Datenbanksysteme WS 2017/18
Prof. Dr.-Ing. Sebastian Michel
Manuel Hoffmann, Marc Dörr
TU Kaiserslautern, FB Informatik – Lehrgebiet Informationssysteme
http://dbis.informatik.uni-kl.de
Übungsblatt 2: Ausgabe 07.11.2017, Abgabe 13.11.2017 16:00 MEZ

Auf diesem Blatt betrachten wir ein Schema bestehend aus folgenden Relationen:

Filiale:	[FNR, Stadt, MNR]
Mitarbeiter:	[MNR, Name, Gehalt, FNR]
Artikel:	[ANR, Bezeichnung, Hersteller, Preis]
Inventar:	[ANR, FNR, Anzahl]
Rechnung:	[RNR, FNR, Kunde]
Posten:	[RNR, FNR, ANR, Anzahl]

`Filiale.MNR` ist Fremdschlüssel auf `Mitarbeiter.MNR` und bezeichnet den Filialleiter. `Mitarbeiter.FNR` ist Fremdschlüssel auf `Filiale.FNR` und gibt an, in welcher Filiale der Mitarbeiter angestellt ist. Die Inventartabelle gibt mit Fremdschlüsseln an, wie viele Arktikel (`ANR`) in welcher Filiale (`FNR`) vorrätig sind. Eine Rechnungen besteht aus mehreren Posten, wobei jeder Posten aus einem Fremdschlüssel auf Artikel (`ANR`) sowie einer Anzahl besteht.

Aufgabe 1: Abfragekosten (1 P.)

Beachten wir die Tabelle `Artikel` genauer. Es seien 500 000 Artikel gespeichert, die Artikelnummern sind von 1 bis 500 000 fortlaufend durchnummeriert. In eine Datenseite passen genau 4 Artikel-Tupel. Es gibt einen B^+-Baum als Primärindex mit Höhe h, die Blätter enthalten je 10 Einträge zusammen mit Verweisen auf Datenseiten.

a) Geben Sie für die Abfrage `select * from Artikel where ANR between 50000 and 80000` bei Verwendung des Primärindex die Kosten in Seitenzugriffen an.

b) Da häufig Artikel des selben Herstellers zusammen in Anfragen benötigt werden, entscheidet der Datenbankadministrator, die Tabelle `Artikel` nach `Hersteller` zu clustern. Weiterhin soll auf dem Primärschlüssel ein Index angelegt sein. Wie lässt sich dieser Index laut Vorlesung klassifizieren? Wie hoch sind jetzt die Kosten in Seitenzugriffen für die Anfrage auf Aufgabenteil a)?

Aufgabe 2: Zusammengesetzte Indexe (1 P.)

Das Unternehmen beschäftigt 20 000 Mitarbeiter. Jeweils zwei Mitarbeiter-Tupel passen in eine Datenseite. In Ermangelung besserer Schätzungen nehmen wir an, dass das Gehalt gleichverteilt Werte zwischen 20 000 und 100 000 annimmt. Außerdem gibt es zehn Filialen, in der jeweils gleich viele Mitarbeiter beschäftigt sind. Über der Spalte `Mitarbeiter` sind zwei sekundäre Composite-Key Indexe angelegt, einmal `(Gehalt, FNR)` und einmal `(FNR, Gehalt)`. Beide Indexe sind B^+-Bäume der Höhe h, deren Blätter je 5 Einträge zusammen mit Verweisen auf Datenseiten fassen.

a) Welcher der beiden Indexe ist für die Anfrage `select * from Mitarbeiter where FNR = 5 and Gehalt > 60000` günstiger? Geben Sie die erwartete Anzahl an Seitenzugriffen an.

b) Welcher dieser Indexe kann genutzt werden, um die folgenden Anfragen effizient zu beantworten? Erklären Sie kurz wie.

- $\delta(\pi_{Gehalt}(Mitarbeiter))$
- $\delta(\pi_{Gehalt,FNR}(Mitarbeiter))$
- $\sigma_{FNR<5}(Mitarbeiter) \cup \sigma_{Gehalt>30\,000}(Mitarbeiter)$

Aufgabe 3: Nested-Loop-Join und Hash-Join (1 P.)

