Lehrstuhl für Praktische Informatik III
Prof. Dr. Guido Moerkotte
Email: moer@db.informatik.uni-mannheim.de

Datenbanksysteme 1 — 9. Übungsblatt
Herbst-/Wintersemester 2016

Aufgabe 1

Zeigen Sie: Wenn R in dritter Normalform ist, dann ist R auch in zweiter Normalform.

Lösung

ZZ: $3NF \implies 2NF$

Notation:
- $\mathcal{R}$:= Relationenschema
- K := $\{k | k \subseteq \mathcal{R}, k \overset{\bullet}{\rightarrow} \mathcal{R}\}$ (Kandidatenschlüssel von R)
- SA := $\{A | \exists k \in K : A \in k\}$ (Menge der Schlüsselattribute in R)
- NSA := $\mathcal{R} \setminus SA$ (Menge der Nicht-Schlüsselattribute in R)

Proof. Die Aussage ist eine Implikation und daher äquivalent zu ihrer Kontraposition:

$$(3NF \implies 2NF) \iff (\neg 2NF \implies \neg 3NF)$$

Wir zeigen durch einen direkten Beweis, dass die Kontraposition gilt.

Sei also $\neg 2NF$ erfüllt.
Das bedeutet $\exists A \in NSA, \exists k \in K$:

$$k \overset{\bullet}{\not\rightarrow} A$$

Daraus folgt $\exists g \subset k$:

$$g \overset{\bullet}{\rightarrow} A$$

Es gilt also die funktionale Abhängigkeit $g \rightarrow A$.

Wir müssen zeigen, dass $\neg 3NF$ gilt. Also, dass eine funktionale Abhängigkeit existiert, die keine der drei Bedingungen aus der Definition von $3NF$ erfüllt.
Betrachen wir, die aus $\neg 2NF$ abgeleitete funktionale Abhängigkeit $g \rightarrow A$:
- $A \in g$ gilt nicht, da $A \in NSA, g \subset k \subseteq SA$ und $NSA \cap SA = \emptyset$
- $A \in SA$ gilt nicht, da $A \in NSA$ und $NSA \cap SA = \emptyset$
- $g \rightarrow \mathcal{R}$ gilt nicht, da $g \subset k$ und $k \overset{\bullet}{\rightarrow} \mathcal{R}$

Also gilt $\neg 3NF$.

Damit ist gezeigt, dass $\neg 2NF \implies \neg 3NF$ gilt und daher auch $3NF \implies 2NF$ gilt. □

Alternative

Proof. Beweis durch Widerspruch:
Angenommen $\mathcal{R}$ ist in $3NF$ und nicht in $2NF$.

Dass $\mathcal{R}$ nicht in $2NF$ ist, bedeutet $\exists A \in NSA, \exists k \in K$:

$$k \overset{\bullet}{\not\to} A$$

Daraus folgt $\exists g \subset k$:

$$g \overset{\bullet}{\to} A.$$

Es gilt also die funktionale Abhängigkeit $g \to A$.

Dass $\mathcal{R}$ in $3NF$ ist, bedeuetet, dass jede funktionale Abhäengigkeit eine der drei Bedingungen aus der Definition von $3NF$ erfüllen muss.
Betrachen wir, die aus $\neg 2NF$ abgeleitete funktionale Abhängigkeit $g \to A$:

$A \in g$	gilt nicht, da $A \in NSA, g \subset k \subseteq SA$ und $NSA \cap SA = \emptyset$
$A \in SA$	gilt nicht, da $A \in NSA$ und $NSA \cap SA = \emptyset$
$g \to \mathcal{R}$	gilt nicht, da $g \subset k$ und $k \overset{\bullet}{\to} \mathcal{R}$

Die Annahme, dass $\mathcal{R}$ in $3NF$ und nicht in $2NF$ ist, ist also falsch.
Das Gegenteil muss gelten

$$\neg(3NF \wedge \neg 2NF).$$

Durch umformen (De Morgan) erhalten wir

$$\neg 3NF \vee 2NF.$$

Das entspricht der Definition von Implikation:

$$3NF \implies 2NF.$$

□

Aufgabe 2

Gegeben sei das aus der Vorlesung bekannte Uni-Datenbankschema mit den folgenden Relationen:

