TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

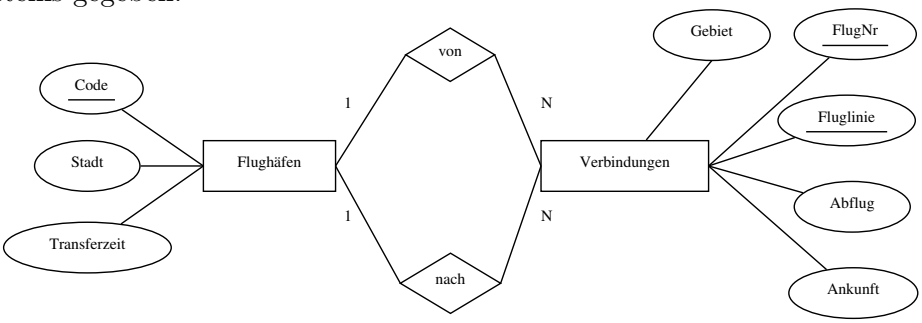
Übung zur Vorlesung *Grundlagen: Datenbanken* im WS23/24

Christoph Anneser, Michael Jungmair, Stefan Lehner, Moritz Sichert, Lukas Vogel
(gdb@in.tum.de)
https://db.in.tum.de/teaching/ws2324/grundlagen/

Blatt Nr. 12

Hausaufgabe 1

Betrachten Sie die Anfrage *„Finde alle Flüge von New-York nach Sydney mit einmaligem Umsteigen“*. Dazu sei das nachfolgende (vereinfachte) ER-Diagramm eines Fluginformationssystems gegeben:



a) Geben Sie eine SQL-Query für die oben genannte Anfrage an.

b) Führen Sie die kanonische Übersetzung des SQL-Statements in die relationale Algebra durch.

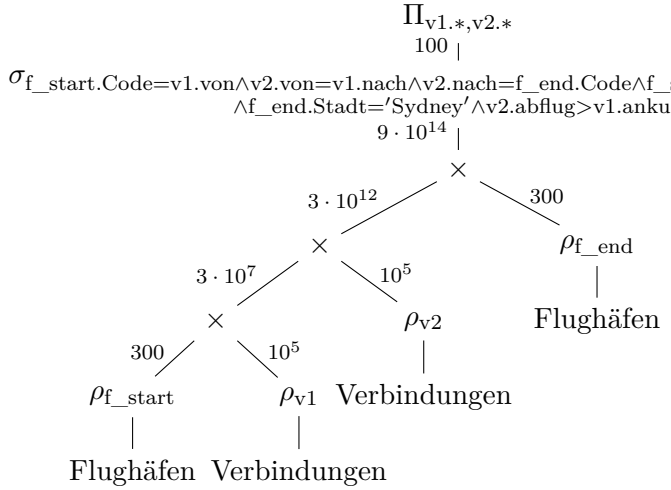
c) Schätzen Sie die Relationsgrößen sinnvoll ab (z.B. so wie in den Beispielen der Vorlesung) und transformieren Sie den kanonischen Operatorbaum aus Teilaufgabe b) zur optimalen Form. Wie haben sich die Kosten dabei geändert? (Kosten = Anzahl der Zwischenergebnistupel)

Lösung:

a) SQL-Anfrage:

```
SELECT DISTINCT v1.*, v2.*
FROM Flughäfen f_start, Verbindungen v1, Verbindungen v2, Flughäfen f_end
WHERE f_start.Stadt = "New York"
  AND f_end.Stadt = "Sydney"
  AND v2.von = v1.nach
  AND v2.nach = f_end.Code
  AND f_start.Code = v1.von
  AND v2.abflug > v1.ankunft
```

