

## Datenbanksysteme I, SS 2005

### Lösungen zum 9. Übungsblatt

1. (a) Ja, das Tupel wird verschoben, an die Stelle des Tupels wird die neue TID geschrieben.
  - (b) Ja, es nach dem gleichen Prinzip wie in (a) verfahren.
  - (c) Wenn sehr wenige oder gar keine Änderungen an der Datenbasis vorgenommen werden (wie z.B. in einem Data Warehouse). Trotz des Verschiebens des Originaltupels muß an der ursprünglichen Anfangsstelle des Tupels eine TID zurückbleiben. Bei vielen Änderungsoperationen fragmentieren die Seiten stark. Diese Fragmentierung ist bei dem Verfahren ohne Datensatztabelle nicht zu reparieren.
2. Nach Eingabe in die Webschnittstelle ergibt sich folgendes Bild:

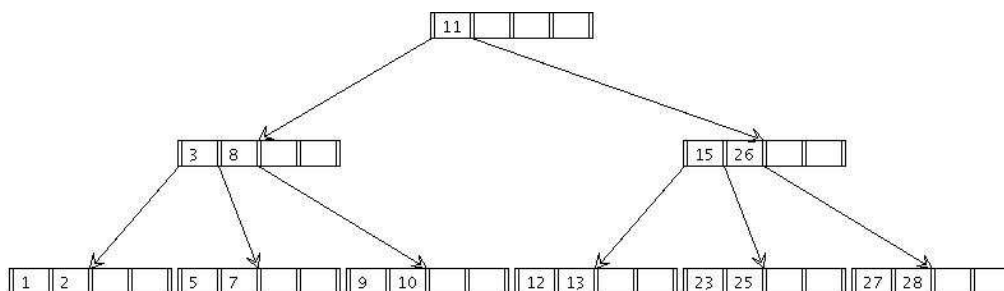


Abbildung 1: B-Baum nach Einfügen der Elemente

3. (a) Ein kanonischer Operatorbaum ist in Abbildung 2 zu sehen.
  - (b) Ein optimierter Operatorbaum ist in Abbildung 3 zu finden.
4. (a)  $\text{Kosten}(\text{minPlan}) := \infty$

```

procedure allPlans(S = {R1, ..., Rm}, Plan, i)
{
    if ( i == n )
    {
        if ( Kosten(Plan) < Kosten(minPlan) )
            minPlan := Plan
    }
}
    
```

