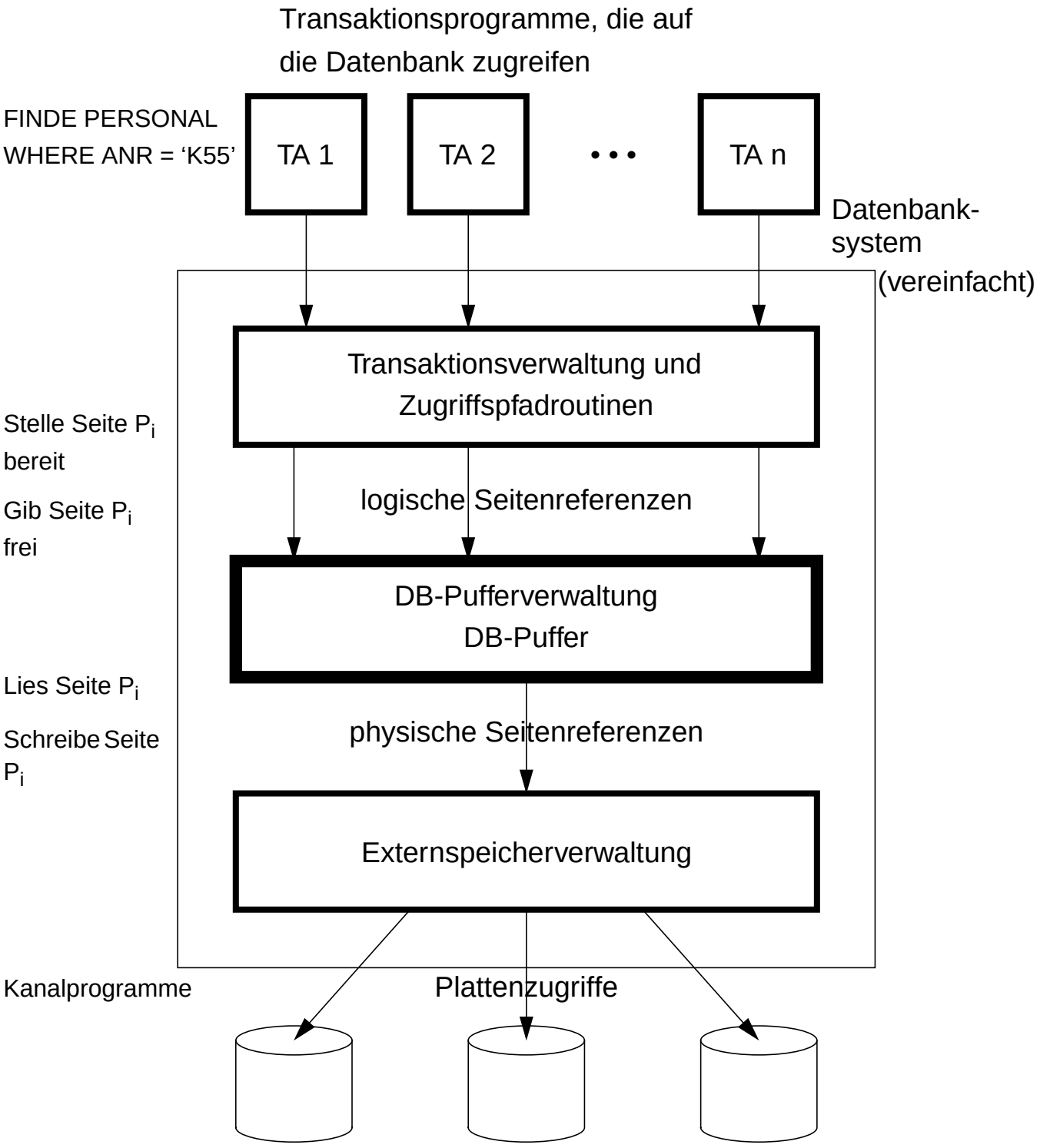
3. DB-Pufferverwaltung

- **Rolle der DB-Pufferverwaltung**[1]
  - Ablauf des Zugriffs auf den DB-Puffer
  - logische und physische Seitenreferenzen
- **Speicherzuteilung im DB-Puffer**
- **Suche im DB-Puffer**
- **Seitenersetzungsverfahren**
  - Klassifikation von Ersetzungsverfahren
  - LRU, FIFO, CLOCK, GCLOCK, LRD, LRU-K ...
- **Ersetzungsverfahren – Einbezug von Kontextwissen**
- **Seitenersetzung bei virtuellem Speicher**

1. Effelsberg, W., Härder, T.: Principles of Database Buffer Management, in: ACM Transactions on Database Systems 9:4, Dec. 1984, pp. 560-595.

