Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Prof. H.-J. Schek *A. Feßler, U. Röhm, H. Schuldt, R. Weber*

Informationssysteme IS-G SS 1997

Übung 8 Beispiellösung

Kostenschätzung einfacher Anfragen

In dieser Aufgabe sollen verschiedene Alternativen für den physischen Entwurf einer Oracle-Datenbank betrachtet und bewertet werden. Die Bewertung erfolgt dabei in zwei Schritten. Im ersten Schritt sollen Sie mittels Analyse des Speicherbedarfs die zu erwartenden Anfragekosten schätzen. Im zweiten Schritt können Sie Ihre Schätzungen mit Hilfe einiger Performance-Messungen überprüfen.

Daten

Wir betrachten eine Relation `R`, die zwei numerische Attribute A und B enthält. Die beiden Attribute haben jeweils die Länge 6 Bytes. In dieser Relation sind 50 000 Tupel gespeichert (rechnen Sie einfach mit *345 Datenbankseiten* à 2 KBytes). Die Attributwerte von A sind aus dem Intervall $[0 \dots 999]$ mit Gleichverteilung der Tupel über die Attributwerte. Die Attributwerte von B sind aus dem Intervall $[0 \dots 999]$, wobei 40 000 Tupel über das Intervall $[0 \dots 99]$ gleichverteilt und 10000 Tupel über das Intervall $[100 \dots 999]$ 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

Für den physischen Datenbankentwurf der Relation `R` stehen verschiedene Möglichkeiten offen. Im Rahmen dieser Übung sollen Sie die folgenden sechs näher betrachten:

1. Es ist lediglich die Relation `R` gespeichert (d.h. es gibt keinerlei Indexe).
2. Es existiert ein Index (B^*-Baum) über dem Attribut A (Relationenname `RA`).
3. Es existiert ein Index (B^*-Baum) über dem Attribut B (Relationenname `RB`).
4. Es existiert sowohl über A als auch über B ein Index (B^*-Baum, Relationenname `RAB`).
5. Es existiert ein „clustered" Index über A, d.h. die Relation ist bezüglich A physisch sortiert (Relationenname `RA$`).
6. Es existiert ein kombinierter Index (B^*-Baum) über A und B, d.h. die Attributwerte werden konkateniert und als ein Wert im Index gespeichert (Relationenname `RC`).

Alle Indexe ausser Alternative 5 seien mittelbar (indirekte B^*-Bäume). Nehmen Sie an, dass jede Relation und jeder Index jeweils einen eigenen Tablespace belegen, also getrennt gespeichert sind, wobei die Seitengrösse 2 KBytes sei mit 70% Füllgrad. Ein TID oder Zeiger im B^*-Baum habe eine Länge von 6 Bytes.

Messen der Kosten

Die berechneten Kosten können mit Messungen am Datenbanksystem ORACLE verglichen werden. Dazu steht Ihnen eine WWW-Schnittstelle zur Verfügung, welche auf einfache Art und Weise die Messung der Kosten erlaubt. Mit dem Kommando

```
netscape http://www-dbs.ethz.ch/is-cgi/uebung8.cgi
```

erhalten Sie diese graphische WWW-Oberfläche. Damit können Sie sowohl Ausführungspläne als auch die I/O- und CPU-Kosten für die verschiedenen Aufgaben ermitteln.

Aufgabe 1: Indexhöhen bestimmen

Ermitteln Sie zuerst die Höhe der Indexe und die nötige Anzahl Zugriffe für die Suche im Index. Verwenden Sie die Werte 10 und 500 für die Suche von A und B mit den Entwürfen und tragen Sie ihre Resulate in die untenstehende Tabelle ein. Vergleichen Sie ihre Messungen mit ORACLE, indem Sie die graphische Schnittstelle benutzen (beachten Sie die Angabe *Logical Read Operations*).