- Professoren(PersNr, Name, Rang, Raum)
- Studenten(MatrNr, Name, Semester)
- Vorlesungen(VorlNr, Titel, SWS, gelesenVon)
- Assistenten(PersNr, Name, Fachgebiet, Boss)
- voraussetzten(Vorgänger, Nachfolger)
- hören(MatNr, VorlNr)

- prüfen(MatNr, VorlNr, PersNr, Note)

Welche der folgenden Umsetzungen einer Anfrage in die Algebra läuft (wahrscheinlich) länger? Begründen Sie Ihre Antwort kurz.

Aufgabe 2 a)

1. $\Pi_{Assistenten.name}(\sigma_{Professoren.name='Curie'}(Assistenten \bowtie_{boss=Professoren.persNr} Professoren))$
2. $\Pi_{Assistenten.name}(Assistenten \bowtie_{boss=Professoren.persNr}(\sigma_{Professoren.name='Curie'}(Professoren)))$

Lösung

Alternative 1 läuft vermutlich länger. Hier werden zuerst für alle Professoren die zugehörigen Assistenten ermittelt. Danach werden nur die Tupel selektiert, die zu Curie gehören. Folglich ist das Zwischenergebnis (Professoren und Assistenten) groß. Bei Alternative 2 ist die Selektion 'gepusht', d.h. zuerst wird Curie aus der Menge der Professoren selektiert und danach der Join ausgeführt, d.h. das Zwischenergebnis ist kleiner.

Aufgabe 2 b)

1. $\Pi_{name}(\sigma_{semester>5 \wedge Titel='Anfrageoptimierung' \wedge Studenten.MatrNr=hören.MatrNr}($
   $\sigma_{hören.VorlNr=Vorlesungen.VorlNr}(Studenten \times hören \times Vorlesungen))$
2. $\Pi_{name}(\sigma_{semester>5 \wedge Titel='Anfrageoptimierung'}(Studenten \bowtie hören \bowtie Vorlesungen)))$
3. $\Pi_{name}(\sigma_{semester>5}(Studenten) \bowtie hören \bowtie \sigma_{Titel='Anfrageoptimierung'}(Vorlesungen)))$

Lösung

Alternative 1 läuft vermutlich am längsten. Hier wird durch die Kreuzprodukte ein sehr großes Zwischenergebnis produziert, bevor die Selektionen durchgeführt werden. Alternative 2 ist hier besser, da nur die Tupel aus den Basisrelationen im Zwischenergebnis auftauchen, die auch einen Joinpartner finden. Die Selektionen sind aber (im Gegensatz zu Alternative 3) nicht gepusht, d.h. Alternative 3 wird nochmal schneller sein als Alternative 2.

Aufgabe 2 c)

1. $\Pi_{name}(Studenten \bowtie (hoeren \bowtie \sigma_{titel='Anfrageoptimierung'}(Vorlesungen))$
2. $\Pi_{name}((Studenten \bowtie hoeren) \bowtie \sigma_{titel='Anfrageoptimierung'}(Vorlesungen))$

Lösung

Hier läuft vermutlich Alternative 2 am längsten. Es wird zuerst für alle Studenten ermittelt, an welchen Vorlesungen sie teilgenommen haben. Danach werden aus diesem Zwischenergebnis nur die Tupel selektiert, bei denen die zugehörige Vorlesung Anfrageoptimierung ist. Geht man davon aus, dass nur sehr wenige Studenten diese Vertiefungsvorlesung hören, ist Alternative 1 vermutlich schneller: Hier werden zuerst die Teilnehmer der Anfrageoptimierungs-Vorlesung ermittelt, für die dann die zugehörigen Studentenattribute ermittelt werden.

Aufgabe 3

Gegeben sei folgendes relationales Schema.

