Prof. Dr.-Ing. Dr. h. c. T. Härder
Fachbereich Informatik
Arbeitsgruppe Datenbanken und Informationssysteme
Universität Kaiserslautern

Übungsblatt 5 – Lösungsvorschläge

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

Aufgabe 1: Direkte und Indirekte Satzadressierung 12

a) Vergleichen Sie den Speicherplatzbedarf für folgende Adressierungsprinzipien:

1. logische Byte-Adresse
2. TID-Konzept
3. DBK-Konzept
4. DBK/PPP-Konzept.

Die Ausprägungen eines zu adressierenden Satztyps seien jeweils auf ein Segment der Größe 2^{32} Byte beschränkt. Es können mehrere Satztypen in einem Segment gespeichert sein. Welche Faktoren werden durch die einzelnen Adressierungskonzepte bei gegebener Pointerlänge begrenzt:

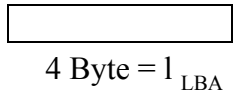
N_{REC} : Anzahl der Sätze,
S_l : logische Satzlänge,
L_s : Seitengröße,
m_k : Anzahl der Seiten (im Segment)?

Wie hoch ist der gesamte Speicherplatzbedarf für die Adressierung eines Satzes, wenn n verschiedene Pointer auf den Satz zeigen? Die Anzahl der Überläufer bei TID-Konzept liege bei 10%.

Mögliche Faktoren, die begrenzen:

- N_{rec} : Anzahl der Sätze
- S_l : logische Satzlänge — $S_l = 128 = 2^7$
- L_s : Seitenlänge — $L_s = 2^{12} = 4$ KByte
- m : Anzahl Seiten [im Segment]

a) **1) logische Byteadresse hat Länge l_{LBA}**



Vor jedem Satz steht ein Satzkennzeichen der Länge $l_T = l_{SKZ}$ Bytes (4 Byte)

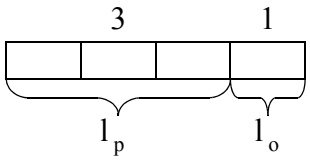
$$m = \frac{2^{lLBA}}{L_S} = \frac{2^{32}}{2^{12}} = 2^{20}$$ Seiten [im Segment]

$$N_{rec} = \frac{2^{lLBA}}{S_l} = \frac{2^{32}}{2^7} = 2^{25}$$ Datensätze

Aufwand für n Adressierungen: $n \bullet l_{LBA} + l_{SKZ} = (4n + 4)$ Bytes

$n = 10 : 44$ Bytes

2) TID



Pointer auf Seite Offset in Seite

$m = 2^{lp}$ Seiten [im Segment]

$N_s = 2^{lo}$ Datensätze / Seite; s wird aber auch durch Seitengröße und Satzlänge begrenzt

$$N_s = \frac{L_s}{S_l}$$

$$\Rightarrow 2^{l0} \geq \frac{L_s}{S_l}$$

$N_{rec} = 2^{lp} \bullet N_s$ Datensätze

Aufwand : $s = n (l_p + l_o) + l_{slot} + l_{SKZ}$

$P_{overflow}$: 10 % $+ P_{ov} \bullet n \bullet (l_p + l_o + l_{slot})$

$= (4n + 6 + 0.1 \bullet 6)$ Byte

L_{slot} : 2 Byte $n = 10 \rightarrow 46{,}6$ Byte

→ Behälter

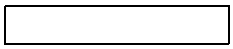
→ Seiteninterner Verweis

3) DB-Key / PP

POINTER 

l_T l_S
1 Byte 3 Byte

Typkennzeichen ein Byte
im DBK und Pageheader

DBTT
PP 

l_{pp} = 3 Byte = 24 Bit

Seite 

l_T l_S l_{RBA}
1 Byte 3 Byte 2 Byte

$m = 2^{l_{pp}} = 2^{24}$ Seiten

$N_{rec} = 2^{l_s} = 2^{24}$ Datenseiten je Typ bei 2^8 Typen

Aufwand: $s = n (l_t + l_s) + l_{pp} + l_T + l_s + l_{RBA}$
$= 4n + 3 + 4 + 2$ Byte
$= 4n + 9$ Byte

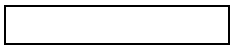
für $n = 10 \rightarrow$ 49 Byte

4) DB-Key / PP / PPP

POINTER 

l_T l_S l_{PP}

1 3 3 Byte

DBTT
PP 

l_{pp} = 3 Byte

Seite 

l_T l_S l_{RBA}
1 Byte 3 Byte 2 Byte

$m = 2^{l_{pp}} = 2^{24}$ Seiten

$N_{rec} = 2^{l_s} = 2^{24}$ Datensätze

Aufwand: $s = n (l_t + l_s + l_{pp}) + l_{pp} + l_T + l_s + l_{RBA}$
$= 7n + 9$ Byte

für $n = 10 \rightarrow$ 79 Byte

b) Schätzen sie den Zugriffsaufwand bei den verschiedenen Adressierungskonzepten ab. Wieviele physische E/A-Zugriffe sind bei einer Folge wahlfreier Zugriffe über einen Indexbaum mit 3 Stufen im Mittel zu erwarten, bis der gesuchte Datensatz gefunden ist,

1. beim TID-Konzept
2. beim DBK-Konzept
3. beim DBK/PPP-Konzept?

Der DB-Puffer besitze 40 Seitenrahmen zu 4 K Byte; die Ersetzungsstrategie sei LRU. Es seien $N_{REC}=10^5$ Datensätze über 10^4 Seiten verteilt. Der Indexbaum habe neben der Wurzel 9 Seiten auf der 2. Indexstufe und 1000 Seiten auf der 3. Indexstufe. Die Zuordnungstabelle bestehe aus Seiten der Größe 4 KB und besitze PP-Einträge der Länge 4 Byte. Die Anzahl der Überläufer sowie die Anzahl der falschen PPPs sei 10%.