Berechnung

Die Berechnung der Indexhöhe für Entwurf `RA` *zeigt exemplarisch, wie man vorgehen muss: Es gibt 1000 verschiedene A-Werte, also enthält jeder Eintrag im Blatt des Indexes eine TID–Liste mit durchschnittlich 50 Zeigern auf die eigentlichen Datensätze. Die Länge der TID–Liste beträgt demnach* $50 \cdot 6 +$ *Länge einer Zahl. Somit ist ein Eintrag im Blatt ungefähr 300 Bytes gross. Da es 1000 verschiedene A-Werte geben kann und die Blätter 1400 Bytes Nutzdaten fassen (48 Bytes für Blattinformationen, 70% Füllgrad), gibt es ungefähr 250 Blattseiten. Oracle gibt in seiner Statistik (Tabelle* `USER_INDEXES`*) sogar 450 Blattseiten an. Schon 250 Blattseiten können nicht von einem Wurzelknoten verwaltet werden* ($1400/(6+6) = 116$ *Verweise je innerem B^*-Baum-Knoten), so dass es noch eine Zwischenstufe geben muss. Die Höhe des Baumes ist daher 3. Oracle gibt in der Statistik 2 an, was der Anzahl Schritte von der Wurzel zum Blatt entspricht (d.h. der Baum hat nach unserer Terminologie die Höhe 3).*

Die Suche nach $A = 10$ *oder* $A = 500$ *ist gleich aufwendig. Da die TID–Listen ˜ 300 Bytes gross sind, passen sie auf eine Seite, so dass mindestens 3 und maximal 4 Zugriffe nötig sind. Bei der Suche nach* $B = 10$ *resp.* $B = 500$ *muss man die unterschiedliche Verteilung beachten. Im ersten*

Fall enthält die TID–Liste im Schnitt 400 Einträge und ist somit ˜ 2400 Bytes lang. Somit müssen mindestens 2, maximal 3 Blattseiten gelesen werden. Beim zweiten Fall enthalten die TID–Listen durchschnittlich 11 Einträge und sind 66 Bytes lang. Dadurch ist mindestens 1 Blatt und höchstens 2 Blätter nötig. Beim Index für `RA$` *sind zwei Zugriffe für die Suche im Index und ein Zugriff für das Zählen der Tupel nötig. Der Index für Entwurf* `RC` *kann nur für die Suche von A verwendet werden (Range-Scan nötig); die Suche nach B-Werten kann nicht unterstützt werden. Dies hängt mit der Art der Konkatenation zusammen: Die Datenbank könnte die beiden Attribute z.B. mit der Funktion $C = 1000 * A + B$ verbinden. Es ist dann klar, dass nur die Suche nach A mittels Range-Scan von $1000 * A \ldots 1000 * A + 999$ möglich ist.*

Entwurf		Höhe	$A = 10$	$B = 10$	$A = 500$	$B = 500$
R:	kein Index	—	345	345	345	345
RA:	Index über A	3	3	345	3	345
RB:	Index über B	3	345	4	345	3
RAB:	Indexe über A und B	3	3	4	3	3
RA$:	Clustered Index	3	3	345	3	345
RC:	Kombinierter Index	3	3	345	3	345

Messung mit ORACLE

Entwurf		Höhe	$A = 10$	$B = 10$	$A = 500$	$B = 500$
R:	kein Index	—	348	348	348	348
RA:	Index über A	3	3	348	4	348
RB:	Index über B	3	348	7	348	3
RAB:	Indexe über A und B	3	3	7	4	3
RA$:	Clustered Index	3	3	1003	3	1003
RC:	Kombinierter Index	3	4	348	4	348

Auffallend ist, dass Oracle für einen Scan 3 Seiten mehr benötigt als wir berechnet haben. Dies lässt sich zum Teil damit erklären, dass ein Index-Scan durchgeführt wird.

Aufgabe 2: Berechnung der Kosten

Berechnen Sie für jede der drei Anfragen Q_1 bis Q_3 und jede Entwurfsalternative die geschätzten I/O-Kosten und tragen Sie die Werte in die untenstehende Tabelle ein. Mit der graphischen WWW–Schnittstelle können Sie als Hilfestellung die Ausführungspläne ORACLEs für die einzelnen Anfragen ermitteln.

