

Prof. Dr. T. Härder
 Fachbereich Informatik
 Arbeitsgruppe Datenbanken und Informationssysteme
 Universität Kaiserslautern

Übungsblatt 2 – Lösungsvorschläge

für die freiwillige Übung

Unterlagen zur Vorlesung: „www.dvs.informatik.uni-kl.de/courses/DBSREAL/“

Aufgabe 1: Indirekte Seitenadressierung, Schattenspeicher- und Zusatzdateikonzept

81

Eine Datenbank bestehe aus zwei Segmenten mit jeweils 8 Seiten, zu deren Ab-
 speicherung insgesamt 32 Blöcke zur Verfügung stehen. Auf dieser DB laufen zeit-
 lich überlappt zwei Transaktionen ab, die Seiten der DB in der nachfolgend aufge-
 führten zeitlichen Sequenz verändern:

T1: P12 P15 P24 P17 EOT

T2: P21 P25 P16 P27 EOT

P_{ij} steht für Seite j in Segment i

EOT bezeichnet das Ende der Transaktion

Machen Sie sich an diesem Beispiel die Funktionsweise der indirekten Adressie-
 rung, des Schattenspeicher- und des Zusatzdateikonzeptes klar. Vor Beginn von T1
 seien beide Segmente geschlossen.

Bei indirekter Adressierung sei folgende Zuordnung der Seiten (P) zu Blöcken (B)
 gegeben:

P11 P12 P13 P14 P15 P16 P17 P18 P21 P22 P23 P24 P25 P26 P27 P28
 B1 B3 B6 B5 B7 B13 B9 B10 B15 B18 B19 B17 B21 B28 B24 B26

a) Bestimmen Sie den Aufbau der Seitenzuordnungstabellen bei indirekter Sei-
 tenadressierung.

Seg. 1: V1

Seg. 2: V2

1	3	6	5	7	13	9	10
---	---	---	---	---	----	---	----

15	18	19	17	21	28	24	26
----	----	----	----	----	----	----	----

Seitentabellen

1		8		16		24		32		Block						
11	12	14	13	15	17	18	16	21	24	22	23	25	27	28	26	Datei

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

1		8		16		24		32																								
1	0	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	1	1	0	1	0	0	1	0	1	0	1	0	0	0	0	Freispeicher- tabelle

nach EOT2:

1	8	16	24	32																						
11	12	27	21	14	13	15	15	17	18	25	24	16	16	21	17	24	22	23	25	27	28	26				

Datei

V11	1	3	6	5	7	13	9	10	V21	4	18	19	12	11	28	24	26
V10	1	2	6	5	8	14	16	10	V20	4	18	19	12	11	28	3	26

Seitentabellen

1	8	16	24	32	Freispeicher-
tabellen					

CMAP

1	1	1	1	1	1	0	1	0	1	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

MAP0

1	1	0	1	1	1	0	1	0	1	1	1	0	1	0	1	0	1	1	0	0	0	0	1	0	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

MAP1

STATUS	0
	0
MAPSWITCH	0

Aktionen:

- vor Änderung von P27 wurde V21 aktualisiert, und CMAP an MAP1 angepaßt
- nach Sicherungspunkt wurde MAP0 erstellt, gesichert und auf MAP0 umgestellt
- Vli sind hier nicht betroffen

d) Was muß am Schattenspeicherkonzept geändert werden, um transaktionsbezogene Sicherungspunkte implementieren zu können? Vollziehen Sie einen solchen Sicherungspunkt bei EOT am Ende von T1 am Beispiel nach.

Im wesentlichen: Einsatz transaktionsspezifischer Tabellen

(siehe Datenbanksysteme - Konzepte und Techniken der Realisierung, S. 101, genauer in Härder T., Implementierung von Datenbanksystemen, Carl Hanser, München, 1978, oder Härder, T., Reuter, A.: Optimization of Logging and Recovery in a Database System, in: Data Base Architecture, Proc. IFIP TC-2 Working Conference, June 1979, Venice, Italy, Bracchi, G. and Nijssen, G.M. (eds.), North Holland Publ. Comp., 1979, pp. 151-168.)

Aufgabe 2: Bloom-Filter

183

Mit Hilfe des Bloom-Filters kann beim Zusatzdatei-Verfahren einfach entschieden werden, ob eine Seite verändert worden sein kann und deshalb möglicherweise in der Zusatzdatei steht, oder ob dieses nicht der Fall sein kann.

Gegeben seien ein Bitvektor der Länge 8 und eine Hashfunktion $h(x)$, welche die Positionen der 1-en liefert, die bei folgender Funktion $g(x)$ gesetzt werden:

$$g(x) = (\text{Binärdarstellung von } x) \text{ XOR } (01010101).$$

Die Seiten mit den Schlüsselwerten 31, 53 und 62 werden verändert. Anschließend sollen die Seiten mit den Schlüsselwerten 53, 93 und 124 gelesen werden.

Welche Ergebnisse liefert der Bloom-Filter bei den Leseoperationen?