- Sachbearbeiter(<u>PersNr</u>, Name, Fachgebiet)
- Kunde(<u>Kundennummer</u>, Name, Adresse)
- betreut(<u>PersNr, Kundennummer</u>)

Ein Benutzer stellt nun die folgende SQL Anfrage.

```
select distinct s.name
from sachbearbeiter s, kunde k, betreut b
where s.PersNr = b.PersNr and b.Kundennummer = k.Kundennummer
  and k.name='Schmidt';
```

Aufgabe 3 a)

Welches Ergebnis soll mit der obigen Anfrage vermutlich ermittelt werden?

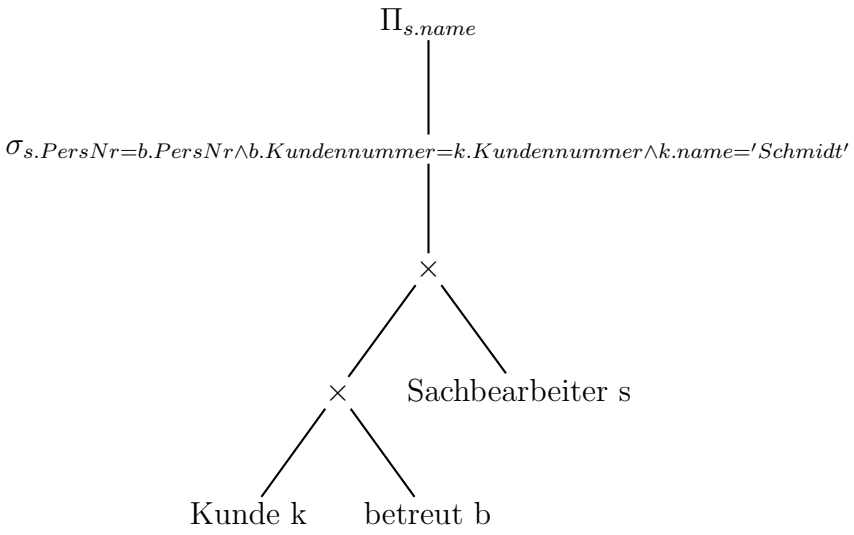
Lösung

Die Namen der Sachbearbeiter, die einen Kunden mit dem Namen 'Schmidt' betreuen.

Aufgabe 3 b)

Überführen Sie die obige SQL Anfrage in einen kanonischen Operatorbaum.

Lösung

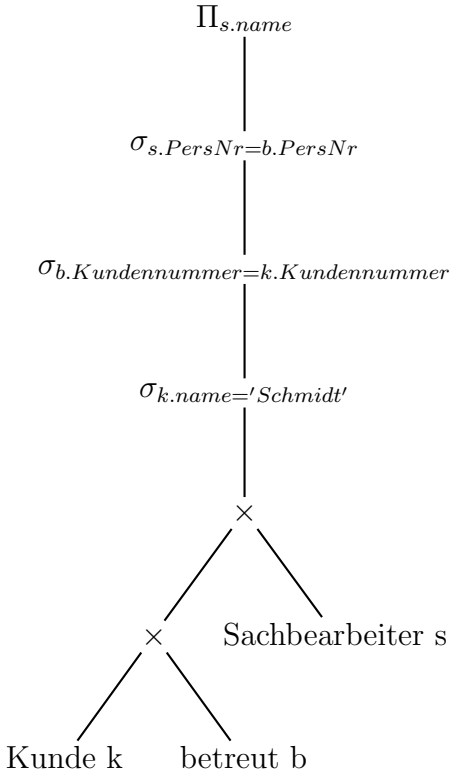


Aufgabe 3 c)