Die Relationen `Rechnung` und `Posten` sollen gejoint werden. Es gibt 300 000 Rechnungen, und 1 000 000 Posten. Die Tupel beider Relationen sind jeweils eine Seite groß, der zur Verfügung stehender Join-Puffer fasst 156 Seiten, und Ergebnistupel des Joins sind auch eine Seite groß (etwa durch sofortige Projektion). Für diesen Join ist `Posten` innere und `Rechnung` äußere Relation.

a) Anstatt wie in der Vorlesung die Anzahl an einzelnen zugegriffenen Seiten zu schätzen, modellieren wir nun die Anzahl der *sequentiellen* Seitenzugriffe. Zum Beispiel zählt das Laden der ersten 156 Einträge einer Tabelle in den Puffer als ein sequentieller Seitenzugriff.

   Im Gegensatz zum Block-Nested-Loop Join aus der Vorlesung, bei dem der Puffer möglichst viele Tupel der äußeren Relation enthält, teilen wir nun den Puffer so auf, dass die eine Hälfte der Pufferseiten mit Tupeln der inneren und die andere Hälfte mit Tupeln der äußeren Relation belegt wird. Berechnen Sie die Anzahl der sequentiellen Seitenzugriffe, die so gebraucht wird, um den Join zu berechnen.

b) Nun soll ein Hash-Join durchgeführt werden, wobei die Hashfunktion für die Partitionierung $\mod k$ ist. Bestimmen Sie k, sodass der Joinpuffer optimal ausgenutzt wird und geben Sie die Zahl der benötigten sequentiellen Leseoperationen für den Join an. Ignorieren Sie Schreiboperationen nach dem Partitionieren und nehmen Sie an, dass alle relevanten Attribute ganze Zahlen ≥ 1 sind und gleichverteilt sind.

Aufgabe 4: Implementierung (1 P.)

Die folgende Aufgabe können Sie in einer Programmiersprache Ihrer Wahl anfertigen. Sie dürfen auch das auf der Vorlesungsseite zur Verfügung gestellte Java-Template benutzen. In diesem Template sind Hinweise zur Benutzung. Löschen Sie zunächst alle Postgres-Indexe auf den beiden Tabellen.

a) Implementieren Sie die Anfrage[1]

```
SELECT l_orderkey, l_shipdate, o_orderdate
FROM lineitem JOIN orders ON l_orderkey = o_orderkey
```

   wobei Sie zunächst die benötigten Werte aus der TPC-H-Datenbank in Listen in ihrem Programm laden. Dann führen Sie den Join einmal als einfachen Nested Loop Join und einmal als Index-based Nested Loop Join aus. Erstellen Sie als Index eine geeignete `HashMap`.

b) Messen Sie die Laufzeiten Ihrer beiden Implementierungen (inklusive der benötigten Zeit, um den Index zu erstellen!) und vergleichen Sie diese mit der Zeit, die Postgres benötigt. Wie erklären Sie sich die Unterschiede?

c) Welchen Join-Algorithmus hat Postgres gewählt? Bringen Sie Postgres dazu, einen Merge-Join durchzuführen und messen Sie die dafür benötigte Zeit.

[1] Im Beispielcode laden wir nur Tupel mit `orderkey < 50000`, Sie können aber gerne größere Zahlen ausprobieren.

Aufgabe 5: Aufgaben Erfinden (1 P.)

In dieser Aufgabe geht es darum, kreativ mit dem bisher behandelten Vorlesungsstoff zu arbeiten, Sie sollen nämlich eine eigene Übungsaufgabe erstellen. Dazu gibt es zwei Möglichkeiten:

a) Sie erstellen eine Aufgabe, die sich im Umfang und Anspruch an den bereits gesehenen Aufgaben orientiert, und die sich hinreichend von diesen Aufgaben unterscheidet. Das bedeutet, Sie sollen nicht nur eine Zahl ändern. Manchmal reicht es auch schon, einen kleinen Aspekt an einer Aufgabe zu ändern, damit bei der Lösung der Aufgabe trotzdem ein interessanter Unterschied zu sehen ist. In diesem Falle, erläutern Sie kurz, welchen Unterschied die Änderung nach sich zieht.

Oder

b) Erstellen Sie eine Aufgabe vom Typ “Dies und Das”. Dieser Typ besteht aus zehn einzelnen Fragen, die jeweils mit einem kurzen Satz beantwortet werden. Zum Beispiel: “Ist der Sort-Merge-Join immer schneller als der Nested-Loops-Join?”.

Wir planen, Ihnen geeignete Aufgaben zur Klausurvorbereitung bereitzustellen.