Rolle der DB-Pufferverwaltung

in einem Datenbanksystem



Vergleich mit Betriebssystemfunktionen

- **Ersetzungsalgorithmen im DB-Puffer in Software implementiert – Seitenersetzung in Adreßräumen bei Virtuellem Speicher ist HW-gestützt**

- **Seitenreferenz vs. Adressierung**

  nach einem FIX-Aufruf kann eine DB-Seite mehrfach bis zum UNFIX referenziert werden

  ➥ unterschiedliches Seitenreferenzverhalten

  ➥ andere Ersetzungsverfahren ?

- **Können Dateipuffer des Betriebssystems als DB-Puffer eingesetzt werden?**

1. **Zugriff auf Dateipuffer ist teuer (SVC:** *supervisor call***)**

2. **DB-spezifische Referenzmuster können nicht mehr gezielt genutzt werden**

   BS-Ersetzungsverfahren sind z. B. nicht auf zyklisch sequentielle oder baumartige Zugriffsfolgen abgestimmt

3. **Normale Dateisysteme (DVS) bieten keine geeignete Schnittstelle für Prefetching**

   In DBVS ist aufgrund von Seiteninhalten oder Referenzmustern eine Voraussage des Referenzverhaltens möglich; durch Prefetching läßt sich in solchen Fällen eine Leistungssteigerung erzielen

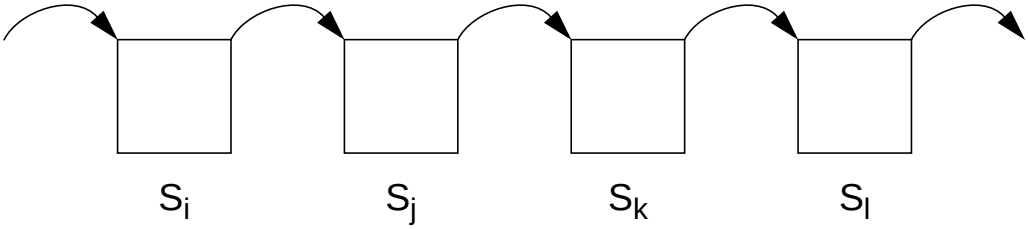
4. **Selektives Ausschreiben von Seiten zu bestimmten Zeitpunkten (z. B. für Logging) nicht immer möglich in existierenden Dateisystemen**

   ➥ DBVS muß eigene Pufferverwaltung realisieren

Eigenschaften von DB-Referenzstrings

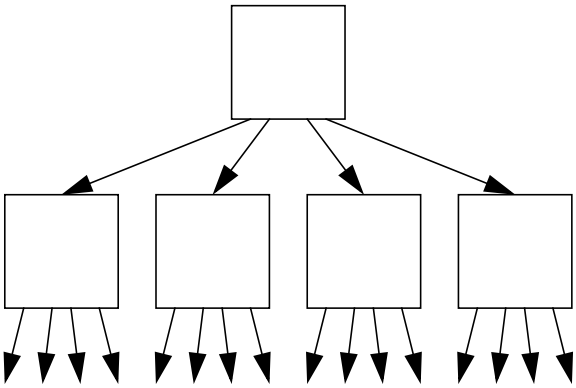
- **Typische Referenzmuster in DBS**

1. Sequentielle Suche



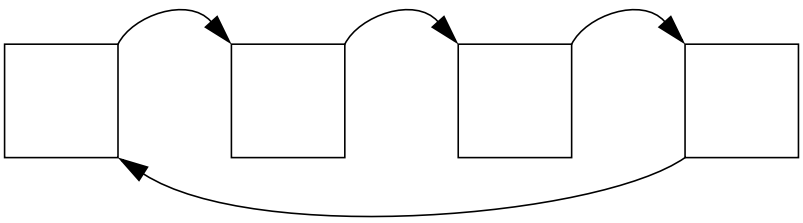
Bsp.: Durchsuchen ganzer Satztypen (Relationen)