Ein Problem stellt die ungleichmässige Verteilung von Attribut B dar. Wir überlegen uns zuerst, wieviele Tupel die Bedingungen $A = 321$ und $B = 50$ erfüllen. Daraus können wir die TID–Listenlänge bestimmen und die Anzahl der benötigten Blattseiten im Index. Sei n_A resp. n_B die Anzahl Treffer für die Bedingung $A = 321$ resp. $B = 50$ (TID hat Länge 6, Blattseiten fassen 1400 Bytes Nutzdaten). Die Höhe der Indexe ($NLevels(I_x)$) übernehmen wir aus der vorherigen Aufgabe. Die Kosten betragen dann:

$$\begin{array}{ll} \textit{Index A,B:} & NLevels(I_A) - 1 + \lceil \frac{n_a \cdot 6}{1400} \rceil + NLevels(I_B) - 1 + \lceil \frac{n_b \cdot 6}{1400} \rceil \\ \textit{Index A:} & NLevels(I_A) - 1 + \lceil \frac{n_a \cdot 6}{1400} \rceil + n_a \\ \textit{Index B:} & NLevels(I_B) - 1 + \lceil \frac{n_b \cdot 6}{1400} \rceil + n_b \end{array}$$

Falls wir einen Index über A und/oder B benutzen können, brauchen wir nicht auf die Datensätze selbst zuzugreifen, da wir nur die Anzahl Tupel (`COUNT(*)`*) bestimmen wollen. Beim Entwurf* `RA$` *muss nach der Indexsuche nur eine Seite geladen werden, da alle Tupel mit gleichem A-Wert auf einer Seite Platz finden. Die Messung zeigt, dass es 45 Tupel mit $A = 321$ und 408 Tupel mit $B = 50$ gibt. Damit ergeben sich die folgenden Kosten:*

Entwurf		Anfrage Q_1	Q_2	Q_3
R:	kein Index	345	345	345
RA:	Index über A	2+1+45	2+1+45	345
RB:	Index über B	345	345	345
RAB:	Indexe über A und B	2+1+2+2	2+1+45	345
RA$:	Clustered Index	2+1	2+1	345
RC:	Kombinierter Index	2+1	2+1	345

Aufgrund der ungleichen Verteilung des Attributs B kann ein Index über B für die gegebenen Anfragen nicht sinnvoll eingesetzt werden. Wenn kein Index über A vorhanden ist resp. für Abfrage Q_3 ist deshalb ein Relationenscan nötig.

Aufgabe 3: Messen der Kosten

Nachdem Sie die verschiedenen physischen Entwürfe analytisch begutachtet haben, sollen Sie die Richtigkeit Ihrer Überlegungen mit Hilfe von Leistungsmessungen überprüfen. Verwenden Sie wieder das WWW-Formular zu dieser Aufgabe, um die I/O-Kosten (*Logical Read Operations*) mit ORACLE zu messen. Tragen Sie die Resultate in die folgende Tabelle ein und vergleichen Sie die Werte mit Ihren Berechnungen. Wo gibt es Abweichungen? Können Sie diese Abweichungen erklären? Warum sind bei einigen Entwürfen die physischen I/O's selbst bei der ersten Anfrage kleiner als die logischen I/O's? Wie erklären sich die Werte des Entwurfs `RA$`?

Entwurf		Anfrage Q_1	Q_2	Q_3
R:	kein Index	348	348	348
RA:	Index über A	93	93	348
RB:	Index über B	819	348	819
RAB:	Indexe über A und B	77	93	819
RA$:	Clustered Index	3	3	1003
RC:	Kombinierter Index	3	3	348

Die gemessenen Resultate weichen z.T. sehr stark von den berechneten Kosten ab. Dies hängt mit Art und Weise zusammen, wie Oracle die logischen I/O's misst: Für jede Seitenfixierung wird der Zähler um eins erhöht. Oracle versucht zudem Seiten sofort wieder freizugeben, um die Deadlock-Gefahr zu verkleinern. Beim Entwurf `RA` *und Anfrage Q_1 wird dies sehr schön sichtbar: Die Bedingung $A = 321$ führt zu einer Blattseite mit einer TID–Liste der Länge 45. Um die weitere Bedingung $B = 50$ zu prüfen, muss jedem Verweis der TID–Liste nachgegangen werden. Dabei wird die Blattseite des Index jedesmal freigegeben und muss nach dem Besuch der Datenseite wieder fixiert werden (Bem.: die Transaktionssperren werden aber für die ganze Dauer der Anfrage gehalten). Im schlechtesten Fall müssen 93 Seiten von der Disk geladen werden (Blattseite wird immer gleich ausgelagert). Im Normalfall (keine Seite im Puffer, keine Verdrängung) muss nur die berechnete Anzahl Seiten (=48) geladen werden.*