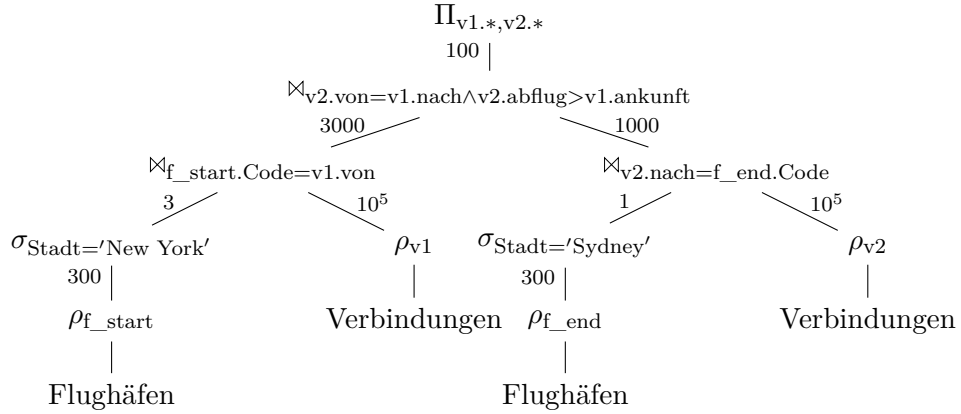
b) **Kanonische Übersetzung:**
Wir gehen davon aus, dass es weltweit 300 (relevante) Flughäfen gibt und 100.000 Verbindungen zwischen Flughäfen. Weiterhin nehmen wir an, dass es 100 Möglichkeiten gibt, über einen Zwischenstop von New York nach Sydney zu fliegen. Andere Werte sind natürlich auch in Ordnung. Hier reicht es aus, grob zu schätzen.



Diese Anfrage kostet über 900 Billionen Tupel. Wenn eine hypotetische CPU mit 3 GHz ein Tupel pro Takt verarbeiten könnte, dann liefe die Anfrage über 3 Tage und 11 Stunden.

c) **Optimaler Ausführungsplan:**

Wir gehen hier zusätzlich davon aus, dass es drei Flughäfen in New York gibt und einen in Sydney. Weiterhin gehen wir davon aus, dass es von jedem New Yorker Flughafen sowie zu dem Flughafen in Sydney jeweils 1000 Verbindungen gibt.



Die optimierte Anfrage kostet nur ca. 205000 Tupel. Unsere hypotetische 3 GHz-CPU würde hier nur noch 68 Mikrosekunden benötigen.

Hausaufgabe 2

Für einen Join-Baum T sei folgende Kostenfunktion gegeben

$$C_{out}(T) = \begin{cases} 0 & \text{falls } T \text{ eine Basisrelation } R_i \text{ ist} \\ |T| + C_{out}(T_1) + C_{out}(T_2) & \text{falls } T = T_1 \bowtie T_2 \end{cases}$$

Die Kardinalität sei dabei

$$|T| = \begin{cases} |R_i| & \text{falls } T \text{ eine Basisrelation } R_i \text{ ist} \\ (\prod_{R_i \in T_1, R_j \in T_2} f_{i,j}) |T_1||T_2| & \text{falls } T = T_1 \bowtie T_2 \end{cases}$$

Sei $p_{i,j}$ das Joinprädikat zwischen R_i und R_j, dann sei

$$f_{i,j} = \frac{|R_i \bowtie_{p_{i,j}} R_j|}{|R_i \times R_j|}$$

und die Kardinalität eines Join-Resultats ist $|R_i \bowtie_{p_{i,j}} R_j| = f_{i,j}|R_i||R_j|$.

Gegeben sei eine Anfrage über die Relationen R_1, R_2, R_3 und R_4 mit $|R_1| = 10, |R_2| = 20, |R_3| = 20, |R_4| = 10$. Die Selektivitäten der Joins seien $f_{1,2} = 0.01, f_{2,3} = 0.5, f_{3,4} = 0.01$, alle nicht gegebenen Selektivitäten sind offensichtlich 1 (Warum?). Berechnen Sie den optimalen (niedrigste Kosten) Join-Tree. Als Vereinfachung reicht es, wenn Sie nur Joins mit Prädikat und keine Kreuzprodukte betrachten.