Vergleichen sie die Anzahl der Zugriffe, wenn über die interne logische Adresse (TID, DB-Key) direkt zugegriffen wird.

b) Annahme :Wurzel + 2. Indexstufe bleiben durch LRU im DB-Puffer, bei anderen Seitentypen verhält sich LRU wie FIFO. 10 Pufferrahmen durch Wurzel und 2. Indexstufe ständig belegt. Im Schnitt 15 Indexseiten der 3. Stufe und 15 Datenseiten im Puffer.

Begründung:

Die Zugriffe erfolgen in der Reihenfolge 1., 2., 3. Indexseite, Datenseite ⇒ Jeder 4. Zugriff referenziert die Wurzel, die dadurch nicht verdrängt wird. Nach 9 Zugriffen sind 1 + 9 + 9 + 9 = 28 Rahmen belegt. Beim 10. Zugriff wird wieder die Wurzel referenziert. Die Seite der 2. Indexstufe befindet sich auch schon im Puffer. Bei gleichmäßiger Verteilung wird also auch die 2. Stufe des Index-Baumes immer im Puffer gefunden.

Wahlfreier Zugriff → Anzahl der Zugriffe, wenn Seite nicht im Puffer steht;

$\frac{\text{Anz. Rahmen}}{\text{Anz. Seiten}}$ gibt an, wie wahrscheinlich es ist, dass eine Seite im Puffer zu finden ist.

Wie häufig findet man die Seite nicht: $1 - \frac{\text{Anz. Rahmen}}{\text{Anz. Seiten}}$

1) TID

1.1 Index

$$N_{TID} = \underbrace{\left(1 - \frac{15}{1000}\right)}_{\text{Index}} + \underbrace{\left(1 - \frac{15}{10000}\right)}_{\text{Daten}} + \underbrace{0{,}1 \cdot \left(1 - \frac{15}{10000}\right)}_{\text{Überlauf}}$$

$$= 2{,}08335$$

1**.2 Direktzugriff über TID ohne Index** (40 Datenseiten):

$$N_{direkt} = 1{,}1 \cdot \left(1 - \frac{40}{10000}\right) = 1{,}0956$$

2) DB-Key / PP-Konzept

DBTT = 4096 / 4 = 1028 Einträge / Seite ⇒ $N_{REC}/1028 \approx 100$ DBTT-Seiten

2.1 Index

$$N_{PP} = \underbrace{\left(1 - \frac{10}{100}\right)}_{\text{DBTT}} + \underbrace{\left(1 - \frac{10}{1000}\right)}_{\text{Index}} + \underbrace{\left(1 - \frac{10}{(10000)}\right)}_{\text{Daten}}$$

10 Seiten : DBTT
10 Seiten : Index
10 Seiten : Daten

$$= 2{,}889$$

2.2 Direktzugriff ohne Indexstruktur

$$N_{direkt} = \underbrace{\left(1 - \frac{20}{100}\right)}_{\text{DBTT}} + \underbrace{\left(1 - \frac{20}{10000}\right)}_{\text{Daten}} = 1{,}798$$

20 DBTT-Seiten
20 Daten-Seiten

3) DB-Key / PP / PPP-Konzept

3.1 mit Index (2 DBTT-Seiten (nur in 10% benutzt!), 14 Index-Seiten, 14 Datenseiten)

$$N_{PPP} = \underbrace{\left(1 - \frac{14}{1000}\right)}_{\text{Index}} + \underbrace{\left(1 - \frac{14}{10000}\right)}_{\text{Daten}} + 0{,}1 \cdot \underbrace{\left(\underbrace{1 - \frac{2}{100}}_{\text{DBTT}} + \underbrace{1 - \frac{14}{10000}}_{\text{Daten}} \right)}_{\text{Fehler}}$$

$$= 2{,}182$$

3.2 ohne Index (4 DBTT-Seiten, 36 Datenseiten)

$$N_{direkt} = \left(1 - \frac{36}{10000}\right) + 0{,}1 \cdot \left(1 - \frac{4}{100} + 1 - \frac{36}{10000}\right) = 1{,}192$$

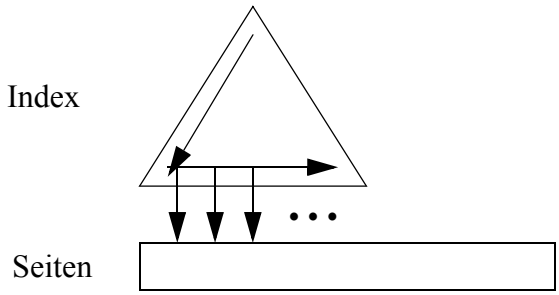
c) Ermitteln Sie die Anzahl der physischen Zugriffe bei der fortlaufenden Verarbeitung aller Datensätze über eine Indexstruktur, wenn mit

1. dem TID-Konzept
2. dem DBK-Konzept
3. dem DBK/PPP-Konzept

adressiert wird. Welche Ergebnisse erhalten Sie unter den Annahmen von b) im *best case* (Clusterbildung) und im *worst case*?

Die Verarbeitung erfolgt derart, dass die Wurzel und die Seite ganz links auf der 2. Indexstufe gelesen werden. Danach werden sequentiell alle Seiten der 3. Indexstufe gelesen und jeweils die

zugehörigen Datenseiten.