Der sehr hohe Wert beim Scan durch den clustered Index von ***RA$*** *bei Anfrage Q_3 kommt daher, dass Oracle in einem Cluster je Seite* ***nur*** *solche Tupel mit gleichem Indexwert sammelt. Wir haben 1000 verschiedene A Werte, demzufolge ist die „geclusterte" Tabelle* ***RA$*** *1000 Seiten gross (zuzüglich der Indexseiten darüber), was einem um den Faktor 3 höheren Speicherbedarf als ohne Clustering entspricht.*

Die physischen I/O's geben die effektiv von der Disk geladene Anzahl Seiten an. Diese ist im Normalfall unter der logischen Anzahl, da viele Seiten bereits im Puffer sind (siehe obige Erklärung zu Entwurf ***RA*** *und Anfrage Q_1).*

Aufgabe 4: Algebraische Optimierung

Gegeben seien die beiden bereits vertrauten Relationen

`Produkte` (<u>PNr</u>, `Bez, Gewicht, Preis, Lagerort, Vorrat)` und
`Bestellungen` (<u>BestNr</u>, `Monat, Tag, KNr, PNr, Menge, Summe, Status)`

sowie die folgende View

`BestInfo (BestNr, PNr, Bez, Gewicht, Menge)`
$:= \pi$[BestNr,PNr, Bez, Gewicht, Menge] (`Produkte` ⋈ `Bestellungen`).

Es gelte die Integritätsbedingung:

π[PNr, Bez, Preis, Gewicht, Vorrat] (`Produkte`) =
π[PNr, Bez, Preis, Gewicht, Vorrat] (`Produkte` ⋈ `Bestellungen`),

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

Betrachten Sie nun die folgende Anfrage:

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

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

π[PNr, Bez] (σ[Gewicht > 100] (BestInfo))
$\equiv \pi$[PNr, Bez] (π[PNr, Bez, Gewicht] (σ[Gewicht > 100]
(π[BestNr,PNr, Bez, Gewicht, Menge] (Produkte ⋈ Bestellungen))))
nach Regel 4
$\equiv \pi$[PNr, Bez] (σ[Gewicht > 100] (π[PNr, Bez, Gewicht]
(π[BestNr,PNr, Bez, Gewicht, Menge] (Produkte ⋈ Bestellungen))))
nach Regel 1
$\equiv \pi$[PNr, Bez] (σ[Gewicht > 100] (π[PNr, Bez, Gewicht] (Produkte ⋈ Bestellungen)))
nach Regel 4
$\equiv \pi$[PNr, Bez] (σ[Gewicht > 100] (π[PNr, Bez, Gewicht]
π[PNr, Bez, Preis, Gew, Vorrat] (Produkte ⋈ Bestellungen)))
nach Regel 4
$\equiv \pi$[PNr, Bez] (σ[Gewicht > 100] (π[PNr, Bez, Gewicht]
π[PNr, Bez, Preis, Gew, Vorrat] (Produkte)))
nach vorgegebener Integritätsbedingung
$\equiv \pi$[PNr, Bez] (σ[Gewicht > 100] (π[PNr, Bez, Gewicht] (Produkte)))
nach Regel 4

Aufgabe 5: Zugriffsstrukturen (Aufgabe 3 von Übung 7)

Wir betrachten einen Teil unserer Musikdatenbank mit der folgenden Schemadefinition:

```
CREATE TABLE Disk
  (DiskID        NUMBER PRIMARY KEY NOT NULL,
   DiskTitel     VARCHAR(255));
CREATE TABLE Musikstueck
  (DiskID     NUMBER,
   StueckID   NUMBER,
   Titel      VARCHAR(255),
   PRIMARY KEY (DiskID, StueckID));
CREATE TABLE Person
  (PID           NUMBER PRIMARY KEY NOT NULL,
   Name          VARCHAR(40) NOT NULL,
   Nationalitaet CHAR(5));
CREATE TABLE Musikstueck_Autor
  (PID         NUMBER NOT NULL REFERENCES Person,
   DiskID      NUMBER NOT NULL,
   StueckID    NUMBER NOT NULL,
   Taetigkeit  VARCHAR(20) NOT NULL,
   FOREIGN KEY (DiskID, StueckID) REFERENCES Musikstueck);
```

