UNIVERSITÄT DES SAARLANDES
Fachbereich 14 – Informatik
Lehrstuhl Prof. Weikum

Datenbanksysteme WS 1999/2000

Übung 8 — Abgabe: 10.1.2000

Aufgabe 1: Algebraische Optimierung

Gegeben seien die beiden Relationen

Produkte (PNr, Bez, Preis, Gewicht, Vorrat) und
Bestellungen (BestNr, Monat, Tag, KNr, PNr, Menge, Status)

sowie die View

BestInfo (BestNr, PNr, Bez, Gewicht, Menge)
:= π[BestNr,PNr, Bez, Gewicht, Menge] (Produkte $\bowtie$ Bestellungen).

Es gelte die Integritätsbedingung:

π[PNr, Bez, Preis, Gewicht, Vorrat] (Produkte) =
π[PNr, Bez, Preis, Gewicht, Vorrat] (Produkte $\bowtie$ Bestellungen),

d.h. jedes Produkt wird mindestens einmal bestellt und es gibt nur Bestellungen für existierende Produkte.

Betrachten Sie nun die folgende Query:

π[PNr, Bez] (σ[Gewicht > 100] (BestInfo)).

Vereinfachen Sie diese Query mit Hilfe der Äquivalenzregeln der Relationenalgebra und der oben angegebenen Integritätsbedingung, so daß die Query ohne Join-Berechnung ausgeführt werden kann.

Aufgabe 2: Algebraische Query-Transformation

Gegeben ist der Ausschnitt eines relationalen Schemas einer Videotheks-Datenbank, das durch folgende SQL-Anweisungen erzeugt wurde:

CREATE TABLE **F**(**FNr** INTEGER PRIMARY KEY, **Titel** VARCHAR(80)); – Filme

CREATE TABLE **B**(**BNr** INTEGER PRIMARY KEY,**FNr** INTEGER,**AL** INTEGER,
FOREIGN KEY *FNr* REFERENCES *F(FNr)* ON DELETE CASCADES); – Bänder

CREATE TABLE **A** (**BNr** INTEGER, **von** INTEGER, **bis** INTEGER); – Ausleihen

Das Attribut AL in Relation B enthalte die Anzahl der Ausleihen eines Bandes. Die folgende Query ermittelt alle Filmnummern von Filmen, von denen ein Band existiert, das schon mehr als 100 mal ausgeliehen wurde.

$\pi[F.FNr](\sigma[AL > 100]((A \bowtie [A.BNr = B.BNr]\ B) \bowtie [B.FNr = F.FNr]\ F))$

Transformieren Sie die Query so, daß unnötige Operationen eliminiert und essentielle Operationen auf möglichst kleinen Operanden ausgeführt werden. Berücksichtigen Sie dabei insbesondere die spezifizierte Fremdschlüsselbedingung. Geben Sie bei jedem Transformationsschritt an, welche der folgenden Regeln Sie angewendet haben:

IP: $\pi[R1]\pi[R2](R) = \pi[R1](R)$, wenn $R1 \subseteq R2$
DSJ: $\sigma[P1](R \bowtie [P2]S) = \sigma[P1](R) \bowtie [P2]S$,
wenn Prädikat $P1$ nur Attribute aus R enthält
DPJ: $\pi[R1, S1](R \bowtie [P]S) = \pi[R1](R) \bowtie [P]\pi[S1](S)$,
wenn $R1 \subseteq sch(R)$ und $S1 \subseteq sch(S)$
und P nur Attribute aus $R1 \cup S1$ enthält
KJ: $R \bowtie [P]S = S \bowtie [P]R$
CO1: $\pi[A](R \bowtie [A = B]S) = \pi[B](R \bowtie [A = B]S)$
CO2: $\pi[R1, A](R) \bowtie [A = B]\pi[B](S) = \pi[R1, A](R)$,
wenn $\pi[A](R) \subseteq \pi[B](S)$

Aufgabe 3: Optimierung von Joins

In einer Videotheks-Datenbank sind u.a. folgende beiden Relationen gegeben:

BAENDER(BNr , ...) mit dem Primärschlüssel BNr (Bandnummer).