1) **TID**: Wurzel 2. Index 3. Index Datenseite

best case: 1 + 1 + 1000 + $10000 + a_1 \times N_{REC} \times \left(1 - \frac{38}{10000}\right)$
= **20964**

worst case: 1 + 1 + 1000 + $N_{REC} \cdot \left(1 - \frac{39}{10000}\right) + \alpha_1 \times N_{REC} \times \left(1 - \frac{38}{10000}\right)$

= **100602**

2) **DB-Key / PP**

best case Clusterbildung DBTT + Datenseiten
: Wurzel + 2. Index + 3. Index + DBTT sequ. + Daten sequ.
1 + 1 + 1000 + 100 + 10 000
= **11102**

best case Clusterbildung Datenseiten
: Wurzel + 2. Index + 3. Index + DBTT + Datenseiten sequ.
1 + 1 + 1000 + N_{REC}(1-38/100)+ 10000
= **73002**

worst case : Wurzel + 2. Index + 3. Index + DBTT + Daten

$$1 + 1 + 1000 + N_{REC} \cdot \left(1 - \frac{19}{100}\right) + N_{REC} \cdot \left(1 - \frac{19}{10000}\right)$$

= **180 810**

3) **DB-Key / PP/PPP**

best case (Clusterung der Daten):
(DBTT irrelevant, da im Normalfall kein Zugriff)

Wurzel	+ 2. Index	+3. Index	+ Daten
1	+1	+ 1000	+10000

$$+ \text{Fehler} = \left(p_{ov} \cdot N_{REC} \cdot \left(1 - \frac{19}{100} + 1 - \frac{19}{10000}\right)\right)$$

$$+ 0{,}1 \cdot N_{REC} \cdot \left(1 - \frac{19}{100} + 1 - \frac{19}{10000}\right)$$

$$= 11002 + 10000 \cdot (0{,}81 + 0{,}9981)$$

$$= 11002 + 10000 \cdot 1{,}8081$$

$$= \mathbf{29083}$$

worst case (Clustering Daten):

Wurzel	+ 2. Index	+3. Index	+
1	+1	+ 1000	+

Daten + (Daten + DBTT)

$$N_{REC} \cdot \left(1 - \frac{35}{10000}\right) + 0{,}1 \cdot N_{REC} \cdot \left(1 - \frac{4}{100} + 1 - \frac{35}{10000}\right)$$

$$= 11002 + 99650 + 9600 + 9965$$

$$= \mathbf{120217}$$

Zusammenfassender Vergleich:

Adressierung:

	Formel	n =10
LBA	4n + 4	44
TID	4 n + 6,6	46,6
DBK	4n + 9	49
PPP	7n + 9	79

Seitenzugriffe

seq. Zugriff	Datencluster	Daten + DBTTcluster	Keine Clusterung
TID	20964		100612
DBK	73002	11102	180810
PPP	29083	29083	120217
wahlfreie Zugriffe	Index	direkt	
TID	2,08335	1,0956	
DBK	2,889	1,798	
PPP	2,182	1,192	

Aufgabe 2: *Pointer-Swizzling* bei Bäumen

Gegeben ist ein vollständig belegter B*-Baum der Klasse *T*(k, k*, h), der in seinen Blattseiten auf Sätze verweist, die keine weiteren Verweise enthalten.

Die B*-Baum-Seiten besitzen folgendes Format:

Innerer Knoten

Z_0	R_1	Z_1	R_2	Z_2	• • •	R_p	Z_p	freier Platz

R_i = Referenzschlüssel (Wegweiser), $k \le p \le 2\,k$

Z_i = Verweis auf Sohnseite

Blatt-knoten

P	S_1	D_1	S_2	D_2	• • •	S_j	D_j	freier Platz	N

P = PRIOR-Zeiger, N = NEXT-Zeiger, $k^* \le j \le 2\,k^*$

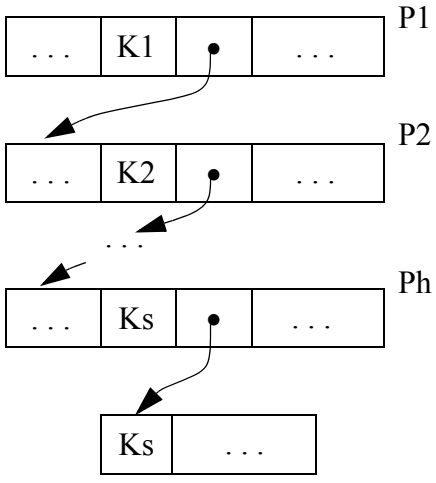
S_i = Satzschlüssel

D_i = Verweis auf Satz (referenzierte Speicherung)

Eine innere Seite (inkl. der Wurzel) habe also bei voller Belegung 2k+1 Verweise auf Seiten und eine Blattseite auf der Stufe h habe 2k* Verweise auf Sätze.

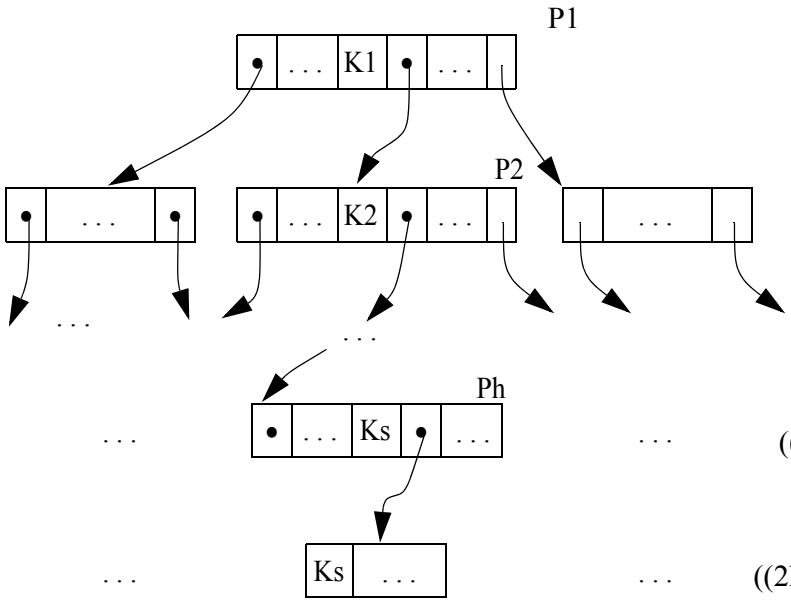
Anfangs seien keine Seiten des B*-Baumes und keine der referenzierten Sätze im Hauptspeicher. Im folgenden soll Copy-Swizzling analysiert werden.

