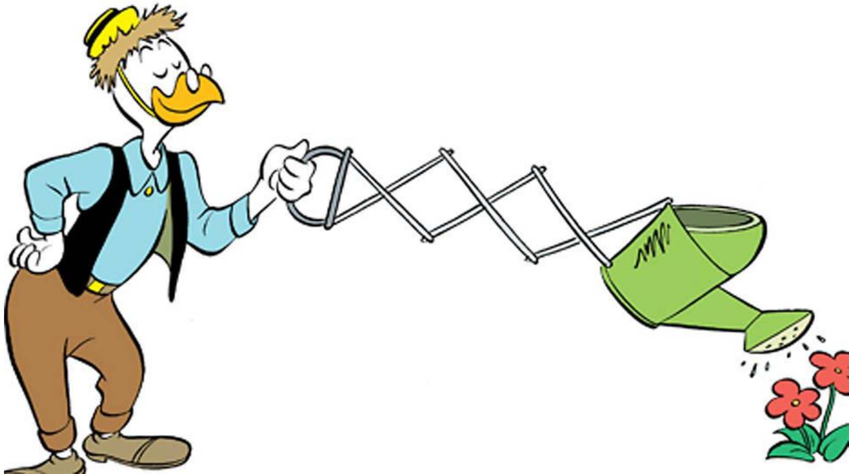
IT Systems Engineering | Universität Potsdam

Übung Datenbanksysteme II

Anfrageoptimierung

Maximilian Jenders

Folien basierend auf
Thorsten Papenbrock



Hausaufgaben 3 + 4

- Praktische Übung: Feedback, Probleme?
  - Java-Aufgabe
  - Picasso-Aufgabe
    - Stabilität des Programms

Aufgabe 1: Ausführungsstrategien

Betrachte die Relation $R(a,b,c,d)$, die 1 Million Tupel enthält. Nimm an, dass das Attribut a Schlüsselkandidat für R ist und die Werte von a zwischen 0 und 999.999 liegen. Jeder Block der Relation R fasst 10 Tupel. Außerdem sind die Tupel in R nicht sortiert.

Benenne für jede der folgenden Anfragen diejenige Ausführungsstrategie, die die wahrscheinlich wenigsten I/Os für die Bearbeitung der Anfrage benötigt. Begründe deine Antwort durch Angabe der I/Os für *alle* Ausführungsstrategien. Die beiden aufgeführten Indexe existieren bereits und müssen nicht für eine Anfrage neu angelegt werden. Sie liegen allerdings auf der Festplatte, so dass für genutzte Index-Strukturen ebenfalls I/O-Kosten anfallen. Falls nötig, triff geeignete Annahmen bzgl. benötigter Parameter und deren Werte!

Die zu betrachtenden Ausführungsstrategien sind:

- Scannen der kompletten Relation R.
- Nutzen eines B+-Baum-Indexes für $R.a$.
- Nutzen eines Hash-Indexes für $R.a$.

Die Anfragen sind:

a) Finde alle Tupel aus R. **3 P**

b) Finde alle Tupel aus R mit $a < 50$. **3 P**

c) Finde alle Tupel aus R mit $a = 50$. **3 P**

d) Finde alle Tupel aus R mit $a > 50$ und $a < 100$. **3 P**

Hausaufgabe 4:
Aufgabe 1

- Berechnung der I/O Kosten vom Hash-Index war größte Fehlerquelle
- B+Baum gab auch einige Fehler (z.B. Auslesen des gesamten Baumes bei den Bereichsanfragen statt Suche nach dem ersten Element und anschließendes Traversieren)

- Schlüsselkandidat != Schlüssel!

Hausaufgabe 4:
Aufgabe 1

Strategie	R	R, a<50	R, a=50	R, 50<a<100
Full Scan	$10^6/B$ $=10^5$	$10^6/B$ $=10^5$	$10^6/B$ $=10^5$	$10^6/B$ $=10^5$
B^+-Baum	$(h-1)+10^6/M+10^6$ $\approx 10^6$	$(h-1)+50/M+50$ ≈ 50	$(h-1)+1+1$ ≈ 3	$(h-1)+50/M+49$ ≈ 49
Hash-Index	$10^6/(N*c)+10^6$ $\approx 10^6$	$50+50$ $=100$	$1+1$ $=2$	$49+49$ $=98$

- B: #Tupel/Block (=10)
- M: #Schlüssel-Pointer-Pare/Blatt-Block (>10)
- h: Höhe des B^+-Baums
- N: #Daten-Pointer/Hash-Block (>10)
- c: Füllgrad eines Hash-Blocks

- Warum?