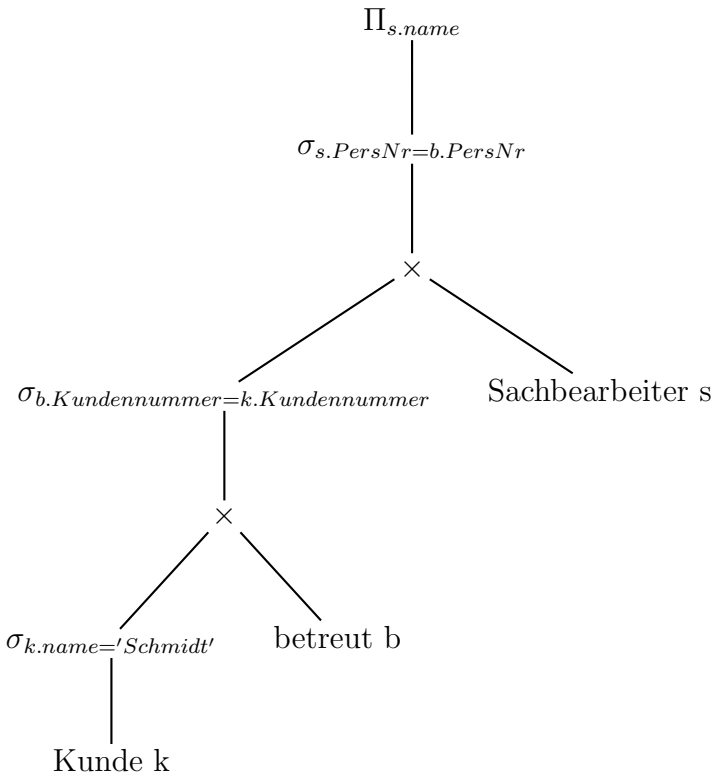
Führen Sie eine (logische) Optimierung des obigen Operatorbaums durch. Verwenden Sie hierfür die in der Vorlesung vorgestellten Heuristiken. Aufgrund fehlender Informationen soll eine Bestimmung der Joinreihenfolge nicht durchgeführt werden.

Lösung

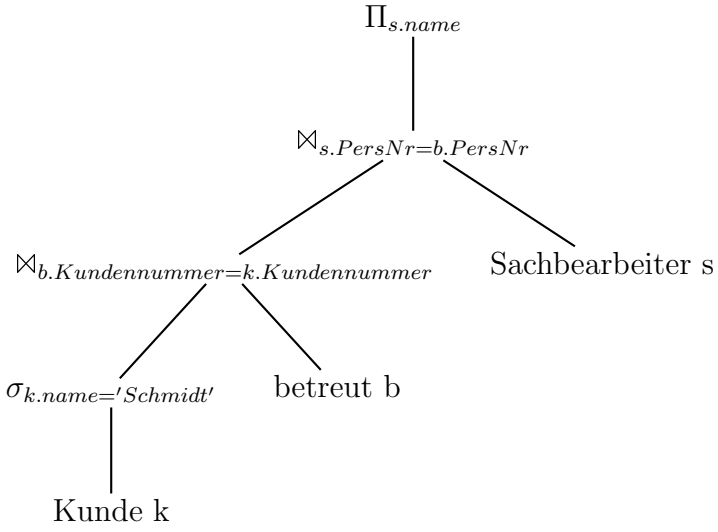
Aufbrechen von Selektionen:



'Pushen' von Selektionen:



Einführen von Joins:



Aufgabe 4

In einer Anfrage werden die Relationen R, S und T verarbeitet. Die Relationen besitzen die folgenden Schemata.

- R (<u>A, B</u>, C)
- S (<u>A, D</u>, E, F)
- T (<u>B</u>, G, H)

Aufgabe 4 a)

Betrachten Sie den (optimierten) Operatorbaum der Anfrage in Abbildung 1. Geben Sie zu diesem Operatorbaum den ursprünglichen, kanonischen Operatorbaum an.