Skizzieren Sie für die verschiedenen Swizzling-Varianten die einzelnen Aktionen, die aus einem Suchvorgang für den Satz mit Schlüssel Ks resultieren, wenn dabei die Wurzelseite P1, auf Baumebene 2 die Seite P2, ... und auf Baumebene h die Seite Ph aufgesucht werden. Der Suchpfad ergebe sich durch die Schlüssel (Wegweiser) K1 in P1, K2 in P2, ..., Ks in Ph:



Wie viele Seiten, Deskriptoren und Sätze sind jeweils einzulagern und aufzusuchen, wenn als Verfahren

a) **eager, direct**



1	Seite
2k+1	Verweise
(2k+1)	Seiten
$(2k+1)^2$	Verweise
$(2k+1)^{h-1}$	Seiten
$((2k+1)^{h-1}) \cdot 2k^*$	Verweise
$((2k+1)^{h-1}) \cdot 2k^*$	Sätze

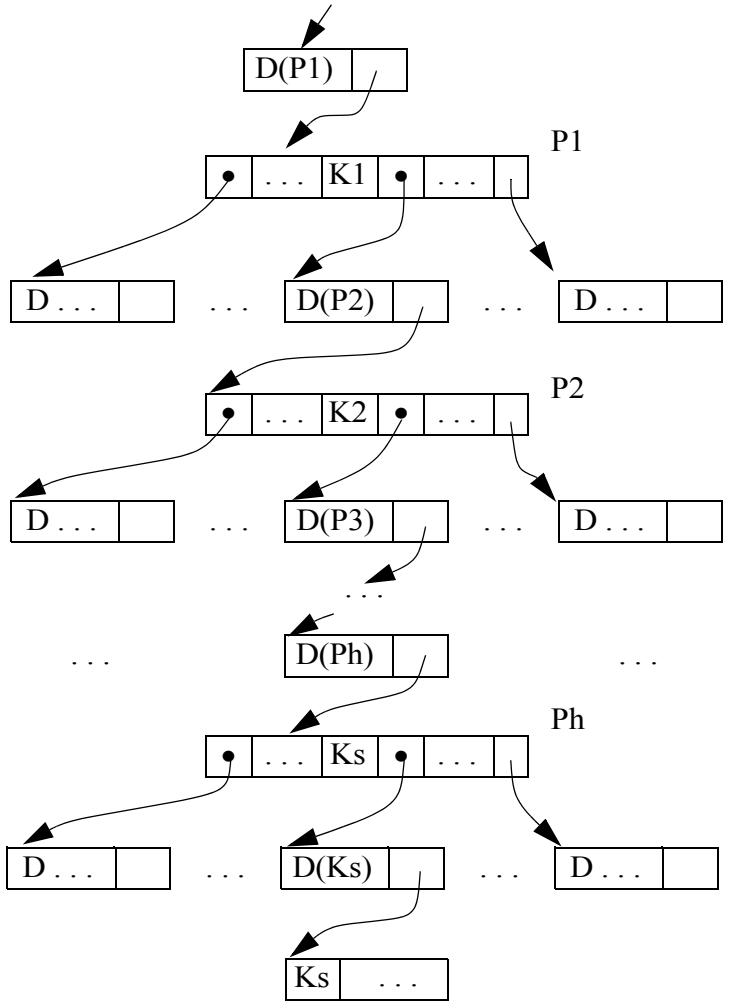
Einlagerung bei erstmaliger Referenz von P1:

- $(1 + \sum_{i=2}^{h} (2k+1)^{i-1})$ Seiten
- $((2k+1)^{h-1}) \cdot 2k^*$ Sätze

Lesen von

- h Seiten
- 1 Satz

b) **eager, indirect**



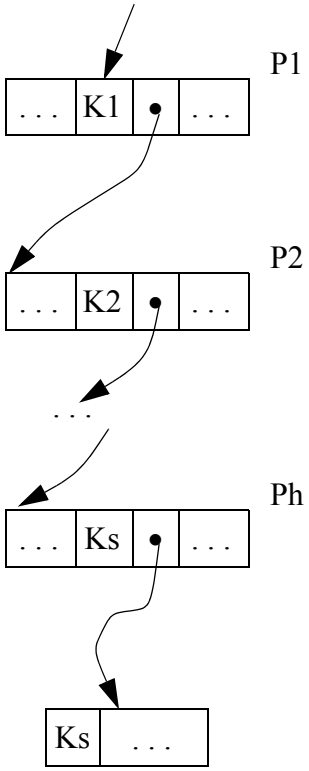
1	Deskriptor
1	Seite
2k+1	Deskriptoren
1	Seite
2k+1	Deskriptoren
2k+1	Deskriptoren
1	Seite
2k*	Deskriptoren
1	Satz

- Beginn der Suche nach Ks:
  - Anlegen von D(P1)
- Zugriff auf P1
  - Einlagerung von P1
  - Anlegen von D(P2) und weiteren 2k Desk.
  - Suche nach K1
- Zugriff auf P2
  - . . .
- Zugriff auf Ph
  - Einlagerung von Ph
  - Anlegen von D(Ks) und weiteren (2k*-1) Desk.
  - Suche nach Ks
- Zugriff auf Ks
  - Einlagerung von Satz mit Ks als Schlüssel
  - Zugriff auf Satz Ks