Hausaufgabe 4:
Aufgabe 2

- Aufgabenstellung nicht richtig gelesen oder falsch verstanden
- Berechnung von Werten wurde nicht angegeben, sondern es gab plötzliche magische Werte, mit denen die richtige Lösung ermittelt werden konnte
- 2b)

Aufgabe 2: Sort-Merge Join-Algorithmus

Gegeben sind zwei Relationen $R(\underline{A}, B)$ und $S(\underline{A}, C)$. Die Tupel in R umfassen $2^{13} = 8.192$ Blöcke und die Tupel in S umfassen $2^{10} = 1.024$ Blöcke. Für die Berechnung von $R \bowtie S$ können insgesamt $2^7 = 128$ Blöcke des Hauptspeichers genutzt werden.

a) Wie viele sortierte Teillisten entstehen in der ersten Phase des Two-Phase Multiway Merge-Sort (TPMMS) bei der Sortierung von R? Wie viele für S? **2 P**

b) Wie viele Blöcke müsste eine der beiden Eingaberelationen *mindestens* umfassen, damit die Sortierung mittels des TPMMS *nicht* mehr möglich ist? **2 P**

Eine Sortierung mit TPMMS ist dann nicht mehr möglich, wenn nicht von jeder Teilliste mindestens ein Block plus ein Outputblock in den Hauptspeicher passt. Das ist der Fall, wenn $B(R)/M > M-1 \Leftrightarrow x/2^7 > 2^7-1 \Leftrightarrow x > 2^{14} - 2^7 = 16.256$, also wenn eine Relation 16.257 oder mehr Blöcke umfasst.

Hausaufgabe 4:
Aufgabe 3

- A3:
  - Es wurde angenommen, dass die Formeln des Hybrid Hash Joins anwendbar sind
  - Der Hash Join wurde noch nicht (gut) verstanden, insbesondere die Unterscheidung zwischen Hashen und Joinen und die Auswirkungen auf die Bucket-Anzahl
  - Erläuterungen/Begründungen waren unvollständig

Hausaufgabe 4:
Aufgabe 3

- Grundalgorithmus
- Gegeben M Speicherblöcke, verteile R auf M−1 Buckets
  - Möglichst gleicher Größe
  - Ein Bucket pro Speicherblock
- Letzter Speicherblock für Einlesen der Tupel aus R

- Idee
  - Für jedes Tupel aus R berechne $h(t)$ und bewege Tupel in entsprechenden Bucket.
  - Falls Block voll: Schreibe als Overflowblock auf Disk
  - Am Ende: Schreibe auch alle Buckets auf Disk

Hashjoin

- Algorithmus wie zuvor
- Aber: Hashschlüssel sind Joinattribute
  - Tupel mit gleichen Joinattributwerten landen im korrespondierenden Bucket.
- Danach One-pass Join Variante für jedes Bucket-Paar
- Beispiel von zuvor: B(R) = 1000, B(S) = 500, M = 101
- Hashing
  - Ca. 10 R-Blocks pro Bucket
  - Ca. 5 S-Blocks pro Bucket
- Min(10, 5) = 5 => One-pass Algorithmus klappt (5 < 101)
  - Hole ersten S-Bucket in Hauptspeicher;
  - Joine Blöcke des passenden R-Buckets hinzu
  - Hole nächsten S-Bucket in Hauptspeicher … usw …
- I/O-Kosten:
  - 1500 für das Hashing + 1500 um Buckets zu schreiben
  - 1500 um Buckets zu lesen
  - Zusammen: 3(B(R) + B(S)) = 4500 (wie sort-basierte Methode)

Aber es geht noch besser.

Hausaufgabe 4:
Aufgabe 3

I/O Einsparungen

Grundidee: Nutze nicht-verwendeten Speicher

- Idee 1: Verwende mehr als 1 Speicherblock pro Bucket
  - Effizienteres Schreiben (aber gleiche I/O-Kosten)
- Idee 2: Hybrid Hashjoin
  - Beim Hashen von S: Behalte *m* Buckets komplett im Speicher
    - Auch nach Ende des Hashens
    - Jeweils mit geeigneter Datenstruktur

Hausaufgabe 4:
Aufgabe 4

- A4:
  - Indizes können benutzt werden, um weniger I/O-Kosten zu erreichen.
  - Kostenmodelle mit 1 I/O pro Tupel und 1 I/O pro Block gemischt ohne zu erklären weshalb

Zu Hausaufgabe 3:

Aufgabe 4

- SpieltMit vollständig lesen
- ‚King Kong'-Titel selektieren
- Über Index die Schauspieler suchen
- ‚w'-Geschlechter selektieren

…mentierung

…n Schauspielerinnen des Films 'King Kong' sucht:

```
SELECT *
FROM SpieltMi…   …pieler S
WHERE M.name =…
AND titel='King…   …ND g…
```