2. Hierarchische Pfade



Bsp.: Suchen mit Hilfe von B*-Bäumen

3. Zyklische Pfade



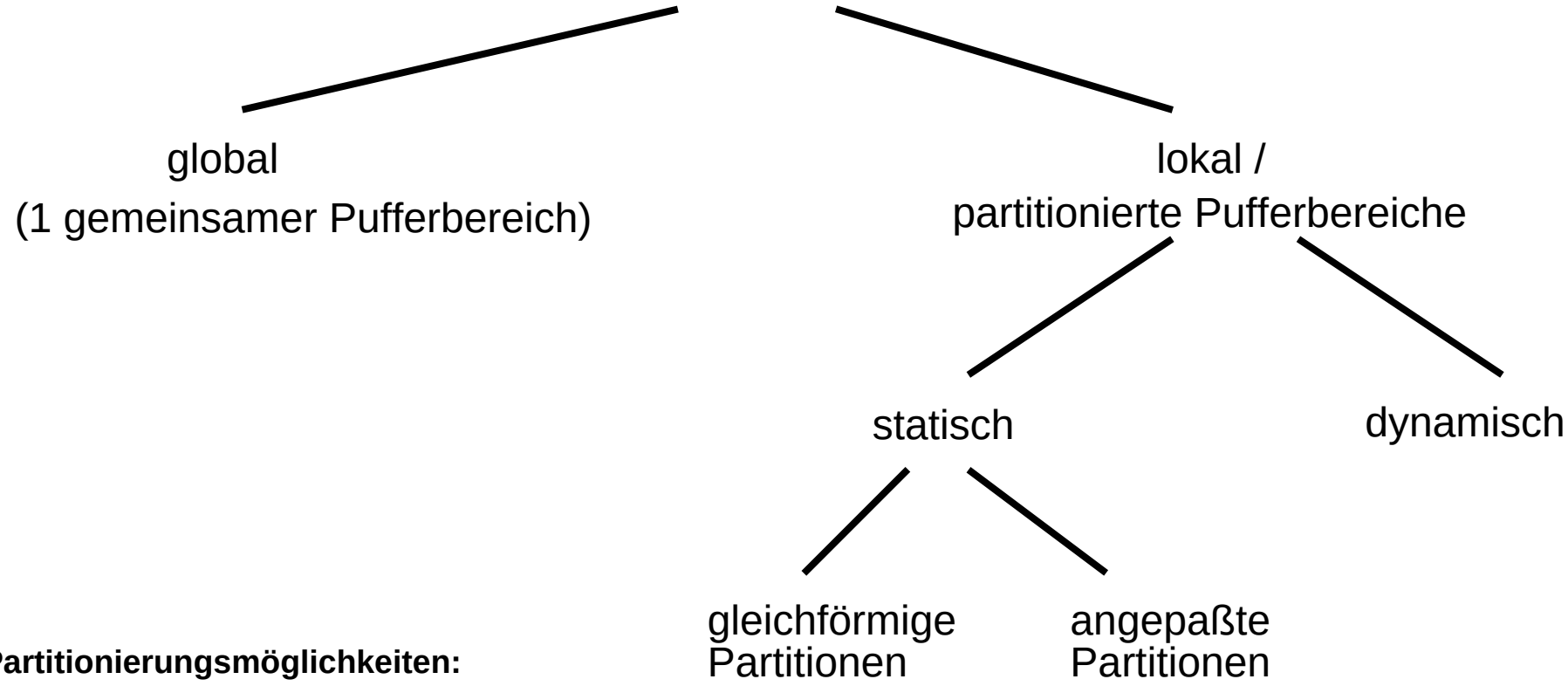
Bsp.: Abarbeiten von Sets, Suchen in DBTT-/Datenseiten

Relative Referenzmatrix (DOA-Last)

ca. 17 500 Transaktionen, 1 Million Seitenreferenzen auf ca. 66 000 verschiedene Seiten