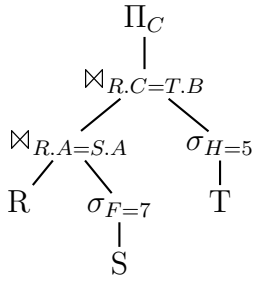


Abbildung 1: Optimierter Operatorbaum

Lösung

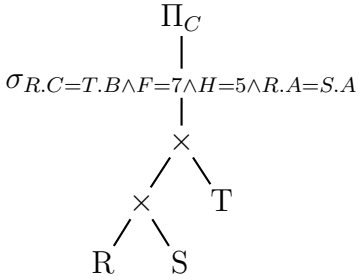


Abbildung 2: Kanonischer Operatorbaum

Aufgabe 4 b)

Betrachten Sie den (optimierten) Operatorbaum der Anfrage in Abbildung 2. Geben Sie zu diesem Operatorbaum den ursprünglichen, kanonischen Operatorbaum an.

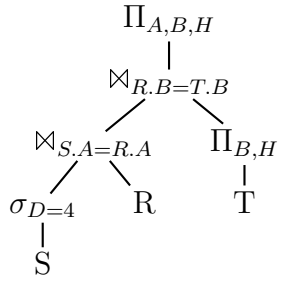


Abbildung 3: Optimierter Operatorbaum

Lösung

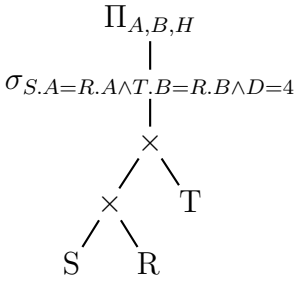


Abbildung 4: Kanonischer Operatorbaum

Aufgabe 5

Aufgabe 5 a)

Gegeben sei der algebraische Ausdruck $\sigma_{p_3}(\sigma_{p1}(R_1) \bowtie_{p1,2} \sigma_{p2}(R_2))$ und folgende Kardinalitäten (Anzahl der Elemente in den Eingaberelationen) bzw. Selektivitäten (Selektivität der Prädikate):

$$\begin{aligned} |R_1| &= 460 \\ |R_2| &= 330 \\ \text{Selektivität}(p_1) &= 1/23 \\ \text{Selektivität}(p_2) &= 1/11 \\ \text{Selektivität}(p_3) &= 1/20 \\ \text{Selektivität}(p_{1,2}) &= 1/6 \end{aligned}$$

Geben Sie für jeden Teilausdruck die Ergebniskardinalität an.

Lösung

$\sigma_{p_1}(R_1): \ 460 * 1/23 = 20$
$\sigma_{p_2}(R_2): \ 330 * 1/11 = 30$
$\sigma_{p_1}(R_1) \bowtie_{p_{1,2}} \sigma_{p_2}(R_2): \ (20 * 30) * 1/6 = 600 * 1/6 = 100$
$\sigma_{p_3}(\sigma_{p_1}(R_1) \bowtie_{p_{1,2}} \sigma_{p_2}(R_2)): \ 100 * 1/20 = 5$

Aufgabe 5 b)

Für die Durchführung von Joins wurden in der Vorlesung verschiedene Implementierunsansätze vorgestellt. Diskutieren Sie die Vor- und Nachteile der verschiedenen Join-Implementierungen am Beispiel von Hash-Join, Merge-Join und (blockwise) Nested-Loop Join.

Lösung

- Nested-Loop Join: Unterstützt beliebige Joinprädikate, jedoch langsam
- Hash-Join: Unterstützt nicht alle Joinprädikate (Probleme z.B. bei $a < b$), dafür schneller als NL Join
- Merge-Join: Unterstützt nicht alle Joinprädikate (Probleme z.B. bei $a < b$), Eingabetupel müssen sortiert sein, dafür schneller als NL Join

Aufgabe 5 c)

In einem einfachen Datenbanksystem steht nur ein Merge-Join zur Verfügung. Welche

Auswirkungen hat dies auf die Plangenerierung? Können in diesem System beliebige Join-Prädikate unterstützt werden?

Lösung

Der Merge-Join kann nicht mit allen Joinprädikaten umgehen (siehe Aufgabenteil b), d.h. es gibt Anfragen, die nicht vom System unterstützt werden können. Um den Merge-Join in einem Auswertungsplan einzusetzen, muss zudem sichergestellt werden, dass die Eingabetupel sortiert sind. Hierfür müssen gegebenenfalls zusätzliche Sortierungsoperationen (sog. Enforcer) in den Plan eingefügt werden .

