



**Übungen zur Vorlesung
“Datenbanksysteme II”**
Wintersemester 2008/2009
Manuel Mayr (manuel.mayr@uni-tuebingen.de)

1. Übungsblatt

Ausgabe: 16. Oktober 2008 · Besprechung: 23. Oktober 2008

Aufgabe 1: Access Time

(6 Punkte)

Bereits in der Vorlesung haben wir gesehen, dass selbst moderne Festplatten einen “Flaschenhals” in der Architektur eines Datenbanksystems darstellen.

In der folgenden Aufgabe sollen Sie eine aktuelle *server-class* Festplatte und eine etwas betagtere Festplatte für “Normalsterbliche” vergleichen.

Ultrastar 15K450^a

Average Seek Time = 3.6 ms
Rotational Speed = 15,000 RPM
Average Transferrate = 150 MB/s

^a<http://hitachigst.com/portal/site/en/products/ultrastar/15K450/>

WD Special Edition Caviar 200GB^a

Average Seek Time = 8.9 ms
Rotational Speed = 7,200 RPM
Average Transferrate = 100 MB/s

^ahttp://stores.tomshardware.com/search_techsspecs_full.php/masterid=605001/st=product_tab

1. Nehmen Sie an, dass sie jeweils 8 KB Blöcke lesen. Berechnen Sie jeweils für beide Festplatten, wie lange es im Mittel dauert
 - sequentiell 10,000 Blöcke zu lesen?
 - zufällig 10,000 Blöcke zu lesen?
2. Vergleichen Sie jeweils die Zeiten für ein sequentielles und zufälliges Auslesen der Festplatten. Wie kommt es zustande, dass sequentielles Lesen so viel schneller als zufälliges ist?

Aufgabe 2: RAID-Systeme

(6 Punkte)

Die Berechnung der einfachen Parität erlaubt die Erkennung eines einzelnen Bitfehlers innerhalb einer Binärsequenz. Die Korrektur eines Fehlers innerhalb einer Binärsequenz benötigt jedoch mehr Informationen, da die Position des “gekippten” Bits identifiziert werden muss, wenn es korrigiert werden soll. Mit einem einzelnen Paritätsbit ist dies nicht möglich, da jeder Bitfehler in jeder denkbaren Position innerhalb der Binärsequenz genau diesselbe Information erzeugt, nämlich einen Paritätsfehler.

Werden mehrere Paritätsbits für eine Binärsequenz gespeichert und werden diese Paritätsbits so angeordnet, so dass unterschiedliche gekippte Bits unterschiedliche Fehlerergebnisse erzeugen, so kann das gekippte Bit erkannt werden. In einer 7-Bit Nachricht z.B. können 7 mögliche einzelne Bitfehler auftreten, d.h. 3 Paritätsbits genügen, dass einerseits der Fehler erkannt wird und andererseits herausgefunden werden kann, welches Bit gekippt ist.

Das *Hamming-Code-Verfahren* ist eine Technik, die ein einzelnes gekipptes Bit innerhalb einer Bitsequenz erkennen und korrigieren kann. Um Datenfehler zu erkennen und zu korrigieren, berechnet das Verfahren sogenannte "code words" anhand folgendem Algorithmus:

- i. Markiere alle Bitpositionen, welche gültige Zweier-Potenzen sind als Paritätsbits (Positionen 1, 2, 4, 8, ...).
- ii. Alle anderen Bitpositionen stehen für die eigentlichen Daten zur Verfügung (Positionen 3, 5, 6, 7, ...)
- iii. Jedes Paritätsbit berechnet die Parität für ausgewählte Datenbits des Codewortes. Die Position des Paritätsbits bestimmt die Sequenz von Bits, die dabei alternativ geprüft bzw. übersprungen wird.
 - Pos 1: überprüfe 1 Bit, überspringe 1 Bit, überprüfe 1 Bit, etc. (1, 3, 5, 7, 9, 11, ...)
 - Pos 2: überprüfe 2 Bit, überspringe 2 Bit, überprüfe 2 Bit, etc. (2, 3, 6, 7, 10, 11, ...)
 - Pos 4: überprüfe 4 Bit, überspringe 4 Bit, überprüfe 4 Bit, etc. (4, 5, 6, 7, 12, 13, ...)
 - Pos 8: ...
 - Pos 16: ...
- iv. Setze das Paritätsbit auf 1, wenn die Anzahl von Einsen in den jeweiligen Bitpositionen ungerade ist. Setze das Paritätsbit auf 0, wenn die Anzahl von Einsen in den jeweiligen Bitpositionen gerade ist.