Die angefragten Relationen…

```
SpieltMit(titel, …   …ame…
Schauspieler(name, …   …sse,…
```

In Relationaler Algebra lässt sic… …Anfr…

$\sigma_{titel='KingKong'}$(SpieltMit) $\bowtie$ … …schle…

- SpieltMit vollständig lesen
- ‚King Kong'-Titel selektieren
- Ergebnis sortieren/hashen
- Schauspieler vollständig lesen
- ‚w'-Geschlechter selektieren
- Ergebnis sortieren/hashen
- Joinen

Nimm an, dass beide Relationen je… …eils etwa 10.000 … …en und es e… …dex auf Schauspieler.name gibt.

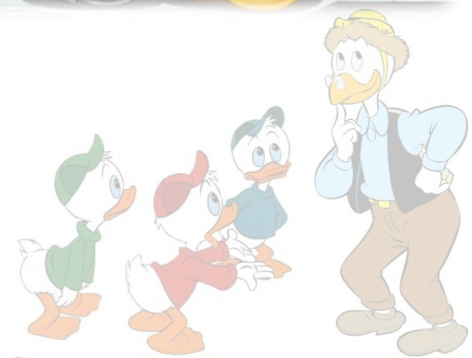
Vergleiche die I/O-Kosten eines Index-basierten, eines Sort-basierten und eines Hash-basierten Joins für die gegebene Anfrage. Falls nötig, triff geeignete Annahmen bzgl. der Parameter der einzelnen Join-Implementierungen (z.B. Blockgrößen). Wähle für jede Join-Variante außerdem die jeweils optimale Ausführungsreihenfolge von Selektionen und Join. Welche Algorithmus-Klasse sollte die Datenbank hier wählen?