AUSLEIHEN(..., BNr, ...) mit BNr als Fremdschlüssel bzgl. BAENDER.

Die Häufigkeit von Ausleihen sei für alle Bänder gleich. Für den Primärschlüssel der Relation BAENDER existiere ein Index in Form eines B*-Baumes. Folgende Parameter der Relationen seien bekannt:

Relation	**Parameter**	**Wert**
BAENDER	Anzahl von Tupeln	4000
	Speicherbedarf eines Tupels	100 Byte
AUSLEIHEN	Anzahl von Tupeln	40 000
	Speicherbedarf eines Tupels	25 Byte

Außerdem habe eine Seite die Größe 2048 Byte, wobei 48 Byte für den Seitenheader verwendet werden und somit 2000 Byte für Tupel zur Verfügung stehen. Das Attribut BNr belege in beiden Relationen 14 Byte und ein TID setzte sich aus 6 Byte zusammen.

Es soll der Natural Join zwischen den beiden Relationen berechnet werden, wobei die Schnittmenge der beiden Attributmengen nur aus dem Attribut BNr bestehe. Für die Ausführung des Joins stehe ein Puffer von zwei Seiten zur Verfügung, der von Index- und Datenseiten gemeinsam genutzt wird.

a) Ist es bezüglich der Plattenzugriffskosten günstiger, einen Nested-Block-Join oder einen Nested-Loop-Join mit Indexzugriff auf die innere Relation zu verwenden? Stellen Sie dazu die Anzahl der benötigten Seitenzugriffe beider Verfahren gegenüber.

b) Ab welcher Kardinalität der Relation BAENDER (Anzahl Tupel) würde der Nested-Loop-Join mit Indexzugriff auf die innere Relation gegenüber dem Nested-Block-Join mit der Relation AUSLEIHEN in der äußeren Schleife eine gleichwertige bzw. kostengünstigere Ausführungsstrategie sein? Gehen Sie bei Ihrer Berechnung davon aus, daß die Relation AUSLEIHEN eine konstante Kardinalität von 40000 Tupeln besitzt.

Aufgabe 4: Kostenschätzung von Joins

Gegeben seien die beiden Relationen R und S mit dem gemeinsamen Joinattribut J vom Typ $NUMBER$. J ist Primärschlüssel der Relation S und Fremdschlüssel von R, wobei angenommen wird, daß die Werte gleichverteilt sind. R enthalte 10000 Tupel der Länge 360 Byte und S 200 Tupel der Länge 180 Byte. Nehmen Sie an, daß beide Relationen und alle Indizes jeweils einen eigenen Tablespace belegen, wobei die Seitengröße 4 KBytes sei und der Seiten-Header die Länge 96 Bytes habe. Datenseiten seien zu 90 Prozent und Indexseiten zu 70 Prozent gefüllt. Der Datentyp $NUMBER$ belege 10 Bytes und ein TID oder Zeiger im B*-Baum habe die Länge 6 Bytes.

Bestimmen Sie die Anzahl der benötigten Blockzugriffe für die Joinberechnung nach Nested-Loop, Nested-Block, Nested-Loop mit Indizes über R bzw. S und Merge-Sort. Nehmen Sie dazu an, daß die Größe des

zur Verfügung stehenden Datenpuffers einmal 2 Seiten und einmal 11 Seiten betrage. Benutzen Sie für das Sortieren durch Mischen den Datenpuffer als internen Speicher für den Sortierschritt. Für die Indexseiten stehe ein separater Puffer der Größe 2 Seiten zur Verfügung.

Aufgabe 5: Kostenschätzung und Tuning

In dieser Aufgabe sollen verschiedene Alternativen für den physischen Entwurf einer Oracle-Datenbank betrachtet und bewertet werden. Hierzu sollen Sie mittels der Analyse von Ausführungsplänen die zu erwartenden Anfragekosten schätzen.