Insgesamt:

- Einlagerung/Anlegen von
  - h Seiten
  - $1 + (h-1) \cdot (2k+1) + 2k^*$ Desk.
  - 1 Satz
- Lesen von
  - h Seiten
  - h + 1 Deskriptoren
  - 1 Satz

c) **lazy, direct**

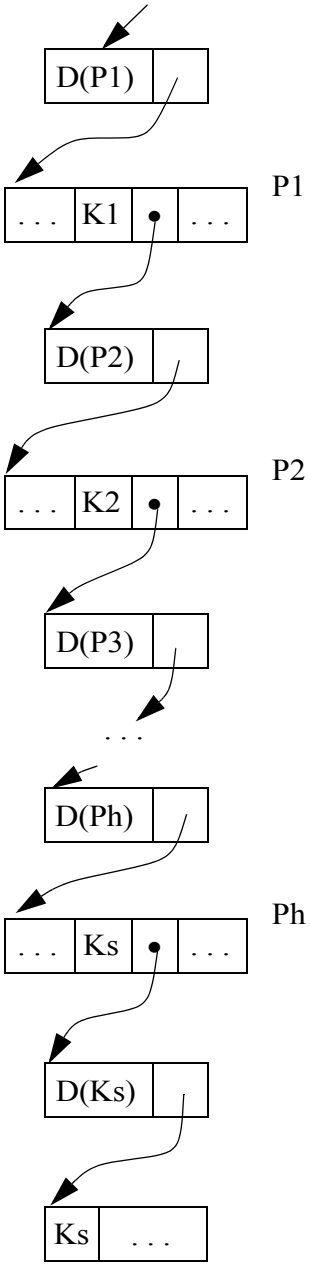


- Beginn der Suche nach Ks:
- Zugriff auf P1
  - Einlagern von P1
  - Suche nach K1
- Zugriff auf P2
  - . . .
- Zugriff auf Ph
  - Einlagern von Ph
  - Suche nach Ks
- Zugriff auf Ks
  - Einlagern von Satz mit Ks als Schlüssel
  - Zugriff auf Satz Ks

Insgesamt:

- Einlagerung von
  - h Seiten
  - 1 Satz
- Lesen von
  - h Seiten
  - 1 Satz

d) **lazy, indirect**



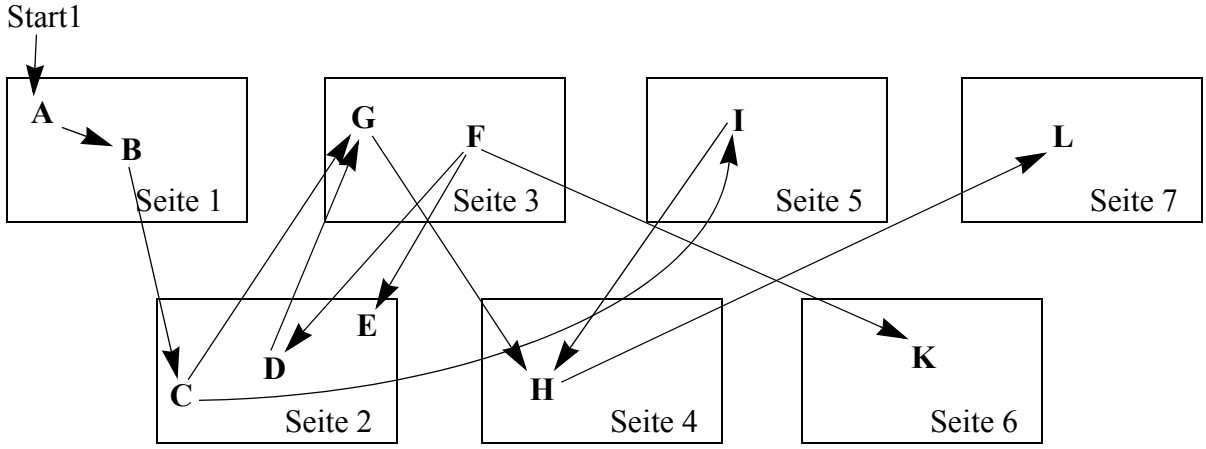
- Beginn der Suche nach Ks:
  - Anlegen von D(P1)
- Zugriff auf P1
  - Einlagerung von P1
  - Suche nach K1
  - Anlegen von D(P2)
- Zugriff auf P2
  - . . .
- Zugriff auf Ph
  - Einlagerung von Ph
  - Suche von Ks
  - Anlegen von D(Ks)
- Zugriff von Ks
  - Einlagerung von Ks
  - Lesen von Ks

Insgesamt:

- Einlagern / Anlegen von
  - h Seiten
  - h + 1 Deskriptoren
  - 1 Satz
- Lesen von
  - h Seiten
  - h + 1 Seiten
  - 1 Satz

Aufgabe 3: ***Pointer-Swizzling*** **(*direct in-place*)** 187

Gegeben seien die Objekte A bis L, die auf externem Speicher in den Seiten 1 bis 7 wie unten abgebildet abgelegt sind. Die Pfeile repräsentieren dabei die Beziehungen der Objekte untereinander.



Es wird angenommen, dass diese Verweise als Externspeicheradressen vorliegen. Ein Transaktion referenziere die einzelnen Objekte in folgender Reihenfolge:

A B C G F E I H L K

Im folgenden werde immer von einem *Swizzling* mit den Eigenschaften *in-place, direct, lazy* ausgegangen, wobei bei der ersten Referenz auf eine Hauptspeicheradresse umgestellt wird.