Aufgabe 6

Der Optimierer eines Datenbanksystems ist noch nicht ganz ausgereift und produziert für eine Anfrage folgenden Operatorbaum:

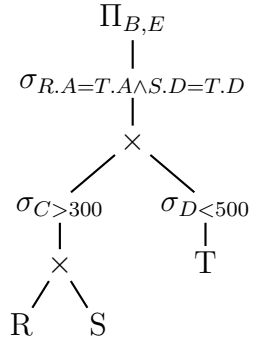


Abbildung 5: Voroptimierter Operatorbaum

Die Relationen besitzen folgendes Schema:

- $R(\underline{A}, B, C)$
- $S(\underline{D}, E, F)$
- $T(\underline{A, D})$

Weiterhin kennt das Datenbanksystem folgende Statistiken über die Relationen.

Kardinalitäten		Selektivitäten	
|R|	100	$\sigma_{C>300}$	$\frac{1}{10}$
|S|	10.000	$\sigma_{T.D<500}$	$\frac{1}{10}$
|T|	100.000	$R \bowtie T$	$\frac{1}{1000}$
		$S \bowtie T$	$\frac{1}{1000}$

Aufgabe 6 a)

Geben Sie für obigen Plan für jeden Teilausdruck dessen Kardinalität (d.h. die Anzahl der von diesem Teilausdruck produzierten Tupel) an.

Lösung

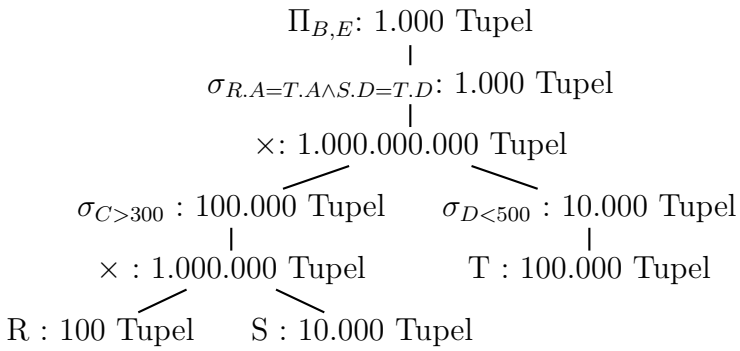


Abbildung 6: Kardinalitäten im voroptimierten Operatorbaum

Aufgabe 6 b)

Verbessern Sie den oben abgebildeten Operatorbaum mit Hilfe der in der Vorlesung vorgestellten Heuristiken.

Lösung

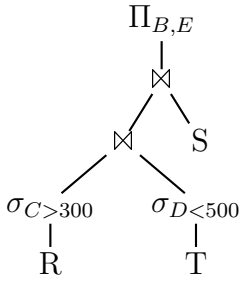


Abbildung 7: Optimierter Operatorbaum

Aufgabe 6 c)

Geben Sie für den in Aufgabenteil b) erstellten, optimierten Operatorbaum für jeden Teilausdruck die Ergebniskardinalität an.

Lösung

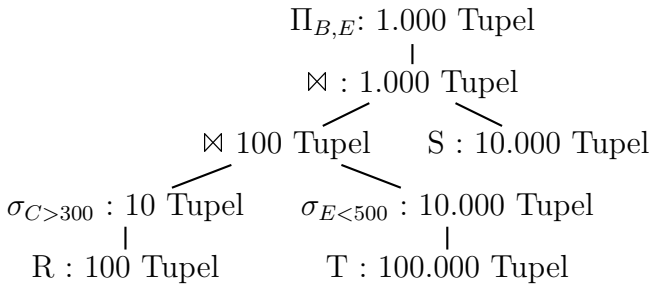


Abbildung 8: Kardinalitäten im optimierten Operatorbaum

Aufgabe 7

In einer Anfrage werden die Relationen R, S und T verarbeitet. Die Relationen besitzen die folgenden Schemata.