4 P

Einleitung:
Themen

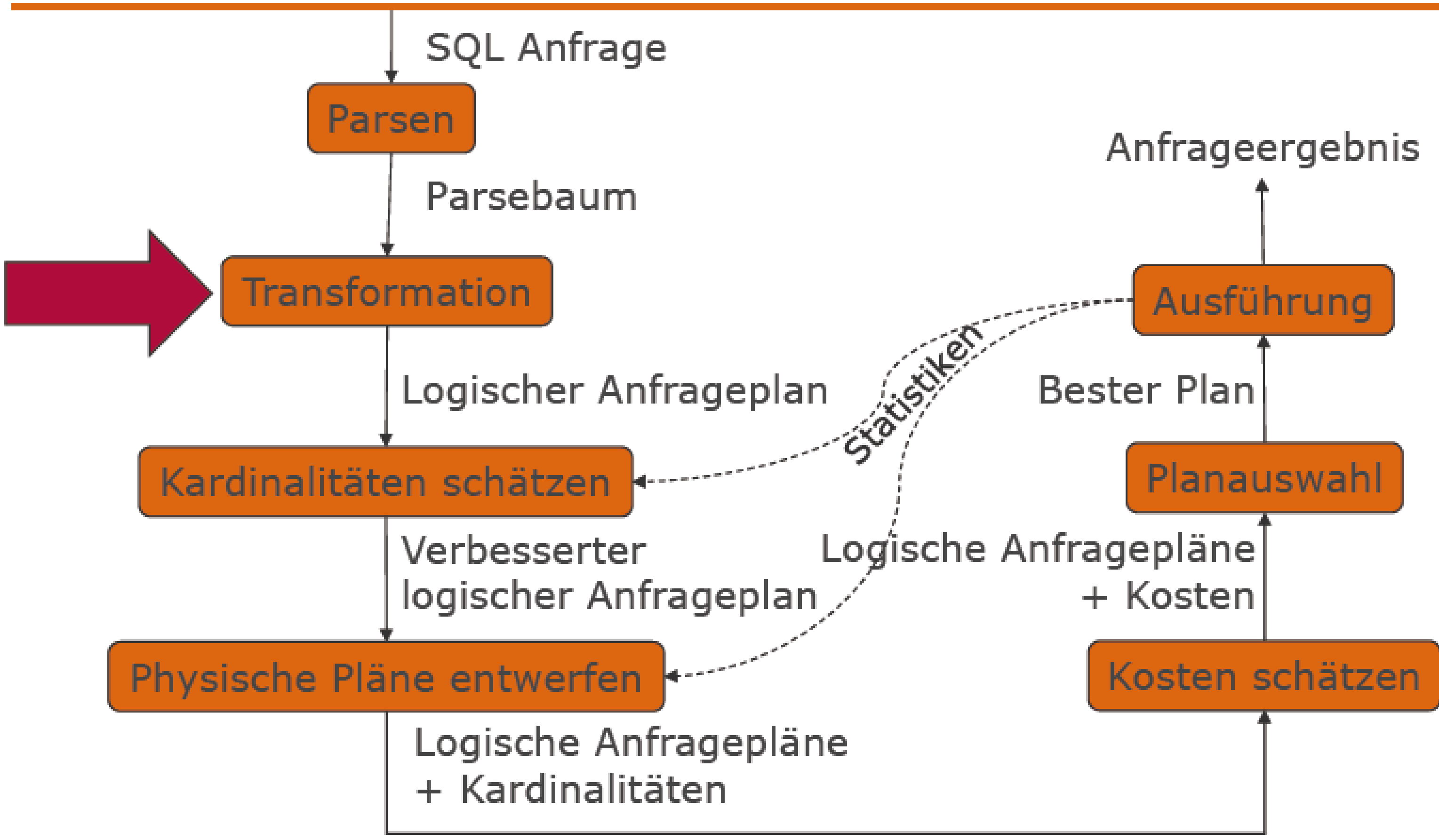
- Algebraische Transformation

- Kardinalitätsschätzung



- Join-Reihenfolge

Recap:
Algebraische Transformation
Ablauf der Anfragebearbeitung



Anfragebearbeitung – Transformationsregeln

- Transformation der internen Darstellung
  - Ohne Semantik zu verändern
  - Zur effizienteren Ausführung
    - Insbesondere: Kleine Zwischenergebnisse
- Äquivalente Ausdrücke
  - Zwei Ausdrücke der relationalen Algebra heißen äquivalent, falls
    - Gleiche Operanden (= Relationen)
    - Stets gleiche Antwortrelation
    - Stets?

Stets = Für jede mögliche Instanz der Datenbank

Recap: Algebraische Transformation

- $\times$ ist kommutativ und assoziativ
  - $R \times S = S \times R$
  - $(R \times S) \times T = R \times (S \times T)$
- $\cup$ ist kommutativ und assoziativ
  - $R \cup S = S \cup R$
  - $(R \cup S) \cup T = R \cup (S \cup T)$
- $\cap$ ist kommutativ und assoziativ
  - $R \cap S = S \cap R$
  - $(R \cap S) \cap T = R \cap (S \cap T)$
- $\bowtie$ ist kommutativ und assoziativ
  - $R \bowtie S = S \bowtie R$
  - $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

Gilt jeweils für Mengen und Multimengen

Ausdrücke können in beide Richtungen verwendet werden.

Recap: Algebraische Transformation

Weitere Regeln

Selektion

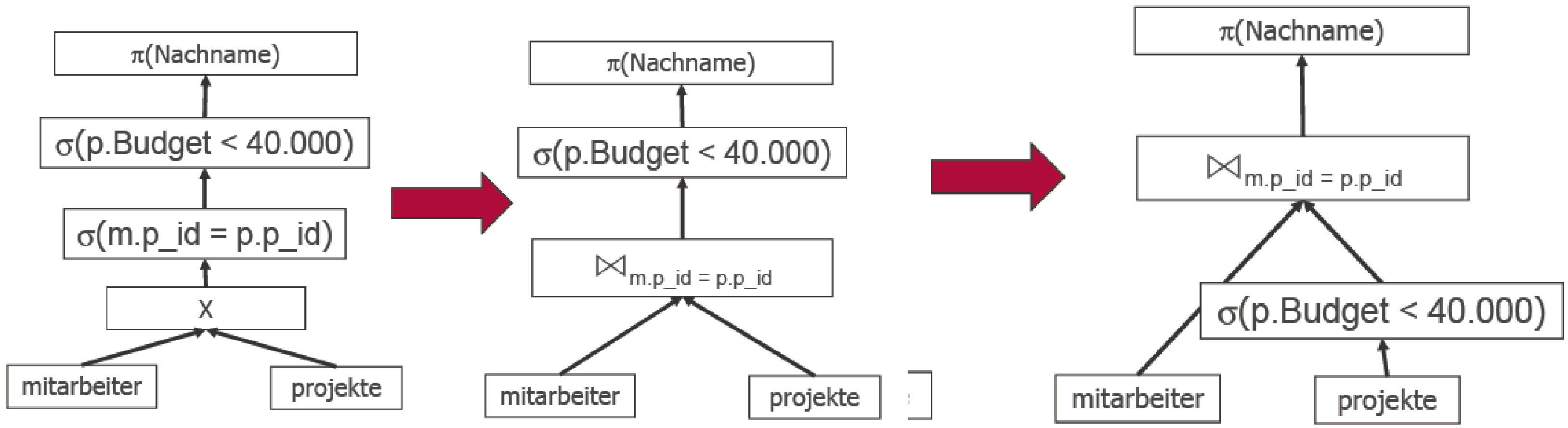
- $\sigma_{c1\ \text{AND}\ c2}(R) = \sigma_{c1}(\sigma_{c2}(R))$
- $\sigma_{c1\ \text{OR}\ c2}(R) = \sigma_{c1}(R) \cup \sigma_{c2}(R)$
  - Nicht bei Multimengen
- $\sigma_{c1}(\sigma_{c2}(R)) = \sigma_{c2}(\sigma_{c1}(R))$
- $\sigma_c(R\ \Phi\ S) \equiv (\sigma_c(R))\ \Phi\ (\sigma_c(S))$
  - $\Phi \in \{\cup, \cap, -, \bowtie\}$
- $\sigma_c(R\ \Phi\ S) \equiv (\sigma_c(R))\ \Phi\ S$
  - $\Phi \in \{\cup, \cap, -, \bowtie\}$
  - Falls sich c nur auf Attribute in R bezieht.