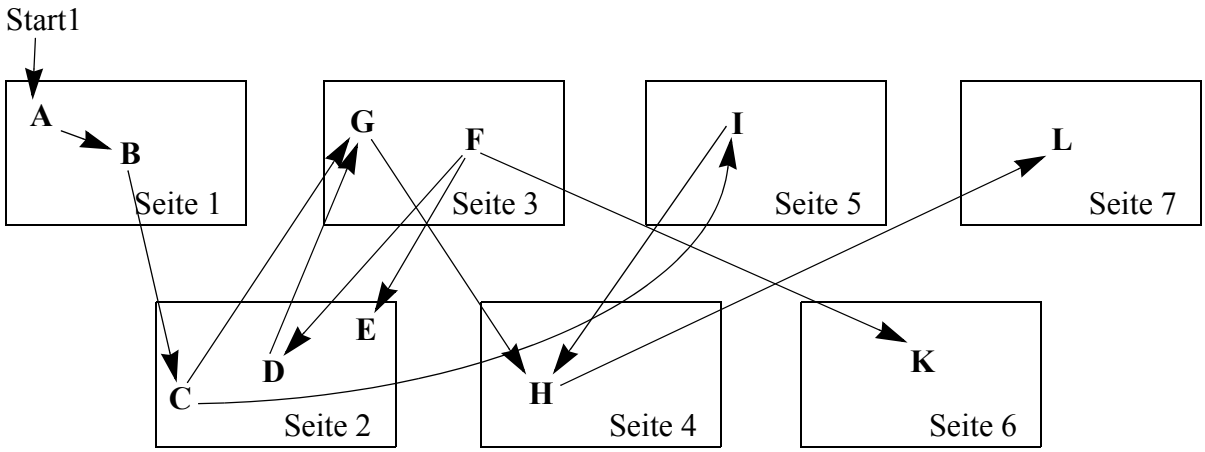
a) Welche Auswirkungen haben die obigen Objektreferenzen auf die Hauptspeicherbelegung, wenn davon ausgegangen werden kann, dass alle Seiten in den Hauptspeicher passen?

Bei Referenz eines Objektes wird die zugehörige Seite eingelagert. Die von einem Objekt ausgehenden Verweise werden nach dem lazy-Verfahren (bei erster Referenz (Nutzung des Verweises)) umgestellt.

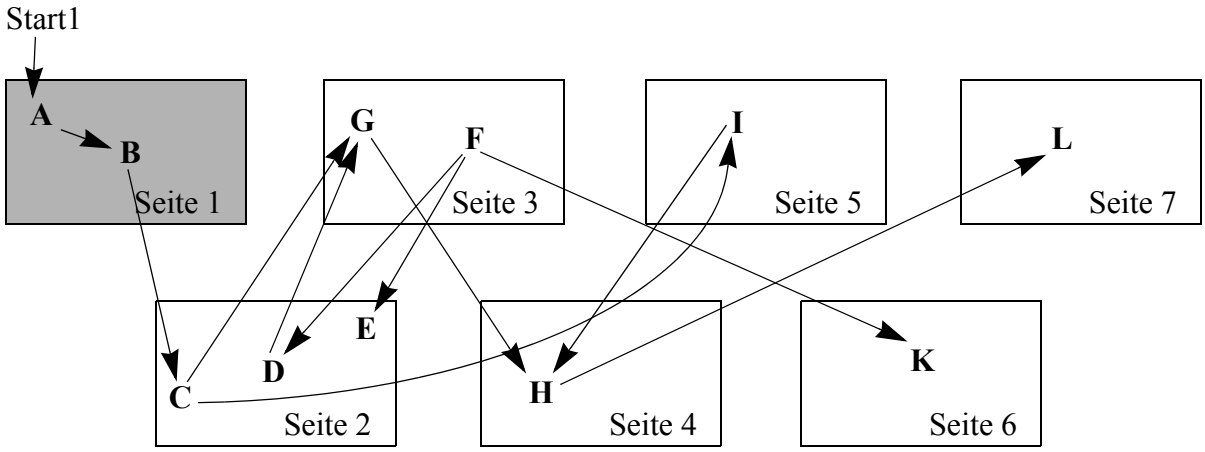
b) Angenommen es stünden nur 5 Seiten im Hauptspeicher zur Verfügung. Beim Einlesen der Seite mit Objekt L in Seite 7 müsste dann Platz geschaffen werden. Welche Seite(n) könnte(n) verdrängt werden?

Dunkelgraue Seiten sind eingelagert:

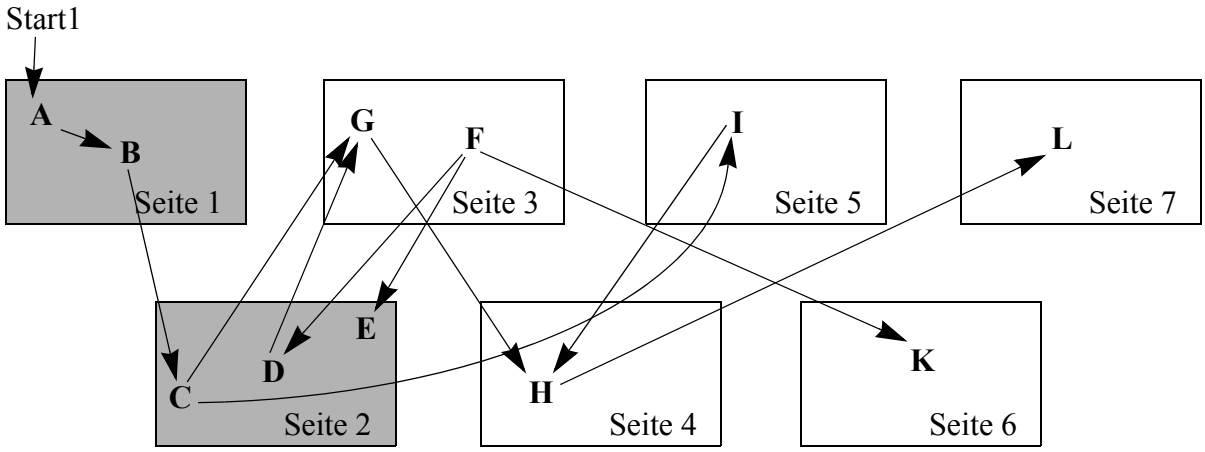
Ausgangssituation:



