9. Übung zur Vorlesung "Datenbanken" im Sommersemester 2007
– mit Musterlösungen –

Prof. Dr. Gerd Stumme, Dr. Andreas Hotho, Dipl.-Inform. Christoph Schmitz

2. Juli 2007

Aufgabe 1

1. An welchem Punkt der Anfragebearbeitung findet die Anfrageoptimierung statt?

   Anfragen werden nach dem Parsen und Erstellen eines Anfrageplanes, und vor der Ausführung des Anfrageplanes optimiert.

2. Warum ist es nicht wünschenswert, die Anfrageoptimierung explizit durch Benutzereingriffe zu steuern, etwa durch Umformulierung von Anfragen oder Vorgabe von Join-Reihenfolgen?

   Die Grundidee von SQL (und der relationalen Algebra) ist die einer deklarativen Anfragesprache, d. h. der Programmierer gibt vor, welches Ergebnis gewünscht ist, aber nicht, wie dieses zu berechnen ist.

   Die Optimierung von Anfrageplänen zur effizienten Ausführung ist daher Aufgabe des Datenbanksystems, nicht des Anfragestellers.

3. Warum ist es manchmal doch notwendig, daß der Benutzer in die Anfrageoptimierung eingreift?

   Bei sehr kostenintensiven Anfragen, z. B. Joins über viele oder große Tabellen, muß oft auf Hintergrundwissen zurückgegriffen werden, das die Datenbank nicht aus Statistiken, Constraints o. ä. ableiten kann.

   In diesem Fall kann der Benutzer durch Vorgabe von Informationen über die Anfrage, durch Erzwingen von Join-Reihenfolgen (z. B. durch Berechnung geeigneter Zwischenergebnisse) oder durch Auslagern von Berechnungen in die Anwendungslogik eingreifen.

Aufgabe 2

Ein Versandhändler hat

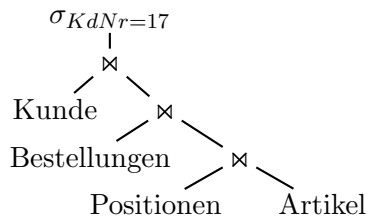
- 100000 Kunden, von denen jeder
- 10 Bestellungen getätigt hat, die jeweils
- 10 Positionen umfassen, jede Position referenziert einen der
- 100000 Artikel im Sortiment.

Kunde 17 klickt im Web auf den Link "Übersicht über meine Bestellungen", woraufhin er seine Bestellungen mit allen Positionen aufgelistet bekommt.

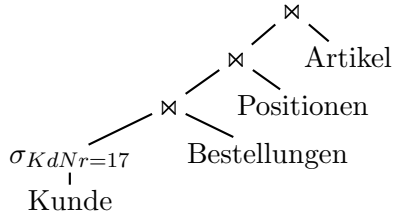
Die Anfrage dazu lautet: $\sigma_{KdNr=17}(Kunden \bowtie Bestellungen \bowtie Positionen \bowtie Artikel)$

Schätzen Sie die Laufzeit der beiden folgenden QEPs ab, falls Nested-Loop-Joins oder Sort-Merge-Joins zum Einsatz kommen. Gehen Sie davon aus, daß die Kosten für $R \bowtie S$ für Nested-Loop-Joins $|R| \cdot |S|$ betragen. Im Falle von Sort-Merge-Joins seien sortierte Indizes vorhanden, so daß die Kosten dort $|R| + |S|$ sind.

QEP1:



QEP2:



Da es sich hier um Abschätzungen handelt, wird jeweils nur die Größenordnung angegeben – in diesem Sinne ist hier also $10^7 + 10^5 = 10^7$.

Für Nested-Loop-Joins:

QEP1:

	Relation 1	#Tupel	Relation 2	#Tupel	#Iter.	#Ergebnistupel
Join 1	Positionen	10^7	Artikel	10^5	10^{12}	10^7
Join 2	Bestellungen	10^6	Join 1	10^7	10^{13}	10^7
Join 3	Kunde	10^5	Join 2	10^7	10^{12}	10^7

Insgesamt: ca. 10^{13} Iterationen.