Projektion

- $\pi_L(R \bowtie S) = \pi_L(\pi_M(R) \bowtie \pi_N(S))$
- $\pi_L(R \bowtie_C S) = \pi_L(\pi_M(R) \bowtie_C \pi_N(S))$
- $\pi_L(R \times S) = \pi_L(\pi_M(R) \times \pi_N(S))$
- $\pi_L(\sigma_C(R)) = \pi_L(\sigma_C(\pi_M(R)))$

Recap: Algebraische Transformation

Anfragebearbeitung – Beispiel



Aufgabe 1:

Algebraische Transformation

- Gegeben: R(a,b,c) und S(c,d,e)
- Gesucht: Kostengünstigste Anfragepläne für folgende Anfragen

d.h. möglichst kleine Zwischenergebnisse, also Selektionen und Projektionen so früh wie möglich

a. $\sigma_{b=3 \wedge e=4 \wedge c>10} (R \bowtie S)$

$$\sigma_{c>10 \wedge b=3}(R) \bowtie \sigma_{c>10 \wedge e=4}(S)$$

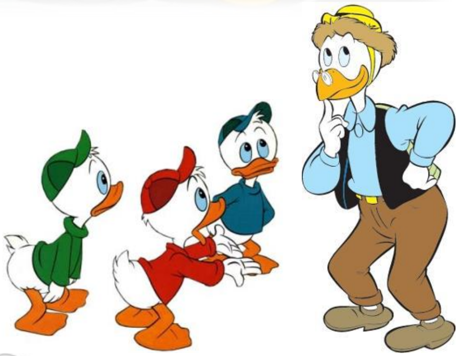
b. $\pi_{a,d} (R \bowtie S)$

$$\pi_{a,d} (\pi_{a,c} (R) \bowtie \pi_{c,d} (S))$$

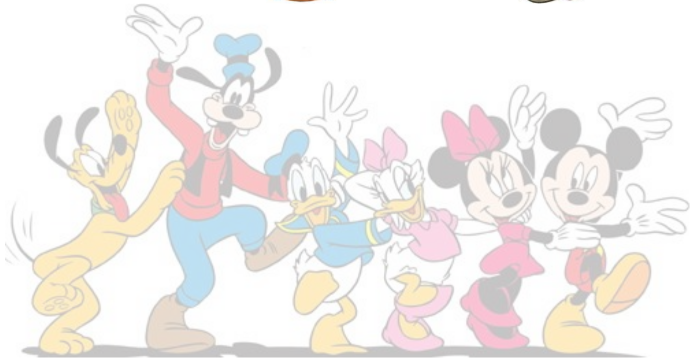
Einleitung:
Themen

- Algebraische Transformation

- Kardinalitätsschätzung



- Join-Reihenfolge



Kosten von Operationen - Zwischenergebnisse

- Wesentliches Kostenmerkmal: Anzahl der Tupel im Input
  - Insbesondere: Passt die Relation in den Hauptspeicher?
  - Selektion, Projektion, Sortierung, Join

- Output ist Input des nächsten Operators.

- Deshalb: Ein Kostenmodel schätzt u.a. für jede Operation die Anzahl der Ausgabetupel.
  - „Selektivität" in Bezug auf Inputgröße
  - #Ausgabetupel = #Eingabetupel x Selektivität
  - Auch „Selektivitätsfaktor" (*selectivity factor, sf*)