- R (RID, A)
- S (SID, B)
- T (TID, SID, C)

```
select distinct r.a
from R r, S s, T t
where r.rid = s.sid and s.sid = t.sid and s.b = 3 and t.c = 4;
```

Aufgabe 7 a)

Überführen Sie die SQL-Anfrage in einen kanonischen Operatorbaum.

Lösung

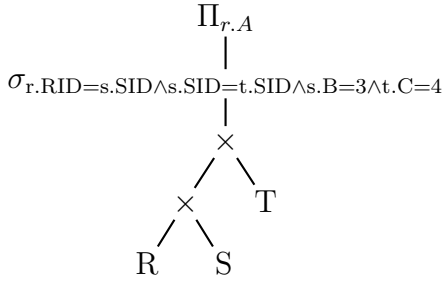


Abbildung 9: Kanonischer Operatorbaum

Aufgabe 7 b)

Führen Sie eine (logische) Optimierung des in Aufgabenteil a) erstellten kanonischen Operatorbaums durch. Verwenden Sie hierfür die in der Vorlesung vorgestellten Heuristiken. Aufgrund fehlender Informationen soll eine Bestimmung der Joinreihenfolge nicht durchgeführt werden.

Lösung

Aufgabe 7 c)

Zu obiger Anfrage sei der untenstehende Operatorbaum gegeben. Die Superskripte der

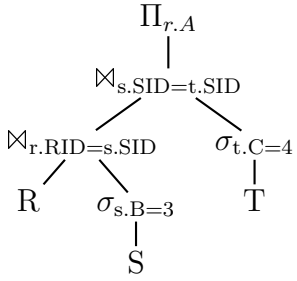
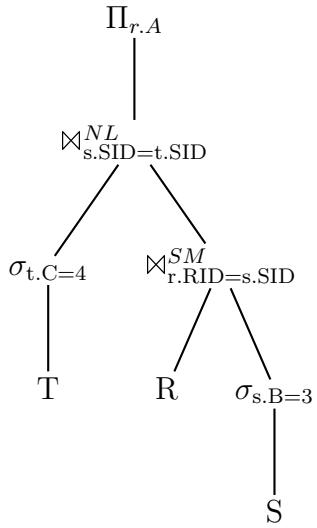


Abbildung 10: Optimierter Operatorbaum

Join Operatoren geben die jeweils verwendete Joinimplementierung (SM = Sort-Merge Join und NL = Nested-Loop Join) an. Die beiden Eingaben des Sort-Merge Joins sind auf dem jeweiligen Joinattribut sortiert. Gehen sie vom Iteratorenkonzept aus, so dass keine Zwischenergebnisse materialisiert/zwischengespeichert werden! Weiterhin seien die Kardinalitäten und Selektivitäten einer Datenbank aus der folgenden Tabelle gegeben.



Kardinalitäten		Selektivitäten	
|R|	1.000	s.B = 3	$\frac{1}{5}$
|S|	500	r.RID = s.SID	$\frac{1}{100}$
|T|	200	s.SID = t.SID	$\frac{1}{200}$
		t.C = 4	$\frac{1}{2}$

Bestimmen Sie für die Ausführung des Operatorsbaum auf der Datenbank die Anzahl der Tupel, auf die insgesamt in den Basisrelationen zugegriffen wird. Beachten Sie, dass mehrfache Zugriffe auf ein Tupel mehrfach gezählt werden.

Lösung

- Beim Sort-Merge Join wird, da das Join Attribut die jeweiligen Schlüssel der Relation vergleicht, jede Eingabe einmal gelesen (1.000 + 500 = 1.500). Achtung: die Ausgabe der Selektion wird nicht zwischengespeichert
- Beim Nested Loop Join wird die linke Eingabe einmal gelesen (200 Tupel) und für $\frac{1}{2}$ der Tupel wird die rechte Eingabe jeweils einmal gelesen, d.h. $200+100\times1.500 = 150.200$. Achtung: die rechte Seite wird für jedes neue Tupel der linken Seite ein

Mal neu ausgeführt, da die Zwischenergebnisse des Sort-Merge Join und damit auch der Selektion über S nicht zwischengespeichert werden!