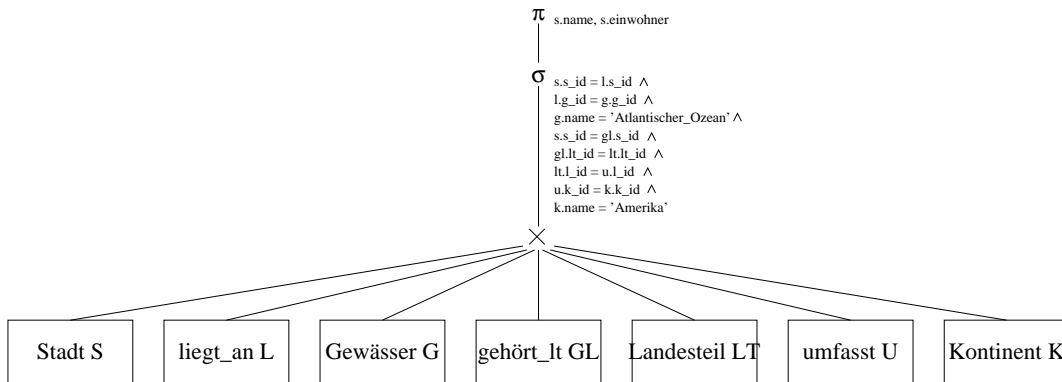


Abbildung 2: Kanonischer Operatorbaum

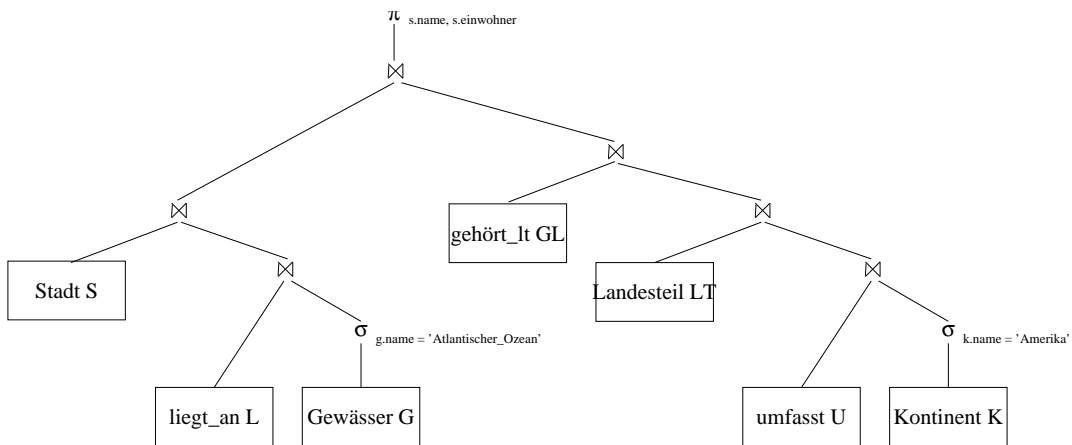


Abbildung 3: Optimierter Operatorbaum

```

}
else
{
  for j := 1 to |S| do
  {
    neuerPlan := Plan ⋈ Rj
    allPlans(S \ {Rj}, neuerPlan, i+1)
  }
}
}

```

Aufruf mit allPlans( $\{R_1, \dots, R_n\}$ , leererPlan, 0)  
 Ergebnis in minPlan

Laufzeit:

$n!$  verschiedene Left-Deep-Tree-Pläne, ein Plan enthält  $n - 1$  Joinoperationen:  
 Gesamtaufwand:  $O(n!n)$

(b) procedure minRel( $S = \{R_1, \dots, R_n\}$ )

```

{
  Suche kleinste Relation  $R_i$ 
  Plan :=  $R_i$ 
  S := S \  $R_i$ 
  while ( S  $\neq \emptyset$  ) do
  {
    suche Relation  $R_j \in S$ , so daß Größe(Plan  $\bowtie R_j$ ) minimal
    Plan := Plan  $\bowtie R_j$ 
    S := S \  $R_j$ 
  }
  return Plan
}

```

Laufzeit:

1. Teil: Minimum aus  $n$  Relationen benötigt  $n$  Schritte.

2. Teil (Schleife):

1. Durchlauf:  $n - 1$  Größen berechnen
2. Durchlauf:  $n - 2$  Größen berechnen
3. Durchlauf:  $n - 3$  Größen berechnen
- usw.

Gesamtaufwand:  $\sum_{i=1}^n i = \frac{n(n+1)}{2} = O(n^2)$

Problem: dieser Algorithmus liefert nicht unbedingt das Optimum.

(c) procedure DynTrees(S = { $R_1, \dots, R_n$ })

```

{
  Q =  $\emptyset$ ;
  for i := 1 to n do
    optPlan( $R_i$ ) =  $R_i$ 
    Q = Q  $\cup$  {optPlan( $R_i$ )}
  }
  for i := 2 to n do
  {
    for all  $q \in Q$  do
    {
      for all  $R_j \in S$  do
        if (  $R_j \notin q$  )
        {
           $p = q \bowtie R_j$ 
          if ( Kosten( $p$ ) < Kosten(optPlan(Rel( $p$ ))) )
            ersetze optPlan(Rel( $p$ )) mit  $p$ ;
        }
      }
    }
  }
  return Q;
}

```

$\text{Rel}(p)$  bestimmt die Relationen, die im Plan  $p$  enthalten sind (ohne Berücksichtigung der Joinreihenfolge). Wir merken uns für jede Untermenge aus  $S$  immer nur einen Plan.

Laufzeit:

Baue  $n$  Pläne mit je einer Relation:  $O(n)$ .

Durchlaufe schrittweise ( $i$  von 2 bis  $n$ ) alle  $i$ -elementigen Mengen aus  $S$  und expandiere sie um die Relationen, die noch nicht dazugejoint wurden:

$$\sum_{i=2}^n \binom{n}{i} \cdot n = n \cdot \sum_{i=2}^n \binom{n}{i} = n \cdot (2^n - 1 - n)$$

Gesamtlaufzeit:  $O(n + n \cdot 2^n - n - n^2) = O(n \cdot 2^n)$

5. Formel zur Berechnung der Seitenzugriffe:

- Durchlaufen aller Seiten von  $R$ :  $b_R$
  - Durchläufe der inneren Schleife:  $\lceil b_R / (m - k) \rceil$
  - Insgesamt:  $b_R + k + \lceil b_R / (m - k) \rceil \cdot (b_S - k)$
- (a) Einsetzen in obige Formel:  $60 + 10 \cdot 416 + 4 = 4224$
- (b) Kosten minimal, wenn  $k = 1$  und  $R$  die kleinere Relation.
- (c) Einsetzen in Formel:  $60 + 7 \cdot 419 + 1 = 2994$