Kostenschätzung – Selektion

- Anzahl Tupel sinkt, Tupelgröße bleibt
- $S = \sigma_{A=c}(R)$
  - $T(S) = T(R)/V(R,A)$
    - Reminder: $V(R,A)$ = Anzahl der *distinct* Werte in Spalte A
    - D.h. selectivity factor ist $1/ V(R,A)$
  - Annahme: Werte sind gleichverteilt
  - Annahme: *c* ist einer dieser Werte
  - Bessere Abschätzung mittels Histogramme
- $S = \sigma_{A<c}(R)$
  - Erste Abschätzung: $T(S) = T(R) /2$
  - Typischer: $T(S) = T(R) /3$
- $S = \sigma_{A\neq c}(R)$
  - Erste Abschätzung: $T(S) = T(R)$
  - Etwas genauer: $T(S) = T(R)\cdot(V(R,A) - 1) / V(R,A)$
- Bei Konjunktionen mehrerer Selektionsbedingungen: Multiplikation der Selektivitätsfaktoren
  - Annahme: Unabhängigkeit der Bedingungen

Kostenschätzung – Selektion mit Disjunktion

- $S = \sigma_{C1 \text{ OR } C2}(R)$
- Idee 1: Summe der Ergebniskardinalitäten
  - Annahme: Kein Tupel erfüllt beide Bedingungen
  - Problem: T(S) > T(R)
- Idee 2: MIN[T(R), Summe der Ergebniskardinalitäten]
- Idee 3: Wahrscheinlichkeitstheorie
  - Annahme: C1 und C2 sind unabhängig
  - $T(R) = n$ und $T(\sigma_{C1}(R)) = m_1$ und $T(\sigma_{C2}(R)) = m_2$
  - => $T(S) = n(1-(1-\frac{m_1}{n})(1-\frac{m_2}{n}))$

Anteil Tupel, die nicht C1 entsprechen

Anteil Tupel, die nicht C2 entsprechen

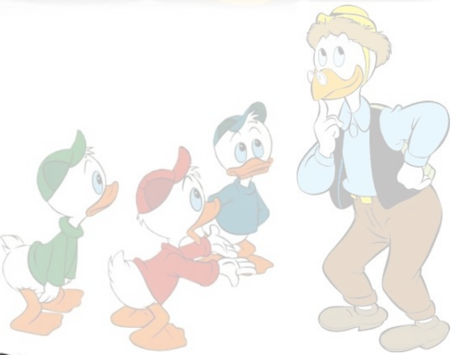
Kardinalitätsschätzung

- Gegeben: R(a,b,c,d) und S(d,e)
  T(R)=100; V(R,a)=100; V(R,b)=10; V(R,c)=1; V(R,d)=50
  T(S)=500; V(S,d)=30; V(S,e)=100
- Gesucht: Geschätzte Ergebniskardinalität für folgende Anfragen
  a. $\sigma_{b=25}$ (R) — T(R)/V(R,b)= 10
  b. $\sigma_{c=30}$ (R) — T(R)/V(R,c) = 100
  c. $\sigma_{b=25 \wedge c=30}$ (R) — T(R)/(V(R,b) · V(R,c)) = 10
  d. $\sigma_{b>25}$ (R) — T(R)/3 ≈ 33
  e. $\sigma_{a>30 \wedge b=10}$ (R) — T(R)/(3 · V(R,b)) ≈ 3
  f. $\sigma_{b>25 \wedge b=11}$ (R) — 0 (widersprüchliche Selektion)
  g. $\sigma_{b=25 \vee d=13}$ (R) — T(R)/V(R,b) + T(R)/V(R,d) - T(R)/(V(R,b) · V(R,d)) = 11,8 ≈ 12
  h. R ⋈ S — T(R) · T(S) / max[V(R,d),V(S,d)] = 1000

Einleitung:
Themen

- Algebraische Transformation

- Kardinalitätsschätzung



- Join-Reihenfolge

Kostenschätzung – Join

- Hier nur natural join
  - Equijoin analog
  - Thetajoin: „<", „>", usw: Wie zuvor schätzen. Z.B. 1/3 T(R) · 1/3 T(S)
- $R(X,Y) \bowtie S(Y,Z)$
  - Vereinfachende Annahme: Y ist nur <u>ein</u> Attribut.
- Problem: Beziehung zwischen R.Y und S.Y
  - Disjunkte Werte: $T(R \bowtie S) = 0$
  - Fremdschlüsselbeziehung (Schlüssel in S): $T(R \bowtie S) = T(R)$
  - Fast alles gleiche Werte: $T(R \bowtie S) = T(R) \cdot T(S)$
- Vereinfachende Annahmen
  - *Containment of Value Sets*: Werte eines Attributs, das in mehreren Relationen auftaucht, werden vom Beginn einer festen Liste gewählt.
    - Falls $V(R,Y) \leq V(S,Y)$ => Jeder Y-Wert in R taucht in S auf
    - Realistisch? Wann?
  - *Preservation of Value Sets*: Anzahl der distinct-Werte eines nicht-Joinattributs bleiben erhalten.
    - $V(R \bowtie S, X) = V(R, X)$
    - Realistisch? Wann?

Kostenschätzung – Join

- Sei $V(R,Y) \leq V(S,Y)$
  - => Jedes Tupel aus R hat eine $1/V(S,Y)$ Chance, mit einem gegebenen S-Tupel zu joinen
  - => (da T(S) S-Tupel): Ein Tupel aus R hat $T(S) \cdot 1/V(S,Y)$ Joinpartner in S
  - => (da T(R) R-Tupel): $T(R \bowtie S) = T(R) \cdot T(S) / V(S,Y)$
- Falls $V(R,Y) \geq V(S,Y)$: $T(R \bowtie S) = T(R) \cdot T(S) / V(R,Y)$
- Allgemein: $T(R \bowtie S) = T(R) \cdot T(S) / \max[V(R,Y),V(S,Y)]$

Hasso Plattner Institut

Kostenschätzung – Join

- $R(X,Y) \bowtie S(Y,Z)$
  - Y enthält mehr als ein Attribut
  - Schreibweise hier: $R(X,Y1, Y2) \bowtie S(Y1, Y2, Z)$

- $T(R \bowtie S) = T(R) \cdot T(S)\ /\ (\ \max[V(R,Y1),V(S,Y1)] \cdot \max[V(R,Y2),V(S,Y2)]\ )$

- Allgemein
  - Ergebniskardinalität von $R \bowtie S$ entspricht dem Produkt der Kardinalitäten von R und S, dividiert durch das Produkt des jeweils größeren von V(R,Y) und V(S,Y) **für jedes** Join-Attribut Y.

Aufgabe 3:

Join-Reihenfolge

- Gegeben: Folgende Relationen und deren Statistiken

$W(A,B)$	$X(B,C)$	$Y(C,D)$	$Z(D,A)$
$T(W) = 100$	$T(X) = 200$	$T(Y) = 300$	$T(Z) = 400$
$V(W,A) = 20$			$V(Z,A) = 100$
$V(W,B) = 60$	$V(X,B) = 50$		
	$V(X,C) = 100$	$V(Y,C) = 50$	
		$V(Y,D) = 50$	$V(Z,D) = 40$

- Gesucht: Optimale Join-Reihenfolge für W ⋈ X ⋈ Y ⋈ Z

- Bestimme die Join-Reihenfolge als Left Deep Tree. Nutze dazu Dynamische Programmierung und gib die Tabellen aller Zwischenschritte an.

Aufgabe 3:
Join-Reihenfolge

$W(A,B)$	$X(B,C)$	$Y(C,D)$	$Z(D,A)$
$T(W) = 100$	$T(X) = 200$	$T(Y) = 300$	$T(Z) = 400$
$V(W,A) = 20$			$V(Z,A) = 100$
$V(W,B) = 60$	$V(X,B) = 50$		
	$V(X,C) = 100$	$V(Y,C) = 50$	
		$V(Y,D) = 50$	$V(Z,D) = 40$

	{W}	{X}	{Y}	{Z}
Kardinalität	100	200	300	400
Kosten	0	0	0	0
Opt. Plan	W	X	Y	Z

Kleinste Relation links

	{[illegible]}	{W,Y}	{W,Z}	{X,Y}	{X,Z}	{Y,Z}
Kard.	10[illegible]00 /6[illegible] = 33[illegible]33	100·300 = 30000	100·400 /100 = 400	200·300 /100 = 600	200·400 = 80000	300·400 /50 = 2400
Kosten	0	0	0	0	0	0
Opt. Plan	W⋈X	W⋈Y	W⋈Z	X⋈Y	X⋈Z	Y⋈Z

Aufgabe 3:
Join-Reihenfolge

$W(A,B)$	$X(B,C)$	$Y(C,D)$	$Z(D,A)$
$T(W) = 100$	$T(X) = 200$	$T(Y) = 300$	$T(Z) = 400$
$V(W,A) = 20$			$V(Z,A) = 100$
$V(W,B) = 60$	$V(X,B) = 50$		
	$V(X,C) = 100$	$V(Y,C) = 50$	
		$V(Y,D) = 50$	$V(Z,D) = 40$

	{W,X}	{W,Y}	{W,Z}	{X,Y}	{X,Z}	{Y,Z}
Kard.	100·200 /60 = 333,33	~~100·300 = 30[illegible]~~	100·400 /100 = 400	200·300 /100 = 600	~~200·400 = 80[illegible]~~	300·400 /50 = 2400
Kosten	0	0	0	0	0	0
Plan	[illegible]	[illegible]	W⋈Z	X⋈Y	X⋈Z	Y⋈Z

Kardinalität ist für alle möglichen Joinreihenfolgen gleich!