Als Kardinalitäten und mittlere Tupellängen der verschiedenen Relationen nehmen wir an:

Disk	1000 Tupel à	100 Bytes
Musikstueck	10000 Tupel à	50 Bytes
Person	6000 Tupel à	40 Bytes
Musikstueck_Autor	30000 Tupel à	40 Bytes

Zur Effizienzsteigerung von Anfragen sind unter anderem die folgenden mittelbaren Indexe angelegt, die jeweils als B*-Baum implementiert sind:

```
CREATE UNIQUE INDEX DiskID ON Disk(DiskID);
CREATE INDEX DiskTitel ON Disk(DiskTitel);
CREATE INDEX AutorStueck ON Musikstueck_Autor (PID);
```

Nehmen Sie an, dass jede Relation und jeder Index jeweils einen eigenen Tablespace belegt, wobei die Seitengrösse 4 KBytes sei und der Seiten-Header die Länge 96 Bytes habe. Datenseiten seien zu 90 Prozent gefüllt, Indexseiten zu 70 Prozent. Der Datentyp NUMBER belege 10 Bytes, der Disktitel sei Schlüsselkandidat der Relation `Disk` und habe eine durchschnittliche Länge von 90 Bytes, ein TID oder Zeiger im B*-Baum habe die Länge 6 Bytes. Nehmen Sie zusätzlich an, dass alle Werte eines Fremdschlüsselattributes jeweils gleichhäufig vorkommen (Gleichverteilungsannahme).

Analysieren Sie die Speicherplatz- und Indexzugriffskosten dieser Datenbank. Dabei können Sie den Platzbedarf für die Slotlisten und für die Längenfelder des Datentyps VARCHAR vernachlässigen.

Platz je Datenseite: $(4096 - 96) \cdot 0{,}9 = 3600$ Bytes.
Platz je Indexseite: $(4096 - 96) \cdot 0{,}7 = 2800$ Bytes.

Speicherplatzbedarf für die Primärdaten:

Relation	**Tupel pro Seite**	**Anzahl Seiten**
Disk	$3600/100 = 36$	$\lceil 1000/36 \rceil = 28$
Musikstueck	$3600/50 = 72$	$\lceil 10000/72 \rceil = 139$
Person	$3600/40 = 90$	$\lceil 6000/90 \rceil = 67$
Musikstueck_Autor	$3600/40 = 90$	$\lceil 30000/90 \rceil = 334$

Speicherplatzbedarf für die Indexe:

Index	**Einträge pro**		**Anzahl**		**Total**
	Blattseite	**Innere Kn.**	**Blattseiten**	**Innere Kn.**	
DiskID	$2800/16 = 175$	$2800/16 = 175$	$\lceil 1000/175 \rceil = 6$	1	7
DiskTitel	$2800/96 = 29$	$2800/96 = 29$	$\lceil 1000/29 \rceil = 35$	1 + 2	38
AutorStueck	$2800/40^{*} = 70$	$2800/16 = 175$	$\lceil 6000^{\dagger}/70 \rceil = 86$	1	87

*Da jede Person im Durchschnitt zu 5 Musikstücken als Autor geführt wird, muss auch in den Blattseiten des `AutorStueck`-Indexes zu jeder `PID` eine TID-Liste mit mittlerer Länge 5 gespeichert werden.

†Es gibt nur 6000 verschiedene Schlüsselwerte (PIDs) in dieser Relation, deren TID-Listen in den Blattknoten gespeichert werden müssen.

Daraus ergeben sich folgende Zugriffskosten (Index- und Datenseitenzugriff):

DiskID	2 + 1 = 3 Seiten
DiskTitel	3 + 1 = 4 Seiten
AutorStueck	2 + 5 = 7 Seiten