UNIVERSITÄT DES SAARLANDES
Fachbereich 14 – Informatik
Prof. Dr.-Ing. G. Weikum
Dipl.-Inform. R. Schenkel

Datenbanksysteme WS 1999/2000

Musterlösung zur ersten Teilklausur

Aufgabe 1: Indexgrößen (1+1=2 Punkte)

Berechnen Sie die Anzahl der Blattseiten, den Fanout und die Baumhöhe für jede der beiden folgenden mittelbaren, unclustered B*-Baum-Indexstrukturen. Tragen Sie Ihre Ergebnisse in die beiliegende Tabelle ein. (Die notwendigen Rechnungen, die Sie zu diesen Ergebnissen geführt haben, müssen Sie natürlich ebenfalls abgeben.)

a) Ein Index *LB* über das Attribut `Bezeichnung` von Lizenzen, und
b) ein Join-Index *J* über die Konkatenation der Attribute `Bezeichnung` und `Sprache` beider Relationen.

Lösung:

a) Anzahl Tupel in L: 10.000.000
Anzahl verschiedener B-Werte (= Anzahl der Schlüssel): **200**
⇒ 10.000.000/200 = 50.000 TIDs pro Schlüssel
Länge eines Indexeintrags auf Blattniveau: 150 + 50.000*6 = 300.150
⇒ Anzahl der Blätter für eine TID-Liste: $\lceil 300.150/3.000 \rceil = 101$
⇒ Anzahl der Blätter insgesamt: 200*101 = **20.200**
Länge eines Indexeintrags in inneren Knoten: 150 + 6 = 156
⇒ Fanout: $\lfloor 3.000/156 \rfloor$ = **19**
⇒ Höhe des Baums: $\lceil \log_{19} 200 \rceil + 1$ = **3**

b) Anzahl Tupel in L+P: 10.000.000+10.000 = 10.010.000
Anzahl verschiedener B||S-Werte (= Anzahl der Schlüssel): 200*50=**10.000**
⇒ 10.010.000/10.000 = 1.001 TIDs pro Schlüssel
Länge eines Indexeintrags auf Blattniveau: 150 + 50 + 1.001*6 = 6.206
⇒ Anzahl der Blätter für eine TID-Liste: $\lceil 6.206/3.000 \rceil = 3$
⇒ Anzahl der Blätter insgesamt: 10.000*3 = **30.000**
Länge eines Indexeintrags in inneren Knoten: 150 + 50 + 6 = 206
⇒ Fanout: $\lfloor 3.000/206 \rfloor$ = **14**
⇒ Höhe des Baums: $\lceil \log_{14} 10.000 \rceil + 1$ = **5**

Aufgabe 2: Ausführungspläne (1+2+2=5 Punkte)

Nehmen Sie an, alle fünf in Aufgabe 1 gegebenen Indexstrukturen seien vorhanden. Bestimmen Sie die kostengünstigsten Ausführungspläne (und ihre Kosten) für die folgenden Anfragen. Das verwendete Datenbanksystem unterstützt lediglich Nested Loop, Nested Loop mit Indexzugriff, Sort-Merge, Index-Scan und Relationen-Scan. Der für das Sortieren zur Verfügung stehende Hauptspeicher beträgt 50 MByte, der Datenbank-Seitenpuffer ist so klein, dass Caching-Effekte vernachlässigbar sind.

a) `SELECT * FROM Lizenzen WHERE Bezeichnung="Windows NT 4.0 Workstation" AND Sprache="Deutsch"`

Scan von L:
$\lfloor 3000/300 \rfloor = 10$ Tupel pro Seite, also
10.000.000/10 = 1.000.000 SIO
Index-Scan über LBS:
Anzahl Indexzugriffe (einschl. Blätter): 5-1+2=6 RIO
Anzahl der Treffertupel: 1.000, also 1.000 RIO
⇒ 1.006 RIO
Index-Scan über LB:
Wesentlich mehr RIO als Scan von LBS (Faktor 50)
Index-Scan über LS:
Wesentlich mehr RIO als Scan von LBS (Faktor 200)
⇒ **Index-Scan über LBS ist am effizientesten.**

b)
```
SELECT * FROM Lizenzen L, Produkte P
WHERE L.Bezeichnung=P.Bezeichnung AND L.Sprache=P.Sprache
```

