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

Datenbanksysteme WS 1999/2000

Übung 8 **Beispiellösung zu den Aufgaben 2 und 3**

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.

$$\pi_{[F.FNr]}\big(\sigma_{[AL>100]}\big((A \;|x|_{[A.BNr=B.BNr]}\; B) \;|x|_{[B.FNr=F.FNr]}\; F\big)\big)$$

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: $\sigma_{[P1]}\,(R\;|x|_{[P2]}\;S) = \sigma_{[P1]}\,(R)\;|x|_{[P2]}\;S$, wenn P1 nur Attribute aus R enthält

DPJ: $\pi_{[R1,S1]}\,(R\;|x|_{[P]}\;S) = \pi_{[R1]}(R)\;|x|_{[P]}\;\pi_{[S1]}(S)$,
wenn $R1 \subseteq sch(R)$ und $S1 \subseteq sch(S)$ und P nur Attribute aus $R1 \cup S1$ enthält.

KJ: $R\;|x|_{[P]}\;S = S\;|x|_{[P]}\;R$

CO1: $\pi_{[A]}\,(R\;|x|_{[A=B]}\;S) = \pi_{[B]}\,(R\;|x|_{[A=B]}\;S)$

CO2: $\pi_{[R1,A]}\,(R)\;|x|_{[A=B]}\;\pi_{[B]}\,(S) = \pi_{[R1,A]}\,(R)$, wenn $\pi_{[A]}\,(R) \subseteq \pi_{[B]}\,(S)$

Lösung:

$$\pi_{[F.FNr]}\big(\sigma_{[AL>100]}\big((A \;|x|_{[A.BNr=B.BNr]}\; B) \;|x|_{[B.FNr=F.FNr]}\; F\big)\big)$$

[KJ] $$\pi_{[F.FNr]}\big(\sigma_{[AL>100]}\big((B \;|x|_{[A.BNr=B.BNr]}\; A) \;|x|_{[B.FNr=F.FNr]}\; F\big)\big)$$

[DSJ] $\pi_{[F.FNr]}(\sigma_{[AL>100]} (B) |x|_{[A.BNr=B.BNr]} A) |x|_{[B.FNr=F.FNr]} F)$

[DSJ] $\pi_{[F.FNr]}((\sigma_{[AL>100]} (B) |x|_{[A.BNr=B.BNr]} A) |x|_{[B.FNr=F.FNr]} F)$

[CO1] $\pi_{[B.FNr]}((\sigma_{[AL>100]} (B) |x|_{[A.BNr=B.BNr]} A) |x|_{[B.FNr=F.FNr]} F)$

[IP] $\pi_{[B.FNr]} \pi_{[B.FNr,F.FNr]}((\sigma_{[AL>100]} (B) |x|_{[A.BNr=B.BNr]} A) |x|_{[B.FNr=F.FNr]} F)$

[DPJ] $\pi_{[B.FNr]} (\pi_{[B.FNr]} (\sigma_{[AL>100]} (B) |x|_{[A.BNr=B.BNr]} A) |x|_{[B.FNr=F.FNr]} \pi_{[F.FNr]} (F))$

[CO2] $\pi_{[B.FNr]} (\pi_{[B.FNr]} (\sigma_{[AL>100]} (B) |x|_{[A.BNr=B.BNr]} A))$

[IP] $\pi_{[B.FNr]} (\sigma_{[AL>100]} (B) |x|_{[A.BNr=B.BNr]} A)$

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.

Lösung:

Relation BAENDER belegt S(BAENDER)= $\left\lceil \frac{4000 \times 100}{2048 - 48} \right\rceil = 200$ Seiten.

Relation AUSLEIHEN belegt S(AUSLEIHEN)= $\left\lceil \frac{40\,000 \times 25}{2048 - 48} \right\rceil = 500$ Seiten.

Anzahl Einträge in einem Index–Knoten ist $\frac{2048 - 48}{14 + 6} = 100$.

Anzahl Blätter im Index ist $\frac{4000}{100} = 40$ Seiten.

Höhe des Index ist $\lceil \log_{100}(40) \rceil + 1 = 2$.

Kosten NESTED–BLOCK–JOIN:

Relation AUSLEIHEN in innere Schleife, BAENDER in äußere Schleife, damit die Kosten (bei gegebener Puffergröße von zwei Seiten) minimal sind.

Anzahl der Plattenzugriffe=

$S(BAENDER) \times (1 + S(AUSLEIHEN)) = 200 + 200 \times 500 = 100200$

Kosten NESTED–LOOP–JOIN mit Indexzugriff auf innere Relation:

Man beachte, daß die Wurzel wegen der Puffergröße von 2 Seiten nicht im Puffer gehalten werden kann und deshalb bei jedem Abwärtslauf im Baum neu geladen werden muß. Außerdem wird wegen der Indizierung eines Primärschlüssels maximal eine TID ermittelt, die dann zu einem Seitenzugriff führt, um das Tupel von BAENDER mit dem Tupel der äußeren Relation zu joinen.

Anzahl der Plattenzugriffe=

$S(AUSLEIHEN) + \#Tupel(AUSLEIHEN) \times (H(Index) + 1) = 500 + 40000 \times 3 = 120500$

Der NESTED–BLOCK–JOIN ist hier also kostengünstiger als der NESTED–LOOP–JOIN mit Indexzugriff auf innere Relation !!!

(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 40 000 Tupeln besitzt.

Lösung:

Gesucht: Minimale Anzahl b von Tupeln in Relation BAENDER, für die gilt:

Kosten(Nested–Block–Join) ≥ Kosten(Nested–Loop–Join mit Index), also

$S(AUSLEIHEN) + (1 + S(BAENDER)) \geq S(AUSLEIHEN) + \#Tupel(AUSLEIHEN)(Hoehe(Index) + 1)$

Es gilt:

$$S(BAENDER) = \left\lceil \frac{b \times 100}{2000} \right\rceil = \left\lceil \frac{b}{20} \right\rceil$$

$$Hoehe(Index) = \left\lceil \log_{100}\left(\frac{b}{100}\right) + 1 \right\rceil$$

Damit muß gelten:

$$500 \times \left(1 + \left\lceil \frac{b}{20} \right\rceil\right) \geq 500 + 40000 \times \left(\left\lceil \log_{100}(\frac{b}{100}) + 1 \right\rceil + 1 \right)$$

.. und nach Umformung:

$$\frac{1}{80} \times \left\lceil \frac{b}{20} \right\rceil - 1 \geq \lceil \log_{100}(b) \rceil$$

Eine Bereichsabschätzung von b ergibt, daß b zwischen 101 und 10000 liegen muß, da für diese Werte die rechte Seite der Ungleichung 2 ergibt und die (monoton mit b steigende) linke Seite der Ungleichung für b=10000 einen Wert größer als 2 ergibt. Es ist also folgende Ungleichung aufzulösen:

$\frac{1}{80} \times \left\lceil \frac{b}{20} \right\rceil - 1 \geq 2$, was zu $\left\lceil \frac{b}{20} \right\rceil \geq 240$ bzw. $\frac{b}{20} > 239$ und damit b>4780 führt.

Wenn also mehr als **4780 Tupel** in der Relation Bänder gespeichert sind, dann ist der Nested–Loop–Join mit Indexzugriff im Vergleich zum Nested–Block–Join mit der Relation AUSLEIHEN in der äußeren Schleife kostengünstiger.