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	Total
TT1	9.1	3.5	3.3		5.0	0.9	0.4	0.1				0.0		22.3
TT2	7.5	6.9	0.4	2.6	0.0	0.5	0.8	1.0	0.3	0.2	0.0			20.3
TT3	6.4	1.3	2.8	0.0	2.6	0.2	0.7	0.1	1.1	0.4		0.0	0.0	15.6
TT4	0.0	3.4	0.3	6.8			0.6	0.4			0.0			11.6
TT5	3.1	4.1	0.4		0.0		0.5	0.0						8.2
TT6	2.4	2.5	0.6		0.7		0.9	0.3						7.4
TT7	1.3		2.6			2.3	0.1							6.2
TT8	0.3	2.3	0.2		0.0		0.1							2.9
TT9	0.0	1.4	0.0					1.1						2.6
TT10	0.3	0.1	0.3			1.0	0.1					0.0		1.8
TT11		0.9						0.2						1.1
TT12		0.1												0.1
Total	30.3	26.6	11.0	9.4	8.3	4.9	4.1	3.3	1.4	0.6	0.0	0.0	0.0	100.0
partition size (%)	31.3	6.3	8.3	17.8	1.0	20.8	2.6	7.3	2.6	1.3	0.8	0.0	0.0	100.0
% refe-renced	11.1	16.6	8.0	2.5	18.1	1.5	9.5	4.4	5.2	2.7	0.2	13.5	5.0	6.9

Speicherzuteilung im DB-Puffer



Partitionierungsmöglichkeiten:

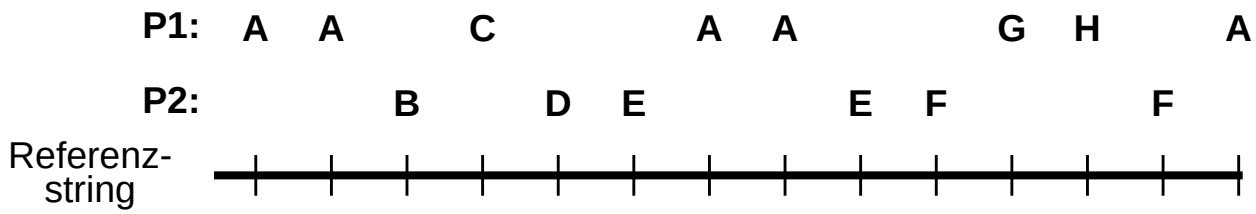
- eigener Pufferbereich pro Transaktion
- TA-Typ-bezogene Pufferbereiche
- Seitentyp-bezogene Pufferbereiche
- DB-(Partitions)spezifische Pufferbereiche

Dynamische Pufferallokation – Working-Set-Ansatz (WS)

- **pro Pufferpartition P soll Working-Set im Puffer bleiben; Seiten, die nicht zum Working-Set gehören, können ersetzt werden**

- **bei Fehlseitenbedingung muß Working-Set bekannt sein, um Ersetzungskandidat zu bestimmen**
  - Fenstergröße (*Window Size*) pro Partition: w (P)
  - Referenzzähler pro Partition: RZ (P)
  - letzter Referenzzeitpunkt für Seite i: LRZ (P, i)
  - ersetzbar sind solche Seiten, für die RZ (P) - LRZ (P, i) > w (P)

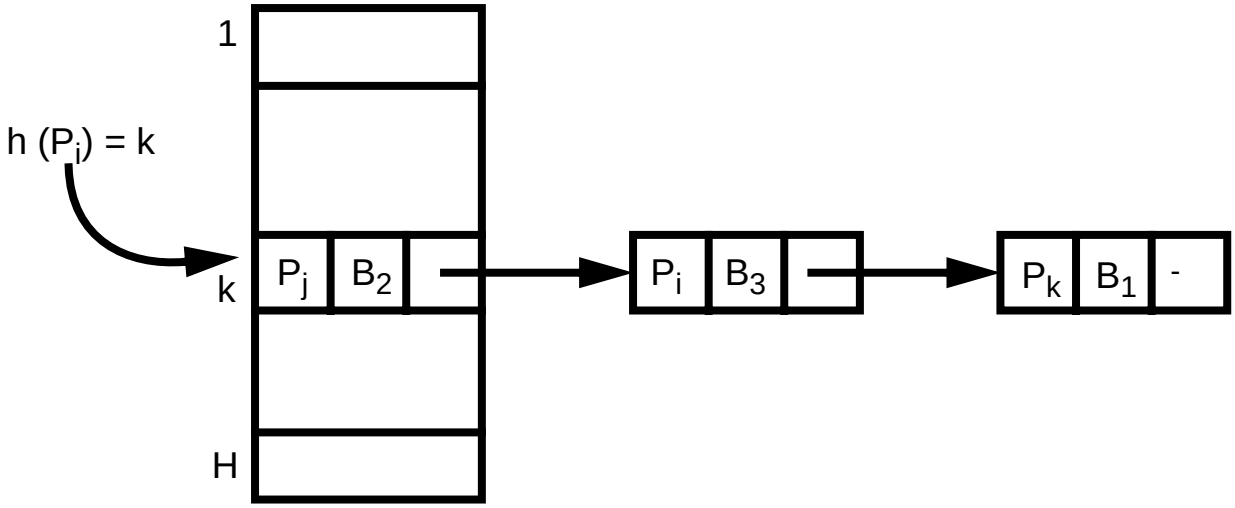
- **Fenstergröße kritischer Parameter $\rightarrow$ Thrashing-Gefahr**