Join mit Join-Index J:
Anzahl Indexzugriffe (einschl. Blätter): 5-1+30000 = 30.004 RIOs
Anzahl der Treffertupel: 10.000*1.001 = 10.010.000 RIOs
⇒ 10.040.004 RIOs
Sort-Merge-Join:
Tupel pro Seite (wie oben): 10
Anzahl Seiten in L: 10.000.000/10 = 1.000.000 (=4GB)
Anzahl Seiten in P: 10.000/10 = 1.000 (=4MB)
Speichergröße 50MB = 12.500 Seiten > sqrt(1.000.000), also genügt ein Sortierdurchlauf
Kosten des Sortierens: 4*1.001.000 SIOs
Kosten des Mischens: 1.001.000 SIOs
⇒ 5.005.000 SIOs
Nested-Loop-Join ohne Index:
P als innere, L als äußere Relation
1.000.000 Seiten in L, 1.000 Seiten in P
⇒ 1.000.000 * (1 RIO + 10*Scan von P) = 1.000.000 * (1 RIO + 10.000 SIO)
= 10.000.000.000 SIO + 1.000.000 RIO
P als äußere, L als innere Relation
⇒ 1.000 * (1 RIO + 10 * Scan von L) = 1.000 * (1 RIO + 10.000.000 SIO)
= 10.000.000.000 SIO + 1.000 RIO
Nested-Loop-Join mit Index:
P als innere, L als äußere Relation
1.000.000 Seiten in L, PBS-Index hat Tiefe 4 und TID-Listenlänge 1
⇒ 1.000.000 * (1 RIO + 10 * Indexzugriff) = 1.000.000 * (1 RIO + 10*(4-1+1) RIO)
= 1.000.000 RIO + 40.000.000 RIO = 41.000.000 RIO
P als äußere, L als innere Relation
1.000 Seiten in P, LBS-Index hat Tiefe 5 und TID-Listenlänge 1000 (⇒ 3 Blätter)
⇒ 1.000 * (1 RIO + 10 * Indexzugriff)
= 1.000 RIO + 10.000*(5-1+2+1.000 RIO)
= 1.000 RIO + 10.060.000 RIO = 10.061.000 RIO
⇒ **Sort-Merge-Join ist am effizientesten.**

c)
```
SELECT Endkundennummer,Lizenznummer,L.Bezeichnung
FROM Lizenzen L, Produkte P
WHERE L.Bezeichnung = P.Bezeichnung AND L.Sprache=P.Sprache
AND P.Freigabedatum>="1. Dezember 1999"
```

Join mit Join-Index J: wie oben 10.040.004 RIOs
Sort-Merge-Join: wie oben 5.005.000 SIOs
Nested-Loop-Join mit P als innerer, L als äußerer Relation:
Wie oben bestenfalls 10.061.000 RIO
Nested-Loop-Join mit P als äußerer, L als innerer Relation, mit LBS-Indexzugriff:
1.000 Seiten von P müssen in jedem Fall gelesen werden
zu 0.001*10.000=10 Tupeln müssen die zugehörigen L-Tupel gefunden werden
⇒ 1.000 RIO + 10*(Zugriff auf zugehörige L-Tupel)
= 1.000 RIO + 10*(Indexzugriff + 1.000 RIO)
= 1.000 RIO + 10*(5-1+2 + 1000 RIO)
= 1.000 RIO + 10*1006 RIO
= 1.000 RIO + 10.600 RIO
= 11.600 RIO
⇒ **NL-Join mit P als Outer und L als Inner ist am effizientesten.**

Aufgabe 3: Algebraische Optimierung (3 Punkte)

Nehmen Sie an, es gelte die folgende Integritätsbedingung, die sicherstellt, dass Softwareprodukte erst ab ihrer Freigabe verkauft werden:

```
CREATE ASSERTION KaufFreigabeConstraint CHECK
(Lizenzen.Kaufdatum>=ALL(SELECT P.Freigabedatum FROM Produkte P WHERE
P.Bezeichnung=Lizenzen.Bezeichnung AND P.Sprache=Lizenzen.Sprache)
```

Zusätzlich gebe es einen Index über das Attribut `Kaufdatum` der Relation `Lizenzen`. Betrachten Sie nun erneut die folgende SQL-Anfrage aus Aufgabe 2c:

```
SELECT Endkundennummer,Lizenznummer,Bezeichnung FROM Lizenzen L,
Produkte P WHERE P.Freigabedatum>=“1. Dezember 1999“ AND
L.Bezeichnung=P.Bezeichnung AND L.Sprache=P.Sprache
```

bzw. den daraus erzeugten Algebraausdruck

```
Π[Endkundennummer,Lizenznummer,Bezeichnung](σ[Freigabedatum>=“1. Dezember
1999“](Produkte |x| Lizenzen))
```

a) Formulieren Sie die Integritätsbedingung als Ausdruck der Relationenalgebra.

```
  (σ[F>=“1.12.1999“](P))|x| L
= (σ[F>=“1.12.1999“](P))|x|(σ[K>=“1. Dezember 1999“](L))
```

b) Optimieren Sie die als Algebraausdruck dargestellte Anfrage mit Hilfe der in der folgenden Liste enthaltenen algebraischen Äquivalenzregeln, so dass der Join auf möglichst kleinen Operanden arbeitet und die vorhandenen Indizes ausgenutzt werden. Nutzen Sie dabei auch die Integritätsbedingung aus. Geben Sie für jeden Transformationsschritt an, welche Äquivalenzregel Sie anwenden.

```
        Π[EK,L,B](σ[F>=“1.12.1999“](P|x|L))
(R8) ⇔  Π[EK,L,B]((σ[F>=“1.12.1999“](P))|x|L)
(IB) ⇔  Π[EK,L,B]((σ[F>=“1.12.1999“](P))|x|(σ[K>=“1.12.1999“](L))
(R4) ⇔  Π[EK,L,B](Π[EK,L,B,S](
          (σ[F>=“1.12.1999“](P))|x|(σ[K>=“1.12.1999“](L)))
(R9) ⇔  Π[EK,L,B]((Π[B,S](σ[F>=“1.12.1999“](P)))
         |x|(Π[EK,L,B,S](σ[K>=“1.12.1999“](L)))
```