Nach der Initialisierung sieht der Vektor folgendermaßen aus: (0000 0000).

$g(31) = (0100 1010)$ --> der Vektor nach Änderung des Satzes: (0100 1010)

$g(53) = (0110 0000)$ --> der Vektor nach Änderung des Satzes: (0110 1010)

$g(62) = (0110 1011)$ --> der Vektor nach Änderung des Satzes: (0110 1011)

Beim Lesen von Satz 53 liefert der Filter natürlich „VIELLEICHT“,

beim Lesen von Satz 93 ($g(93)=(0110 1001)$) liefert der Filter ebenfalls „VIELLEICHT“,

beim Lesen von Satz 124 ($g(124)=(0010 1001)$) liefert der Filter „VIELLEICHT“.

Warum ist die angegebene Hashfunktion für den Bloom-Filter ungeeignet und welche Eigenschaften muß eine bessere Hashfunktion haben?

Ein geeigneter Bloom-Filter sollte für jeden Satz ungefähr gleich viele Positionen im Vektor adressieren, was hier nicht der Fall ist.

Aufgabe 3: Lokalität und Sequentialität

173

Eine Transaktion erzeuge folgenden Referenzstring:

AABBCEDABGHAAGGHHIIKKLKLKLMABABKLKLKLFMFAGHI

Beachten Sie hierbei, daß als sequentieller Reihenfolge angenommen wird:
ABCDEFGHIJKLMNO

Berechnen Sie folgende Größen:

- Die aktuelle Lokalität zu den Zeitpunkten $t = 6, 18, 28$ und der Fenstergröße 6.

$$AL(t,6) = W(t, 6) / 6$$

$$AL(6,6) = 4/6$$

$$AL(18,6) = 4/6$$

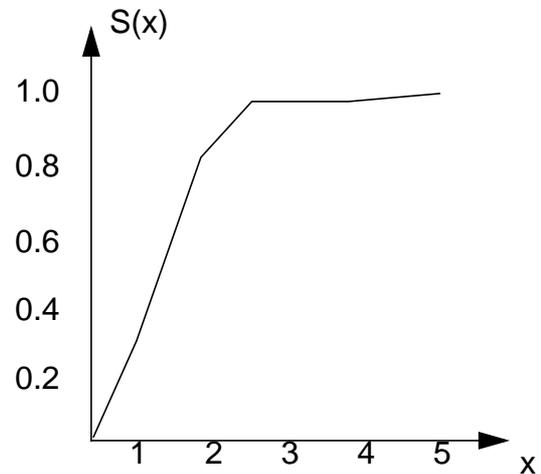
$$AL(28,6) = 2/6$$

- Die durchschnittliche Lokalität.

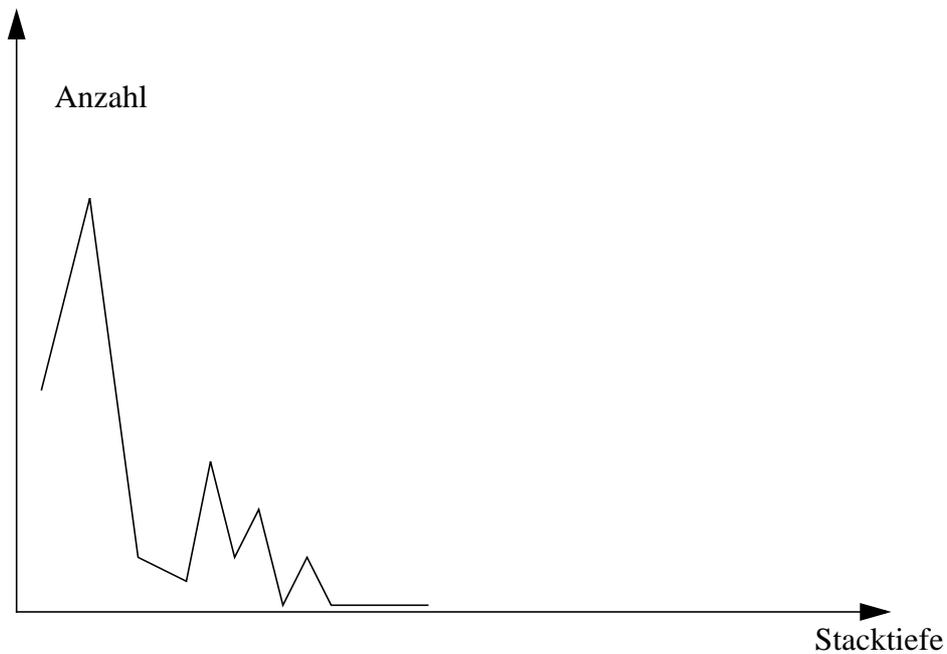
$$L(6) = ((4+5+5+5+6+6+5+\dots)/41) / 6 = (158 / 41) / 6 = 0,642$$

- Die sequentiellen Zugriffsfolgen sowie das Maß der Sequentialität der Transaktion.

seq. Folge	Länge
AABBC	3
E	1
D	1
AB	2
GH	2
AA	1
GGHHIIKKL	5
KL	2
KL	2
KLM	3
AB	2
AB	2
KL	2
KL	2
KL	2
F	1
M	1
F	1
A	1
GHI	3



- Die LRU-Stacktiefenverteilung.