8-Bit Binärsequenz: 10011010

Erzeugen des Codewortes, indem Platzhalter für die Paritätsbits eingefügt werden: `1_001_1010`

Berechnung der Parität für jedes Paritätsbit (das Symbol ? repräsentiert die zu setzende Bitposition):

- o Position 1 überprüft Bits 1, 3, 5, 7, 9, 11: `?_1_001_1010`. Gerade Parität, d.h., Paritätsbit 1 wird auf 0 gesetzt: `0_1_001_1010`.
 - o Position 2 überprüft Bits 2, 3, 6, 7, 10, 11: `0?1_001_1010`. Ungerade Parität, d.h. Paritätsbit 2 wird auf 1 gesetzt: `011_001_1010`.
 - o Position 4 überprüft Bits 4, 5, 6, 7, 12: `011?001_1010`. Ungerade Parität, d.h. Paritätsbit 4 wird auf 1 gesetzt: `0111001_1010`.
 - o Position 8 überprüft Bits 8,9,10,11,12: `0111001?1010`. Gerade Parität, d.h. Paritätsbit 4 wird auf 0 gesetzt: `011100101010`.
- Codewort: 011100101010

Das Erkennen und die Korrektur von Bitfehlern erfolgt über die Validierungsmatrix V . Für diese Matrix gilt, dass jeder j -te Spaltenvektor jeweils die Binärdarstellung der Zahl $j \in 1, \dots, n$, wobei $n \in \mathbb{N}$ die Länge des Codewortes repräsentiert.

Für das obige Beispiel mit 8 Daten- und 4 Paritätsbits ergibt sich die folgenden Matrix:

$$V = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Die Position eines gekippten Bits innerhalb eines fehlerhaften Codewortes c erhält man dadurch, indem man das fehlerhafte Codewort mit der Validierungsmatrix V multipliziert.

Das nachfolgende Beispiel illustriert die Fehlererkennung:

Für die 8-Bit Binärsequenz 10011010 ist das zugehörige "korrekte" Codewort 011100101010. Nehmen Sie an, dass im gespeicherten Codewort c ein Bit gekippt ist und dieses daher wie folgt aussieht: 011100101110. Bilden wir nun das Produkt von c und V . so erhalten wir den Spaltenvektor e , welcher die Position des gekippten Bits im Codewort anzeigt.

$$e = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Der Ergebnisvektor indiziert, dass das 10-te Bit fehlerhaft ist und von 1 auf 0 gesetzt werden muss.

Nach der Einführung nun die eigentlichen Aufgabenstellungen:

- Bestimmen Sie den Hamming Code für die beiden 4-Bitsequenzen (nibbles) 0110 und 1010.
- Gegeben sind folgende Hamming-Codewörter, welche möglicherweise jeweils ein fehlerhaftes Bit beinhalten können:
 - 0100011
 - 1111111
 - 0110010

Überprüfen Sie diese Codewörter hinsichtlich ihrer Korrektheit und gebt gegebenenfalls die Position des fehlerhaften Bits an.

Aufgabe 3: BÉLÁDY'S ANOMALIE

(8 Punkte)

"Eine Vergrößerung des Bufferpools (Anzahl der verfügbaren Seiten) verbessert in jedem Fall die Effektivität der Seitenersetzungsstrategie, und minimiert die Anzahl der Seitenfehler."

Im Jahr 1969 zeigte der ungarische Informatiker Lázló BÉLÁDY, dass es sehr wohl möglich ist, trotz einer Vergrößerung des Buffers, mehr Seitenfehler auszulösen als bei einem kleineren Buffer. Dieses Phänomen wird in der Literatur als *BÉLÁDY'S ANOMALIE* bezeichnet.

- Zählen Sie die Anzahl der Seitenersetzungsfehler FIFO und LRU jeweils für eine Buffergröße von drei bzw. vier Seiten miteinander, wenn die verfügbaren Seiten in der folgenden Reihenfolge vom Datenbanksystem *gepinnt* werden:

$$p_1, p_2, p_3, p_4, p_1, p_2, p_5, p_1, p_2, p_3, p_4, p_5$$

- Welche Eigenschaften muss ein Seitenersetzungsalgorithmus haben, damit die Anomalie nicht auftritt?