Suche im DB-Puffer

- **Sequentielles Durchsuchen der Pufferrahmen**
  - sehr hoher Suchaufwand
  - Gefahr vieler Paging-Fehler bei virtuellen Speichern
- **Nutzung von Hilfsstrukturen**
  (Eintrag pro Pufferrahmen)

  1. **unsortierte oder sortierte Tabelle**
  2. **Tabelle mit verketteten Einträgen**
     - geringere Änderungskosten
     - Anordnung in LRU-Reihenfolge möglich
  3. **Suchbäume (z. B. AVL-, m-Weg-Bäume)**
  4. **Hash-Tabelle mit Überlaufketten**
     - beste Lösung

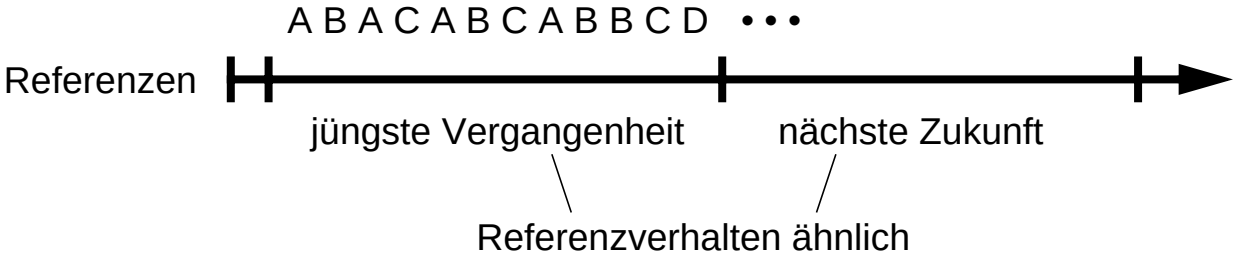


Seitenersetzungsverfahren

- **Klassifikation**

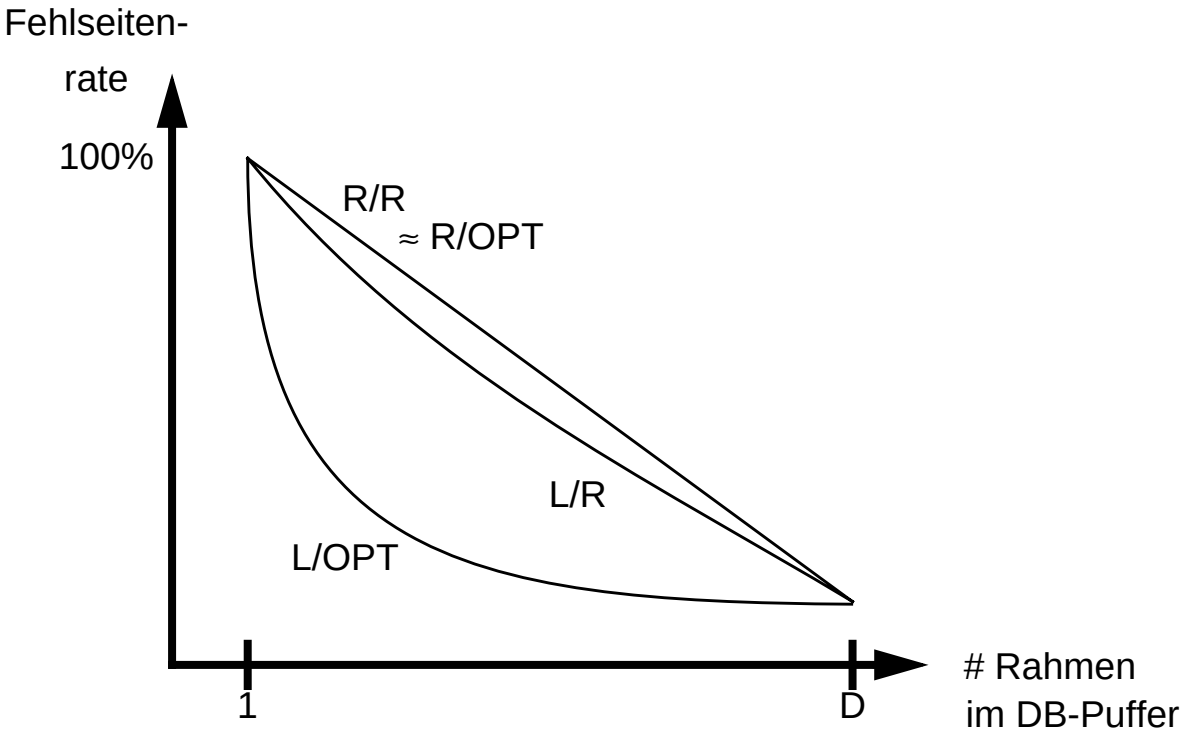
Programmanalyse, Vorabuntersuchung des Datenbedarfs	physische Datenstrukturierung, Clusterbildung, Verarbeitungswissen	keine Vorausaktionen
➥ große Fehlrate, ungenaue Obermengen	➥ datenmodellbezogen (hierarchisch), spekulative Entscheidungen	➥ Lokalitätserhaltung im DB-Puffer

- **Grundannahme bei Ersetzungsverfahren**:



Referenzverhalten und Ersetzungsverfahren

- **typischerweise** hohe Lokalität: Optimierung durch Ersetzungsverfahren
- **manchmal** Sequentialität oder zufällige Arbeitslast (RANDOM-Referenzen)
- **Prinzipielle Zusammenhänge**, die die Fehlseitenrate bestimmen



D = DB-Größe in Blöcken

- **Kombinationen:**

Referenzen:	RANDOM	RANDOM	Lokalität	Lokalität
Ersetzung:	RANDOM	OPT	RANDOM	OPT

➥Grenzfälle des Referenzverhaltens und der Ersetzungsverfahren zeigen Optimierungsmöglichkeiten auf