Lösung:

Es ist kein Algorithmus angegeben. Aufgrund der geringen Anzahl von Relationen ist es möglich, die Kosten aller möglichen Join-Bäume zu berechnen und den kostengünstigsten auszuwählen (Bruteforce).

Zunächst gilt es zu überlegen, für welche Join-Bäume die Kosten tatsächlich zu berechnen sind.

Left-Deep:

$$((R_1 \bowtie R_2) \bowtie R_3) \bowtie R_4 \tag{1}$$
$$((R_4 \bowtie R_3) \bowtie R_2) \bowtie R_1 \tag{2}$$
$$((R_3 \bowtie R_2) \bowtie R_1) \bowtie R_4 \tag{3}$$
$$((R_3 \bowtie R_2) \bowtie R_4) \bowtie R_1 \tag{4}$$

Bushy:

$$(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4) \tag{5}$$

Alle anderen Left Deep oder Bushy Trees enthalten Kreuzprodukte oder sind im Bezug auf die Kosten äquivalent. Ersteres entsteht, wenn Relationen in einer Reihenfolge gejoint werden, in der bei einem der Joins kein Prädikt möglich ist, beispielsweise ist dies für den Left-Deep Tree

$$((R_1 \bowtie R_2) \times R_4) \bowtie R_3$$

der Fall. Im Bezug auf die Kosten bei der gegebenen Kostenfunktion äquivalent sind Join-Trees, bei denen die Kinder eines Join Operators vertauscht wurden, etwa

$$(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)$$

und

$$(R_3 \bowtie R_4) \bowtie (R_1 \bowtie R_2).$$

Im Beispiel müssen lediglich die Kosten für die Join-Reihenfolgen 1, 3 und 5 berechnet werden. Dies liegt am Aufbau der Kostenfunktion sowie den symmetrischen Größen der Relationen sowie ihrer Join Selektivitäten.

Die Berechnung von $C_{out}((R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4))$ sei hier exemplarisch in epischer Breite ausgeführt (Machen Sie es selber; Sie erkennen äußerst schnell ein Muster und müssen keine derartigen Formel-Konvolute schreiben):

$$
\begin{aligned}
& C_{out}((R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)) \\
= \; & |(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)| + C_{out}(R_1 \bowtie R_2) + C_{out}(R_3 \bowtie R_4) \\
= \; & |(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)| + |R_1 \bowtie R_2| + C_{out}(R_1) + C_{out}(R_2) + |R_3 \bowtie R_4| + C_{out}(R_3) + C_{out}(R_4) \\
= \; & |(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)| + |R_1 \bowtie R_2| + |R_3 \bowtie R_4| \\
= \; & f_{1,3} \cdot f_{1,4} \cdot f_{2,3} \cdot f_{2,4} \cdot |R_1 \bowtie R_2| \cdot |R_3 \bowtie R_4| + |R_1 \bowtie R_2| + |R_3 \bowtie R_4| \\
= \; & 0.5 \cdot (0.01 \cdot 10 \cdot 20) \cdot (0.01 \cdot 20 \cdot 10) + (0.01 \cdot 10 \cdot 20) + (0.01 \cdot 20 \cdot 10) \\
= \; & 2 + 2 + 2 \\
= \; & 6
\end{aligned}
$$

Die Ergebnisse der anderen Relevanten Join-Reihenfolgen sind:

$((R_1 \bowtie R_2) \bowtie R_3) \bowtie R_4$	24
$((R_3 \bowtie R_2) \bowtie R_1) \bowtie R_4$	222
$(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4)$	6

Der Bushy-Tree 5 ist also der optimale Join-Tree.