Diskutieren Sie die dynamischen Zuordnungsentscheidungen von Pufferseiten für diese Transaktion. Wie sind die vorgestellten Entscheidungsgrößen effizient im DBMS zu ermitteln.

Aufgabe 4: Suche im DB-Puffer

Im Skript, 3 - 15, sind mehrere Verfahren zur Unterstützung der schnellen Suche einer DB-Seite im DB-Puffer skizziert. Analysieren Sie die folgenden Verfahren im Hinblick auf Wartungs- und Suchkosten:

1. unsortierte Tabelle
2. sortierte Tabelle
3. Tabelle mit verketteten Einträgen (LRU-Reihenfolge)
4. AVL-Baum
5. Hash-Tabelle mit Überlaufketten

Annahmen:

Bei jeweils 10 Suchvorgängen wird die Seite neunmal gefunden, einmal wird sie nicht gefunden (Fehlseitenbedingung), so daß eine Ersetzung erfolgen muß.

Das i -te Verfahren habe dabei m_i Kosteneinheiten w an Wartungskosten und pro Suchvorgang s_i Kosteneinheiten c an Suchkosten.

Jede Tabelle oder Datenstruktur habe $n = 2^{10}$ Einträge. Zur Aufrechterhaltung der Tabellensortierung nehmen wir an, daß $n/2$ Einträge verschoben werden müssen zu $n/2$ Kosteneinheiten w .

Bei LRU-Reihenfolge nehmen wir an, daß eine Seite, die sich im Puffer befindet, im Mittel mit $n/10$ Kosteneinheiten c gefunden wird. Beim Hash-Verfahren sollen im Mittel 1,5 Vergleiche anfallen. Bestimmen Sie die Kosten C_i , die für jeweils 10 Suchvorgänge mit einer Ersetzung anfallen.

Schätzen Sie die anfallenden Kostenanteile in geeigneter Weise ab!

Welches Verfahren gewinnt, wenn $w = 5 * c$ gesetzt wird?

1. unsortierte Tabelle

$$C_1 = 9 * c * n/2 + 1 * c * n + w * 1 = (11 * 512 + 10) * c = 5637 * c$$

2. sortierte Tabelle

$$\begin{aligned} C_2 &= 10 * c * \log_2(n) + 1 * w * n/2 = 10 * c * \log_2(2^{10}) + w * 2^{10}/2 \\ &= (100 + 2560) * c = 2660 * c \end{aligned}$$

3. Tabelle mit verketteten Einträgen (LRU-Reihenfolge)

$$C_3 = 9 * c * n/10 + 1 * c * n + 1 * w * 2 = 1440 * c$$

4. AVL-Baum

$$C_4 = 10 * c * 1,44 \log_2(n) + 1 * w * 1,44 \log_2(n) = 10 * c * 1,44 * 10 + w * 1,44 * 10 \\ = 216 * c$$

5. Hash-Tabelle mit Überlaufketten

$$C_5 = 10 * c * 1,5 + 1 * w * 2 = 25 * c$$

Es gibt also einen klaren Sieger. Die Eignungsreihenfolge, die das Ergebnis festlegt (C_5, C_4, C_3, C_2, C_1), ergibt sich nicht nur für die speziellen Parameter; sie scheint ganz allgemein typisch und praxisnah.

Ändern sich diese prinzipiellen Aussagen, wenn die mittlere Lokalität nicht 90% ($10/1$), sondern $x\%$ ($(x+y)/y$) beträgt? Diskutieren Sie $x = y$, also $x=y=5$, um den direkten Vergleich zu ermöglichen.

1. unsortierte Tabelle

$$C_1 = x * c * n/2 + y * c * n + w * y = (11 * 512 + 10) * c = 7705 * c$$

2. sortierte Tabelle

$$C_2 = (x+y) * c * \log_2(n) + y * w * n/2 = 10 * c * \log_2(2^{10}) + 5 * w * 2^{10}/2 \\ = (100 + 12800) * c = 12900 * c$$

3. Tabelle mit verketteten Einträgen (LRU-Reihenfolge)

$$C_3 = x * c * n/10 + y * c * n + y * w * 2 = 5682 * c$$

4. AVL-Baum

$$C_4 = (x+y) * c * 1,44 \log_2(n) + y * w * 1,44 \log_2(n) \\ = 10 * c * 1,44 * 10 + 5 * w * 1,44 * 10 = 504 * c$$

5. Hash-Tabelle mit Überlaufketten

$$C_5 = (x+y) * c * 1,5 + y * w * 2 = 65 * c$$

Es ergibt sich die Reihenfolge (C_5, C_4, C_3, C_1, C_2)! Allerdings erscheint die Kosteneinheit w für das Verschieben in Tabellen zu hoch gewählt zu sein!