QEP2:

	Relation 1	#Tupel	Relation 2	#Tupel	#Iter.	#Ergebnistupel
Join 1	$\sigma(\dots)$	1	Bestellungen	10^6	10^6	10
Join 2	Join 1	10	Positionen	10^7	10^8	100
Join 3	Join 2	100	Artikel	10^5	10^7	100

Insgesamt: ca. 10^8 Iterationen. Es ergibt sich also eine Verbesserung um Faktor 10000 (z. B.: 10 s gegenüber 27 Stunden) gegenüber QEP1.

Für Sort-Merge-Joins:

QEP1:

	Relation 1	#Tupel	Relation 2	#Tupel	#Iter.	#Ergebnistupel
Join 1	Positionen	10^7	Artikel	10^5	10^7	10^7
Join 2	Bestellungen	10^6	Join 1	10^7	10^7	10^7
Join 3	Kunde	10^5	Join 2	10^7	10^7	10^7

Insgesamt: ca. $3 \cdot 10^7$ Iterationen.

QEP2:

	Relation 1	#Tupel	Relation 2	#Tupel	#Iter.	#Ergebnistupel
Join 1	$\sigma(\dots)$	1	Bestellungen	10^6	10^6	10
Join 2	Join 1	10	Positionen	10^7	10^7	100
Join 3	Join 2	100	Artikel	10^5	10^5	100

Insgesamt: ca. 10^7 Iterationen. Durch die Verwendung von Indizes und Sort-Merge-Joins liegt hier QEP1 also wesentlich dichter an QEP2.

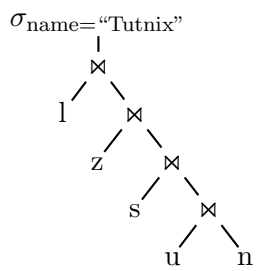
Aufgabe 3

Folgende Anfrage rechnet den Durchschnitt der Noten aus, die der Lehrer "Tutnix" vergeben hat:

```
select l.name, avg(noten)
from lehrer l,
     schueler s,
     unterricht u,
     notennumerisch n,
     zeugnis z
where l.name='Tutnix'
and   s.buchst = u.buchst
and   s.jahrgang = u.jahrgang
and   u.lehrername = l.name
and   n.notea = z.note
and   z.schuelername = s.name
group by l.name
```

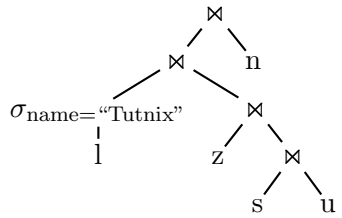
Dafür könnte ein ungünstiger Auswertungsplan (Query Execution Plan, QEP) wie folgt aussehen:

QEP1:



Die Datenbank DB2 hat dies zum folgenden QEP optimiert:

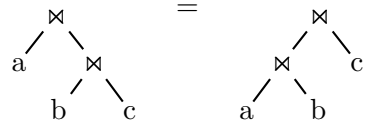
QEP2:



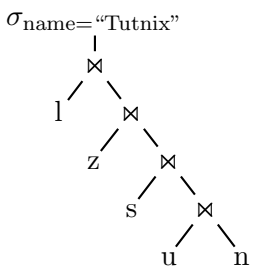
Bringen sie mit den äquivalenzerhaltenden Transformationsregeln aus der Vorlesung den Plan QEP1 in die Form QEP2.

Zur Erinnerung – die Assoziativität des Joins sieht wie folgt aus:

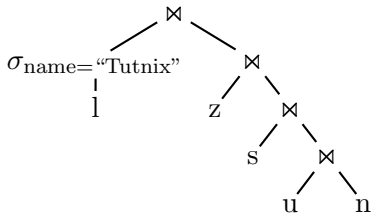
$a \bowtie (b \bowtie c)$ = $(a \bowtie b) \bowtie c$



