



**Übungen zur Vorlesung
“Database Systems and Modern CPU Architecture”
Sommersemester 2009**

Jan Rittinger (jan.rittinger@uni-tuebingen.de)

8. Übungsblatt

Ausgabe: 9. Juli 2009 · Besprechung: 15. Juni 2009

Aufgabe 1: Selektion in einem Array

(Abgabe)

Zu den wohl grundlegendsten Operationen in einem (relationalen) Datenbanksystem gehört die **Selektion** von Tupeln aus einer Tabelle.

Wir gehen hier von einer einspaltigen Tabelle `src` aus, die im Hauptspeicher als C-Array implementiert ist. In einen Ergebnis-Array `tgt` sollen alle diejenigen Tupel kopiert werden, deren Wert kleiner als ein gegebener Suchwert `limit` ist:

```
1  j = 0;
2  for (unsigned int i = 0; i < arrsize; i++)
3      if (src[i] < limit)
4          tgt[j++] = src[i];
```

Der Arbeitsaufwand dieser Routine hängt damit auch von der Selektivität des Suchwerts `limit` ab.

1. **Welches Laufzeitverhalten erwarten Sie in Abhängigkeit von der Selektivität des Suchwerts?**
2. **Implementieren Sie die Routine in C.** (Als Vorlage können Sie z. B. den Quellcode verwenden, den Sie zum ersten Übungsblatt erhalten hatten.)
3. **Welche tatsächliche Laufzeit beobachten Sie?** Welche Erklärung finden Sie für das beobachtete Verhalten?

Aufgabe 2: Mixed-Mode Selection

(Abgabe)

In der Vorlesung wurden auf Folie 43 Kapitel 5 verschiedene Auswertungsmodelle für konjunktive Prädikate vorgestellt (`&&`, `&`, `+=`). Diese lassen sich beliebig in einer *mixed-mode selection* kombinieren.

Erweitern Sie das Programm aus Aufgabe 1 um ein zweites Prädikat:

```
1  j = 0;
2  for (unsigned int i = 0; i < arrsize; i++)
3      if (src[i] < limit && src[i] % 2 == 0)
4          tgt[j++] = src[i];
```

und **ermitteln Sie welcher Selektions-Modus sich in Abhängigkeit der Selektivität des Suchwerts `limit` am Besten verhält.**

Aufgabe 3: Lightweight Compression

(Abgabe)

In der Vorlesung wurde ein Satz von Kompressionsverfahren vorgestellt, der sich durch besondere CPU-Effizienz auszeichnet. Insbesondere vermeiden die Algorithmen bedingte Sprunganweisungen, die zu hohen *branch misprediction penalties* führen würden.

Schreiben Sie ein Programm, das zu einem gegebenen Datensatz die Kompressionsrate des PFOR-Algorithmus (Folie 33, Kapitel 5) ermittelt. Verifizieren Sie dabei, welchen Einfluß die Bitbreite eines einzelnen Codes sowie der Referenzwert auf die Kompressionsrate hat.

Im letzten Übungsblatt haben wir bereits auf den TPC-H-Benchmark zurückgegriffen. Die zugehörigen Datensätze zeichnen sich durch eine recht realistische Datenverteilung aus, daher eignet sich dieser Benchmark auch für zu erwartende Kompressionsraten aus dem PFOR-Algorithmus. Ermitteln Sie mit Hilfe Ihres Programms die besten Kompressionsraten für die Spalten `partkey`, `linenumber` und `quantity` der `lineitem` Tabelle.