Aufgabe 8

Gegeben sei der unvollständige Auswertungsplan aus untenstehender Abbildung 4. Fügen Sie an den gekennzeichneten Stellen, wenn nötig, Sortieroperationen ein. Geben Sie genau an, nach welchen Attributen sortiert werden muss. Der resultierende Plan soll möglichst wenige Sortieroperationen enthalten.

Weiterhin ist bekannt, dass die `TableScan`s auf Relationen R und T keine sortierte Ausgabe liefern und dass der `IndexScan` auf einen B^+-Baum zugreift, dessen Ausgabe sortiert auf den Schlüsseln des Indexes, $S.B$, $S.D$, ist.

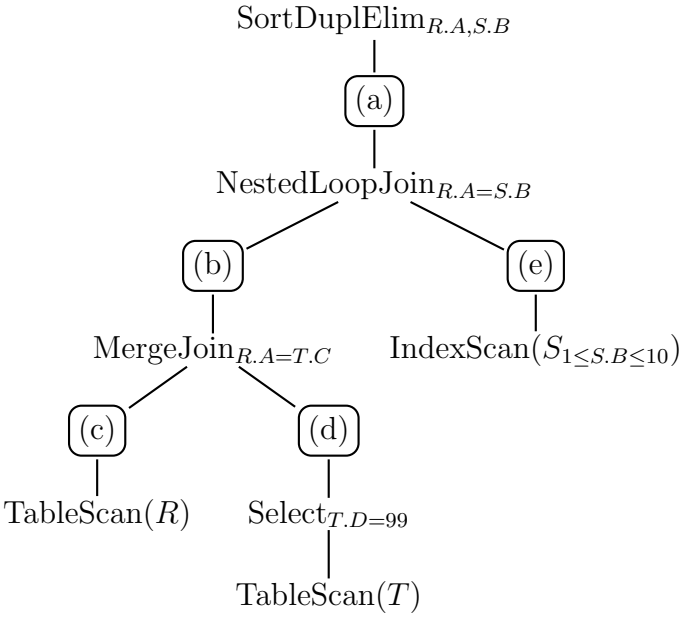


Abbildung 11: ein unvollständiger Auswertungplan

Lösung

(a) keine Sortierung

(b) keine Sortierung

(c) Sort$_{R.A}$

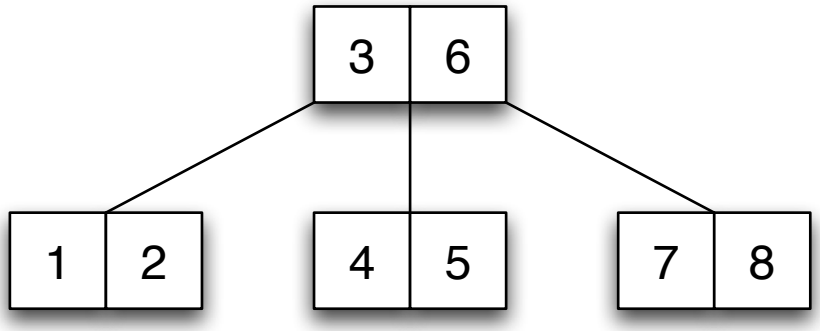
(d) Sort$_{T.C}$

(e) keine Sortierung

Aufgabe 9

Zeichnen Sie einen B-Baum (Grad k = 1) der Höhe *eins* mit *maximaler* Speicherauslastung. Tragen Sie nur Schlüssel aus dem Wertebereich von 1 - 20 ein. Bringen Sie die von Ihnen eingetragenen Schlüssel in eine Reihenfolge, so dass der von Ihnen gezeichnete B-Baum entsteht.

Lösung



Bei maximaler Auslastung des B-Baumes ergeben sich $(2k+1)^{h+1}-1 = (2*1+1)^{1+1}-1 = 8$ Einträge. Eine mögliche Sequenz ist $(2, 4, 3, 7, 6, 1, 5, 8)$.