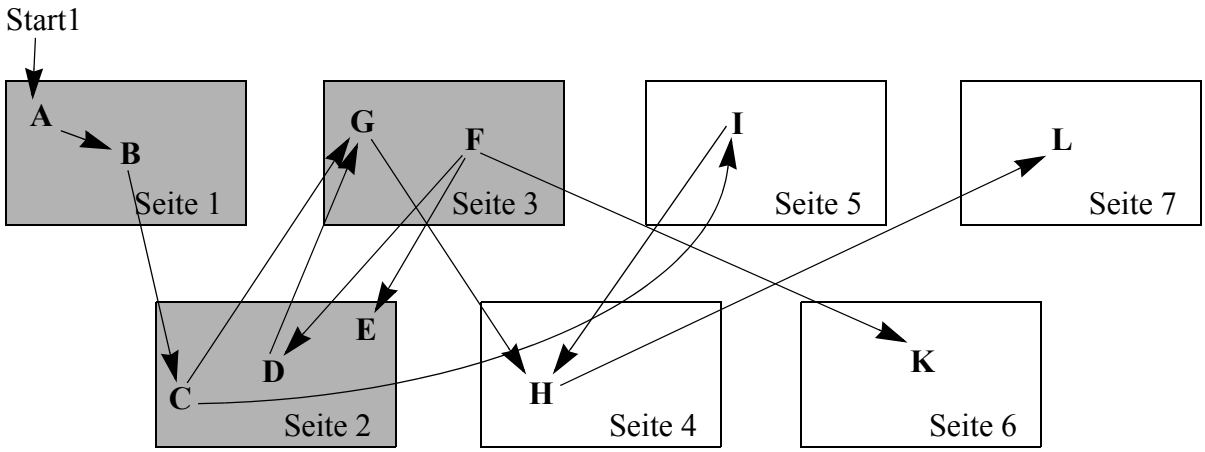
Nach der Referenz auf A und auf B:



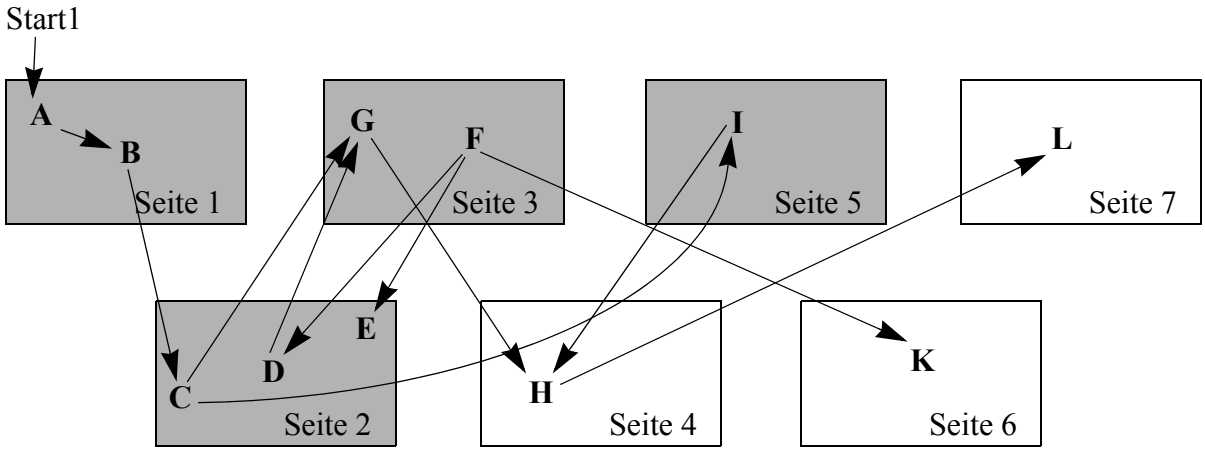
nach der Referenz auf C:



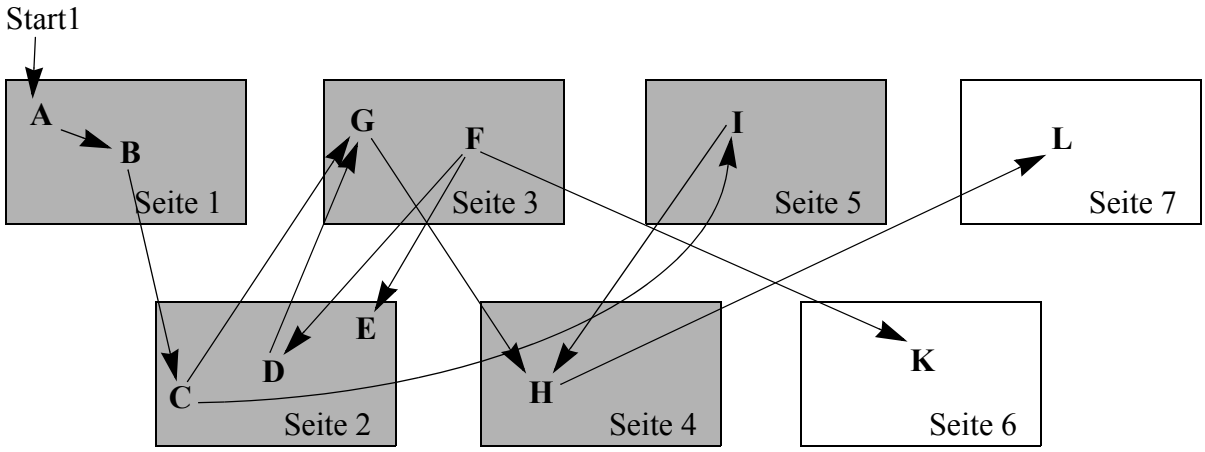
nach der Referenz auf G:



nach der Referenz auf F, E und I:



nach der Referenz auf H:



Da es keine Referenzen auf Objekte in Seite 1 gibt, könnte diese Seite verdrängt werden. Alle anderen Seiten können nicht so einfach verdrängt werden, weil dann Verweise auf in diesen Seiten enthaltene Objekte nicht mehr gültig sein können.

c) Wie wäre die Situation, wenn zwischenzeitlich in Seite 1 ein neues Objekt eingefügt worden wäre?