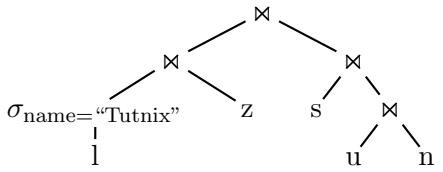
QEP 1 =



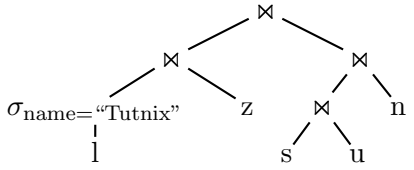
Selektionen nach unten →



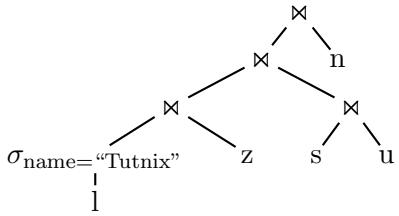
$\xrightarrow{\bowtie \text{ ass.}}$



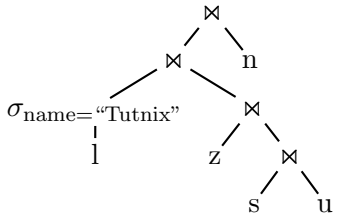
$\xrightarrow{\bowtie \text{ ass.}}$



$\xrightarrow{\bowtie \text{ ass.}}$



$\xrightarrow{\bowtie \text{ ass.}}$ = QEP 2



Aufgabe 4

1. Formen Sie die Anfrage

$$\pi_A((R \bowtie_{B=C} S) \bowtie_{D=E} T)$$

nach

$$\pi_A((\pi_E(T) \bowtie_{E=D} \pi_{ACD}(S)) \bowtie_{C=B} R)$$

um.

$$\begin{aligned}
& \pi_A((R \bowtie_{B=C} S) \bowtie_{D=E} T) && (1)\\
= (2)\quad & \pi_A(T \bowtie_{E=D} (S \bowtie_{C=B} R)) && (3)\\
= (4)\quad & \pi_A((T \bowtie_{E=D} S) \bowtie_{C=B} R) && (5)\\
= (6)\quad & \pi_A((\pi_E(T) \bowtie_{E=D} \pi_{ACD}(S)) \bowtie_{C=B} R) && (7)
\end{aligned}$$

Erklärung der Umformungsschritte:

(2) $\bowtie$ ist kommutativ

(4) $\bowtie$ ist assoziativ

(6) Einführung der Projektionen; dies darf nur dort geschehen, wo die entsprechenden Attribute vorhanden sind und keine benötigten Attribute entfernt werden.

2. Geben Sie an, welche Attribute die Relationen R, S und T mindestens haben müssen oder nicht haben dürfen, damit die Umformung korrekt ist.

   Ein Eintrag "(1)" bedeutet: dieses Attribut muß wegen Zeile (1) vorhanden sein. "¬(2)" bedeutet: darf wegen Zeile (2) nicht vorhanden sein.

	A	B	C	D	E
R	¬(6)	(1)			
S	(6)		(1)	(1)	
T	¬(6)				(1)

SQL-Übungsaufgabe

Formulieren Sie zwei Varianten der Anfrage: Welche Studenten sind über alle Vorlesungen geprüft worden? Stellen Sie den Allquantor einmal über eine Abzählung dar und einmal über eine doppelte Negation!

Lösung zur SQL-Aufgabe vom 18.6.

1. Welche Fachgebiete decken die Assistenten jedes Professors ab? (Name + Fachgebiets-Name gesucht)

```
select distinct p.name, a.fachgebiet
from professoren p, assistenten a
where p.persnr = a.boss
```

2. Wieviele Fachgebiete werden in jedem Lehrstuhl abgedeckt? (Name des Professors + Fachgebiets-Name gesucht)

```
select proffach.name, count(*) from (
  select distinct p.name, a.fachgebiet
  from professoren p, assistenten a
  where p.persnr = a.boss
) proffach
group by proffach.name
```

Vorsicht! Hier ist das innere `distinct` wichtig, damit nicht Fachgebiete doppelt gezählt werden!

3. Welche Professoren (Name gesucht) decken über ihre Assistenten mehr als drei Fachgebiete ab?

```
select proffach.name, count(*) from (
  select distinct p.name, a.fachgebiet
  from professoren p, assistenten a
  where p.persnr = a.boss
) proffach
group by proffach.name
having count(*) > 3
```