Kosten: (Kardinalität + [illegible]) des Zwischenergebniss[illegible]

	[illegible],Y}	{W,X,Z}	{W,Y,Z}	{X,Y,Z}
Kardinalität	[illegible]33·300 /100 = 1000	333,33·400 /100 = 1333,33	400·300 /50 = 2400	600·400 /50 = 4800
Kosten	333,33	333,33	400	600
Opt. Plan	(W⋈X)⋈Y	(W⋈X)⋈Z	(W⋈Z)⋈Y	(X⋈Y)⋈Z

Aufgabe 3:
Join-Reihenfolge

$W(A,B)$	$X(B,C)$	$Y(C,D)$	$Z(D,A)$
$T(W) = 100$	$T(X) = 200$	$T(Y) = 300$	$T(Z) = 400$
$V(W,A) = 20$			$V(Z,A) = 100$
$V(W,B) = 60$	$V(X,B) = 50$		
	$V(X,C) = 100$	$V(Y,C) = 50$	
		$V(Y,D) = 50$	$V(Z,D) = 40$

	{W,X}	{W,Y}	{W,Z}	{X,Y}	{X,Z}	{Y,Z}
Kard.	100·200 /60 = 333,33	100·300 = 30000	100·400 /100 = 400	200·300 /100 = 600	200·400 = 80000	300·400 /50 = 2400
Kosten	0	0	0	0	0	0
Opt. Plan	W⋈X	W⋈Y	W⋈Z	X⋈Y	X⋈Z	Y⋈Z

	{W,X,Y}	{W,X,Z}	{W,Y,Z}	{X,Y,Z}
Kardinalität	333,33·300 /100 = 1000	333,33·400 /100 = 1333,33	400·300 /50 = 2400	600·400 /50 = 4800
Kosten	333,33	333,33	400	600
Opt. Plan	(W⋈X)⋈Y	(W⋈X)⋈Z	(W⋈Z)⋈Y	(X⋈Y)⋈Z

Aufgabe 3:
Join-Reihenfolge

$W(A,B)$	$X(B,C)$	$Y(C,D)$	$Z(D,A)$
$T(W) = 100$	$T(X) = 200$	$T(Y) = 300$	$T(Z) = 400$
$V(W,A) = 20$			$V(Z,A) = 100$
$V(W,B) = 60$	$V(X,B) = 50$		
	$V(X,C) = 100$	$V(Y,C) = 50$	
		$V(Y,D) = 50$	$V(Z,D) = 40$

	{W,X,Y}	{W,X,Z}	{W,Y,Z}	{X,Y,Z}
Kardinalität	333,33·300 /100 = 1000	333,33·400 /100 = 1333,33	400·300 /50 = 2400	600·400 /50 = 4800
Kosten	333,33	333,33	400	600
Opt. Plan	(W⋈X)⋈Y	(W⋈X)⋈Z	(W⋈Z)⋈Y	(X⋈Y)⋈Z

	{W,X,Y,Z}	{W,X,Y,Z}	{W,X,Y,Z}	{W,X,Y,Z}
Kard.	1000·400 /(100·50) = 80	1333,33·300 /(100·50) = 80	2400·200 /(100·60) = 80	4800·100 /(100·60) = 80
Kosten	1333,33	1666,66	2800	5400
Plan	((W⋈X)⋈Y)⋈Z	((W⋈X)⋈Z)⋈Y	((W⋈Z)⋈Y)⋈X	((X⋈Y)⋈Z) ⋈W

Sonderfall:

Mehrfach-Join über ein Attribut

X(a,b)	**Y(a,b)**	**Z(a,b)**
T(X)=100	T(Y)=200	T(Z)=300
V(X,a)=10	V(Y,a)=20	V(Z,a)=30
V(X,b)=3	V(Y,b)=2	V(Z,b)=1

$X \bowtie Y = 100 \cdot 200 / (20 \cdot 3) = 333{,}33$

$(X \bowtie Y) \bowtie Z = 333{,}33 \cdot 300 / (30 \cdot \mathbf{2}) = 1666{,}66$

- *Preservation of Value Sets* gilt nicht, wenn das Attribut ein Join-Attribut war!
- Wegen *Containment of Value Sets* wissen wir aber, dass nach dem Join $V(X \bowtie Y,a) = 10$ und $V(X \bowtie Y,b) = 2$ sein müssen.

Allgemein:

$X \bowtie Y \bowtie Z = T(X) \cdot T(Y) \cdot T(Z) / [V(Y,a) \cdot V(Z,a)] \cdot [V(X,b) \cdot V(Y,b)]$

Für jedes Join-Attribut teile durch alle Kardinalitäten außer der Kleinsten!