Wenn zwischenzeitlich ein Objekt in Seite 1 eingefügt worden ist, hängt eine mögliche Seitenersetzung davon ab, von welchen Objekten dieses neue Objekt referenziert wird (s. o.).

d) Welche Informationen für die einzelnen Seiten/Objekte werden beim *Pointer-Swizzling* benötigt?

Informationen in einer Seite:
1. welche Objekte sind in der Seite
2. an welcher Stelle stehen diese Objekte innerhalb der Seite

Informationen über die Objekte:
1. welche externe Adresse wird auf welche interne Adresse abgebildet
2. welche Objekte werden von einem Objekt referenziert

Aufgabe 4: Abbildung von Datensätzen in Seiten

Jede Datenseite benötigt gewisse Informationen, die der Identifikation und der Selbstbeschreibung dienen. Sie werden im sogenannten *Page-Header* abgelegt. Überlegen Sie, welche Informationen zur Beschreibung einer Datenseite notwendig sind.

a) Entwerfen Sie für den *Page-Header* eine geeignete Datenstruktur und legen Sie den Typ der einzelnen Elemente dieser Struktur fest.

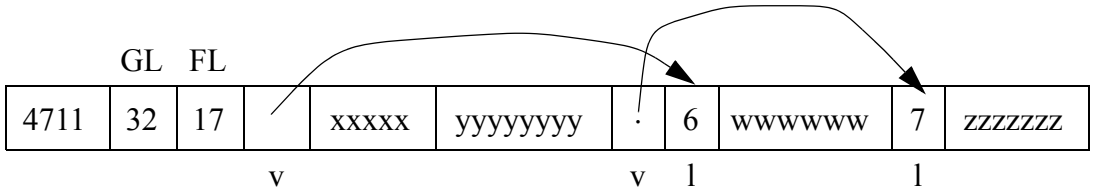
Seiten-Hader-Informationen:
- Segment_Nr.,
- Seiten_Nr.,
- next_ptr zur sequentiellen Verarbeitung
- # freier Bytes
- Seitentyp (Datenseite, Freispeicherverwaltung, Zugriffspfad, ...)

Die Sätze einer Tabelle TAB seien nach dem 4. Verfahren im Skript als Speicherungsstrukturen abgelegt. Anfangs haben sie 4 Felder und sind im Katalog durch v, f5, f8, v beschrieben.

Ein bestimmter Satz mit SKZ = 4711 habe folgende Repräsentation, wobei der Zeiger eines v-Feldes 2 Byte und die Längendarstellung eines v-Feldes (l) 1 Byte ausmachen.

Durch

Alter Table TAB (Add A-Feld var) und Alter Table TAB (Add B-Feld fixed (3))



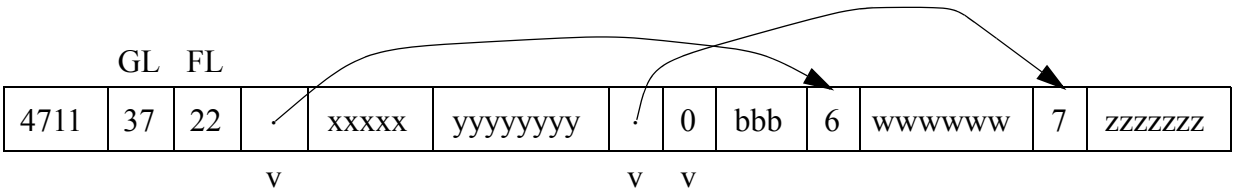
werden dynamisch zwei Felder an TAB-Sätze angehängt, was zunächst nur Auswirkungen auf die Katalog-Information von TAB hat: v, f5, f8, v, v, f3. Die einzelnen Sätze von TAB werden erst bei ihrer Aktualisierung mit Werten für A-Feld und B-Feld modifiziert.

Welche Änderungen an der Speicherungsstruktur von Satz 4711 ergeben sich, wenn

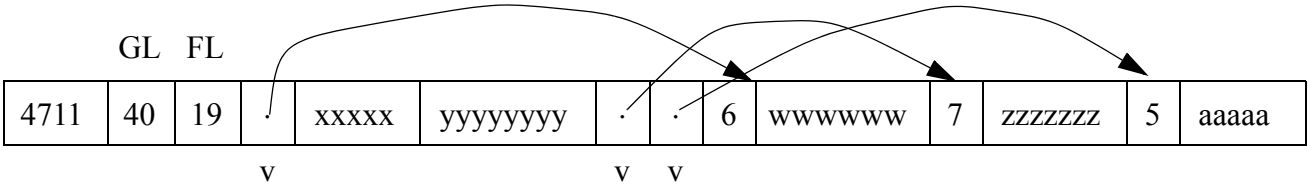
b) beide neuen Felder noch keinen Wert besitzen

keine Änderung

c) B-Feld den Wert bbb und A-Feld noch keinen Wert besitzt



d) A-Feld den Wert 5aaaaa und B-Feld noch keinen Wert besitzt



e) beide neuen Felder die obigen Werte haben?