Hausaufgabe 3

Gegeben sei die Anfrage:

```
select *
  from R, S, T
 where R.A = S.A and S.B = T.B and T.C = R.A
```

Des Weiteren soll gelten:

- `S.A` und `T.C` seien Fremdschlüssel auf `R`
- `S.B` sei Fremdschlüssel auf `T`
- `R.A`, `T.B` seien Primärschlüssel von `R` respektive `T`
- Ihre Query-Engine unterstützt nur Nested-Loop-Joins
- Kardinalitäten: $|R| = 100, |S| = 1000, |T| = 10$
- Es gibt keine Indexe

a) Für Equijoins zwischen R_i mit Fremdschlüssel auf den Primärschlüssel in R_j gilt die Abschätzung:

$$f_{i,j} = \frac{1}{|R_j|}$$

Warum?

b) Bestimmen Sie, wie in der Vorlesung gezeigt, den optimalen Ausführungsplan als Baum mit Kosten-/Kardinalitätsabschätzungen mit Hilfe von Dynamischem Programmieren. Verwenden Sie die Kostenfunktion C_{out}.

Lösung:

a) Da das Datenbanksystem die referentielle Integrität sicherstellt, referenziert jeder Fremdschlüssel mindestens ein existierendes Tupel. Jedes Tupel aus R_i hat also mindestens einen Join-Partner in R_j. Es gibt also mindestens so viele Joinpaare wie R_i Tupel hat.

Gleichzeitig ist das Zielattribut in R_j Primärschlüssel, also ist jeder Wert einzigartig. Für jeden Wert in R_i kann es also höchstens einen Joinpartner in R_j geben.

Damit gibt es genau $|R_i|$ viele Joinpaare. Die Selektivität ist damit genau $\frac{1}{|R_j|}$, da $\frac{1}{|R_j|} \cdot |R_j| \cdot |R_i| = |R_i|$.

b) Da alle Relationen über Fremdschlüssel verbunden sind, gelten (vereinfachend) folgende Kardinalitäten:

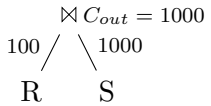
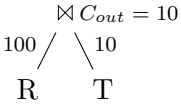
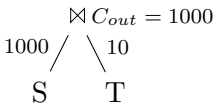
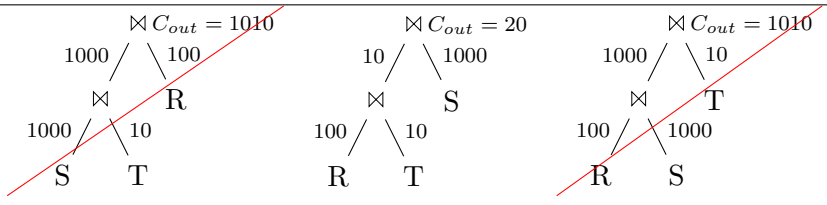
$$|R \bowtie S| = |S \bowtie R| = \frac{1}{|R|} \cdot |R| \cdot |S| = |S| = 1000$$

$$|R \bowtie T| = |T \bowtie R| = \frac{1}{|R|} \cdot |R| \cdot |T| = |T| = 10$$

$$|S \bowtie T| = |T \bowtie S| = \frac{1}{|T|} \cdot |S| \cdot |T| = |S| = 1000$$

$$|R \bowtie S \bowtie T| = \frac{1}{|R|} \cdot \frac{1}{|R|} \cdot \frac{1}{|T|} \cdot |R| \cdot |S| \cdot |T| = \frac{|S|}{|R|} = 10$$

Für die Berechnung der Kosten ergibt sich dann folgende DP-Tabelle:

DP-Tabelle		
Index	Pläne	Kosten
R	R	0
S	S	0
T	T	0
R,S		1000
R,T		10
S,T		1000
R,S,T		20

Hausaufgabe 4

Geben Sie die kanonische Übersetzung der folgenden Anfrage in die relationale Algebra an. Verwenden Sie zur Darstellung des relationalen Algebraausdrucks die Baumdarstellung.

```
select v.VorlNr, v.Titel, p.Name, count(h.MatrNr) as hoerer
from
```

```
    Vorlesungen v left outer join
    hoeren h on (v.VorlNr = h.VorlNr),
    Professoren p
where
    v.gelesenVon = p.PersNr
group by v.VorlNr, v.Titel, p.Name
having count(h.MatrNr) > 3
```

Lösung: