auch
- *Pseudotransitivitätsregel*:
  Falls $\alpha \to \beta$ und $\gamma\beta \to \delta$ gelten, dann gilt auch $\gamma\alpha \to \delta$

Schlüssel

$R: \{A, B, C, D\}$ z.B. $\{AB\} \rightarrow R$

$\{A\} \rightarrow R$

- Schlüssel identifizieren jedes Tupel einer Relation $\mathcal{R}$ eindeutig.
- Eine Attributmenge $\alpha \subseteq \mathcal{R}$ ist ein **Superschlüssel**, gdw. $\alpha \rightarrow \mathcal{R}$
- Ist α zudem noch *minimal*, ist es auch ein **Kandidatenschlüssel** (meist mit κ bezeichnet)
  - Es existiert also kein $\alpha' \subset \alpha$ für das gilt: $\alpha' \rightarrow \mathcal{R}$

- I.A. existieren mehrere Super- und Kandidatenschlüssel.
- Man muss sich bei der Realisierung in SQL für einen Kandidatenschlüssel entscheiden, dieser wird dann **Primärschlüssel** genannt.
- Der triviale Schlüssel $\alpha = \mathcal{R}$ existiert immer.

denn: $R \rightarrow R$

Übung: Schlüsseleigenschaft von Attributmengen ermitteln

- Ob ein gegebenes α ein Schlüssel ist, kann mithilfe der Armstrong Axiome ermittelt werden (i.A. zu aufwendig!)
- Besser: Die **Attributhülle** $AH(\alpha)$ bestimmen.
- Beispiel: $\mathcal{R} = \{A, B, C, D\}$, mit $F_{\mathcal{R}} = \{\underbrace{AB \rightarrow CD}_{1}, \underbrace{B \rightarrow C}_{2}, \underbrace{D \rightarrow B}_{3}\}$

ist D ein Superschlüssel?

–> $AH(\{D\})$: $D \overset{3}{\rightsquigarrow} B,D \overset{2}{\rightsquigarrow} \overbrace{B,C,D}^{\neq \mathcal{R}}$ kein Schlüssel in $\mathcal{R}$

$AH(\{A, D\})$: $AD \overset{3}{\rightsquigarrow} ABD \overset{2 \text{ oder } 1}{\rightsquigarrow} ABCD = \mathcal{R}$ ✓ Schlüssel in $\mathcal{R}$

$AH(\{A, B, D\})$: ✓ Schlüssel, da Teilmenge $\{A,D\}$ bereits ein Schlüssel ist.

↳ sicher kein Kandidatenschlüssel, weil nicht minimal

Mehrwertige Abhängigkeiten

multivalued dependencies (MVDs)

"Halb-formal":

- Seien α und β disjunkte Teilmengen von $\mathcal{R}$
- und $\gamma = (\mathcal{R} \backslash \alpha) \backslash \beta$ $\rightarrow$ $\alpha \cup \beta \cup \gamma = \mathcal{R}$
- dann ist β mehrwertig abhängig von α ($\alpha \twoheadrightarrow \beta$), wenn in jeder gültigen Ausprägung von $\mathcal{R}$ gilt:
- Bei zwei Tupeln mit gleichem α-Wert kann man die β-Werte vertauschen, und die resultierenden Tupel müssen auch in der Relation enthalten sein.

Wichtige Eigenschaften:

- Jede FD ist auch eine MVD (gilt i.A. nicht umgekehrt)
- wenn $\alpha \twoheadrightarrow \beta$, dann gilt auch $\alpha \twoheadrightarrow \gamma$ (Komplementregel)
- $\alpha \twoheadrightarrow \beta$ ist trivial, wenn $\beta \subseteq \alpha$ ODER $\alpha \cup \beta = \mathcal{R}$ (also $\gamma = \emptyset$)

Beispiel: Mehrwertige Abhängigkeiten

Beispiel: $R = \{\text{Professor}, \text{Vorlesung}, \text{Assistent}\}$

2

MVD 1 : $\underbrace{\text{Prof}}_{\alpha} \twoheadrightarrow \underbrace{\text{Vorl}}_{\beta}$

$(R \setminus \alpha) \setminus \beta = \gamma = \text{Assi}$

MVD 1 und 2 sind komplementär

ProfessorIn	Vorlesung	AssistentIn
K	GDB	Linnea
K	WebDB	Linnea
K	GDB	Harald
K	WebDB	Harald
K	ERDB	Linnea
K	ERDB	Harald

+ Assi "Harald" → 2 neue Tupel

+ Vorl. 'ERDB'

Normalformen: 1NF $\supset$ 2NF $\supset$ 3NF $\supset$ BCNF $\supset$ 4NF

$R: \{[A, \{B_1, B_2\}, C]\}$ ↯

- **1. NF**: Attribute haben nur atomare Werte, sind also nicht mengenwertig.
- **2. NF**: Jedes Nichtschlüsselattribut (NSA) ist voll funktional abhängig von jedem Kandidatenschlüssel.
  in keinem Kandidatenschl. enthalten
  - β hängt **voll funktional** von α ab ($\alpha \xrightarrow{\bullet} \beta$), gdw. $\alpha \to \beta$ und es existiert kein $\alpha' \subset \alpha$, so dass $\alpha' \to \beta$ gilt.
    Bsp: $\alpha = \{A, B\}$ und es gilt $B \to C$ ↯
- **3. NF**: Frei von transitiven Abhängigkeiten (*in denen NSAe über andere NSAe vom Schlüssel abhängen*).
  $\underline{A} \to B \to C$ ↯
  - für alle geltenden nicht-trivialen FDs $\alpha \to \beta$ gilt entweder
    - α ist ein Superschlüssel, oder
    - jedes Attribut in β ist in einem Kandidatenschlüssel enthalten
- **BCNF**: Die linken Seiten (α) aller geltenden nicht-trivalen FDs sind Superschlüssel.
- **4. NF**: Die linken Seiten (α) aller geltenden nicht-trivalen MVDs sind Superschlüssel.

Axiome Definition

Übung: Höchste NF bestimmen

$$\mathcal{R} : \{[\ A\ ,\ B\ ,\ C\ ,\ D\ ,\ E\ ]\}$$

1) $A \rightarrow BCDE$

2) $AB \rightarrow C$

- ○ 1. NF ✓
- ○ 2. NF ✓
- ○ 3. NF ✓
- ○ BCNF ✓
- ☒ 4. NF ✓
- ○ keine der angegebenen

NF1: ✓ keine mengenwertige Attribute

NF2: $K = \{A\}$ $NSA = \{BCDE\}$

nur 1 Attribut
nur 1 K
=> in 2. NF

NF3: ✓ ausführlich $\underline{A} \rightarrow BCDE$
ist Superschlüssel

$\underline{AB} \rightarrow C$
ist Superschlüssel, da A Superschl. ist

BCNF: ✓

NF4: ✓ da in BCNF und keine MVD gegeben. … die nicht auch FD's sind.

Übung: Höchste NF bestimmen (2)

$\mathcal{R} : \{[\ A, B, C, D, E\ ]\}$

$$A \rightarrow BCDE$$
$$B \rightarrow C$$

- ○ 1. NF ✓
- ☒ 2. NF ✓
- ○ 3. NF ✗
- ○ BCNF
- ○ 4. NF
- ○ keine der angegebenen

NF1: ✓
NF2: ✓ $\mathcal{K} = \{A\}$ NSAs $= \{BCDE\}$
NF3: ✗ $B \rightarrow C$ ↯

linke Seite kein Schlüssel

rechte Seite nicht in Kandidatenschlüsseln enthalten

weitere Übungsaufgaben → zu Folien früherer Semester

Schema in 3. NF überführen

verlustlos und Abhängigkeitsbewahrend

Synthesealgorithmus

überflüssige Attr. aus den FDs entfernen

- Eingabe:
  - **Kanonische Überdeckung** $\mathcal{F}_c$
    - Linksreduktion
    - Rechtsreduktion
    - FDs der Form $\alpha \rightarrow \emptyset$ entfernen (sofern vorhanden)
    - FDs mit gleicher linke Seite zusammenfassen

der aufwendige Teil

- Algorithmus:
  1. Für jede FD $\alpha \rightarrow \beta$ in $\mathcal{F}_c$ forme ein Unterschema $\mathcal{R}_\alpha = \alpha \cup \beta$, ordne $\mathcal{R}_\alpha$ die FDs $\mathcal{F}_\alpha := \{\alpha' \rightarrow \beta' \in \mathcal{F}_c \mid \alpha' \cup \beta' \subseteq \mathcal{R}_\alpha\}$ zu
  2. Füge ein Schema $\mathcal{R}_\kappa$ mit einem Kandidatenschlüssel hinzu
  3. Eliminiere redundante Schemata, d.h. falls $\mathcal{R}_i \subseteq \mathcal{R}_j$, verwerfe $\mathcal{R}_i$
- Ausgabe:
  - Eine Zerlegung des unsprünglichen Schemas, wo alle Schemata in 3.NF sind.
  - Die Zerlegung ist **abhängigkeitsbewahrend** und **verlustfrei**.

Übung: Synthesealgorithmus

$$\mathcal{R} : \{[\ A\ ,\ B\ ,\ C\ ,\ D\ ,\ E\ ,\ F\ ]\}$$

1) $B \rightarrow ACDEF$
2) $EF \rightarrow BC$
3) $A \rightarrow D$

$K_1 = \{B\}$

$K_2 = \{E, F\}$

F_c: LR: FD2: $EF \rightarrow BC$

$AH(\{F\}) = \{F\} \not\ni B$

$AH(\{E\}) = \{E\} \not\ni B$

RR:

FD1: $B \rightarrow A\cancel{C}\cancel{D}EF$

FD2: $EF \rightarrow BC$

FD3: $A \rightarrow D$

$\Big\} F_c$

Synthese:

Lösung:

$R_1 : \{\underline{B}, A, E, F\}$

$R_2 : \{\underline{E, F}, B, C\}$

$R_3 : \{\underline{A}, D\}$

~~$R_K : \{B\}$ oder $\{E, F\}$~~ bereits enthalten

eliminiere

Schema in BCNF überführen

BCNF-Dekompositionsalgorithmus (nicht abhängigkeitsbewahrend)

- Starte mit $Z = \{\mathcal{R}\}$ Zerlegung
- Solange es noch ein $\mathcal{R}_i \in Z$ gibt, das nicht in BCNF ist:
  - Finde eine FD $(\alpha \to \beta) \in F^+$ mit
    - $\alpha \cup \beta \subseteq \mathcal{R}_i$ (FD muss in $\mathcal{R}_i$ gelten)
    - $\alpha \cap \beta = \emptyset$ (linke und rechte Seite sind disjunkt)
    - $\alpha \to \mathcal{R}_i \notin F^+$ (linke Seite ist kein Superschlüssel) -> BCNF verletzt
  - Zerlege $\mathcal{R}_i$ in $\mathcal{R}_{i.1} := \alpha \cup \beta$ und $\mathcal{R}_{i.2} := \mathcal{R}_i - \beta$
  - Entferne $\mathcal{R}_i$ aus Z und füge $\mathcal{R}_{i.1}$ und $\mathcal{R}_{i.2}$ ein, also $Z := (Z - \{\mathcal{R}_i\}) \cup \{\mathcal{R}_{i.1}\} \cup \{\mathcal{R}_{i.2}\}$

Schema in 4.NF überführen

4NF-Dekompositionsalgorithmus (nicht abhängigkeitsbewahrend)

- Starte mit $Z = \{\mathcal{R}\}$
- Solange es noch ein $\mathcal{R}_i \in Z$ gibt, das nicht in 4NF ist:
  - Finde eine MVD $\alpha \twoheadrightarrow \beta \in \mathcal{F}^+$ mit
    - $\alpha \cup \beta \subset \mathcal{R}_i$ (FD muss in $\mathcal{R}_i$ gelten)
    - $\alpha \cap \beta = \emptyset$ (linke und rechte Seite sind disjunkt)
    - $\alpha \rightarrow \mathcal{R}_i \notin \mathcal{F}^+$ (linke Seite ist kein Superschlüssel)
  - Zerlege $\mathcal{R}_i$ in $\mathcal{R}_{i.1} := \alpha \cup \beta$ und $\mathcal{R}_{i.2} := \mathcal{R}_i - \beta$
  - Entferne $\mathcal{R}_i$ aus Z und füge $\mathcal{R}_{i.1}$ und $\mathcal{R}_{i.2}$ ein, also $Z := (Z - \{\mathcal{R}_i\}) \cup \{\mathcal{R}_{i.1}\} \cup \{\mathcal{R}_{i.2}\}$

Übung: BCNF-Dekompositionsalgorithmus

$$\mathcal{R} = \{A, B, C, D, E, F\},\ F_{\mathcal{R}} = \{B \overset{1}{\to} AD,\ DEF \overset{2}{\to} B,\ C \overset{3}{\to} AE\}$$

Finde eine FD, die die BCNF verletzt:

- $B \to AD$:
  kein Schlüssel
  Zerlege in $\mathcal{R}_1 : \{\underline{B}, A, D\}_{1}$ $\mathcal{R}_2 : \{B, C, E, F\}_{3''}$
  damit wird B Schlüssel

FD 2 geht verloren
FD 3 "teilweise"

FD 3: $C \to AE$
Dekomp.regel
FD 3': $C \to A$
FD 3'': $C \to E$

- $C \to E$:
  kein Schlüssel
  Zerlege in $\mathcal{R}_{2.1} : \{\underline{C}, E\}_{3''}$ $\mathcal{R}_{2.2} : \{\underline{B, C, F}\}$
  trivialer Superschlüssel, da nur noch triviale FDs gelten.

Lösung
kenntlich machen ;)

Fragen ?

Wir wünschen viel Erfolg. :-)