Daten

Wir betrachten eine Relation R, die zwei numerische Attribute A und B enthält. Die beiden Attribute haben jeweils die Länge 4 Bytes. In dieser Relation sind 50 000 Tupel gespeichert (das entspricht 345 Datenbankseiten à 2KBytes). Die Attributwerte von A sind aus dem Intervall $[0 \ldots 999]$ mit Gleichverteilung der Tupel über die Attributwerte. Die Attributwerte von B sind aus dem Intervall $[0 \ldots 999]$, wobei 40 000 Tupel über das Intervall $[0 \ldots 99]$ gleichverteilt und 10000 Tupel über das Intervall $[100 \ldots 999]$ gleichverteilt sind.

Problemstellung

Voraussetzung für die Bewertung eines Entwurfs ist die genaue Kenntnis der Anwendungslast. Wir haben dieses in der Praxis nichttriviale Problem stark vereinfacht und wollen die verschiedenen Entwurfsalternativen nur bezüglich der drei folgenden Anfragen beurteilen:

```
Q1:  SELECT COUNT(*)
     FROM R
     WHERE A = 321 AND B = 50;
Q2:  SELECT COUNT(*)
     FROM R
     WHERE A = 321 AND B <> 50;
Q3:  SELECT COUNT(*)
     FROM R
     WHERE A <> 321 AND B = 50;
```

Entwurfsalternativen

Es sollen insgesamt sechs verschiedene Alternativen des physischen Entwurfs miteinander verglichen werden.

1. Es ist lediglich die Relation R gespeichert (d.h. es gibt keinerlei Indizes)
2. Es existiert ein Index über dem Attribut A
3. Es existiert ein Index über dem Attribut B
4. Es existiert sowohl über A als auch über B ein Index
5. Es existiert ein „clustered“ Index über A, d.h. die Relation ist zusätzlich bezüglich A physisch sortiert
6. Es existiert ein kombinierter Index über A und B (die Attributwerte werden konkateniert und als ein Wert im Index verwaltet)

Alle Indizes seien mittelbar und — mit Ausnahme der Alternative 5 — jeweils „unclustered“.

Versuchen Sie nun, die verschiedenen physischen Entwürfe miteinander zu vergleichen, indem Sie für die drei Anfragetypen die aufgrund der physischen Entwürfe möglichen Ausführungsstrategien bezüglich ihrer I/O-Kosten analysieren.

Füllen Sie dann untenstehende Tabelle aus, indem Sie für jede Query und jede Entwurfsalternative die geschätzten I/O-Kosten der jeweils besten Ausführungsstrategie eintragen.

Entwurfsalternative	Queries		
	Q1	Q2	Q3
kein Index			
Index über A			
Index über B			
Indizes über A und B			
Clusterd Index			
Kombinierter Index			

Es ist nicht nötig, für alle Queries und alle Entwürfe eine vollständige Analyse durchzuführen. Durch Überlegung lassen sich einige Resultate auf andere Entwürfe übertragen.

Aufgabe 6: Kostenschätzung mit Histogramm

Für die im Hinblick auf Queries wichtigsten Attribute einer Datenbank sollen Histogramme der folgenden Form im Datenbankkatalog abgelegt sein:

```
hist(R.A): /* Histogramm fuer
              Attribut A der
              Relation R    */

           array[1..maxbin] of
              record
                 low  : dom(A);
                 high : dom(A);
                 count: integer;
                 freq : integer;
              end;
```

Dabei sind $hist(R.A)[i].count$ die Anzahl der in der Relation vorkommenden verschiedenen Attributwerte und $hist(R.A)[i].freq$ die Anzahl der vorkommenden Tupel, die in das Intervall von $hist(R.A)[i].low$ bis $hist(R.A).high$ fallen (einschließlich der Grenzen).

Es gilt (bei $dom(A) = integer$, sonst analog):
$hist(R.A)[i].high = hist(R.A)[i+1].low - 1$ für $1 \leq i < maxbin$.

Geben Sie (als Formel oder Pseudocode) an, wie man aufgrund dieser Histogramme die Resultat-Kardinalitäten der folgenden Anfragetypen schätzen kann:

a) $\sigma[R.A = x](R)$ Exact-Match-Query mit $x \in dom(A)$

b) $\sigma[a \leq R.A \leq b](R)$ Range-Query mit $a, b \in dom(A)$

c) $\pi[R.A](R)$ Projektion mit Duplikateliminierung

d) $R \bowtie [R.A = S.B]\ S$ Equijoin