Behandlung geänderter Seiten im DB-Puffer

- **Ersetzung einer geänderten Seite erfordert ihr vorheriges (synchrones) Zurückschreiben in die DB**

  ➥ Antwortzeitverschlechterung

- **Abhängigkeit zur gewählten *Ausschreibstrategie*:**

  FORCE: alle Änderungen einer Transaktion werden spätestens beim EOT in die DB zurückgeschrieben („write-through“)

  \+ i. a. stets ungeänderte Seiten zur Ersetzung vorhanden

  \+ vereinfachte Recovery (nach Rechnerausfall sind alle Änderungen beendeter TA bereits in die DB eingebracht)

  \- hoher E/A-Overhead

  \- starke Antwortzeiterhöhung für Änderungstransaktionen

  NOFORCE: kein Durchschreiben der Änderungen bei EOT (verzögertes Ausschreiben, „deferred write-back“)

  ➥Seite kann mehrfach geändert werden, bevor ein Ausschreiben erfolgt (geringerer E/A-Overhead, bessere Antwortzeiten)

  vorausschauendes (asynchrones) Ausschreiben geänderter Seiten

  erlaubt auch bei NOFORCE, vorwiegend ungeänderte Seiten zu ersetzen

  ➥ synchrone Schreibvorgänge in die DB lassen sich weitgehend vermeiden

Kriterien für die Auswahl der zu ersetzenden Pufferseite

Verfahren	Kriterien			
	Alter	letzte Referenz	Referenz-häufigkeit	andere Kriterien
OPT	-	-	-	**Vorauswissen**
RANDOM	-	-	-	---
LFU				
FIFO				
LRU				
CLOCK				
GCLOCK				
LRD(V1)				
LRD(V2)				
LRU-K				

Least-Frequently-Used (LFU)

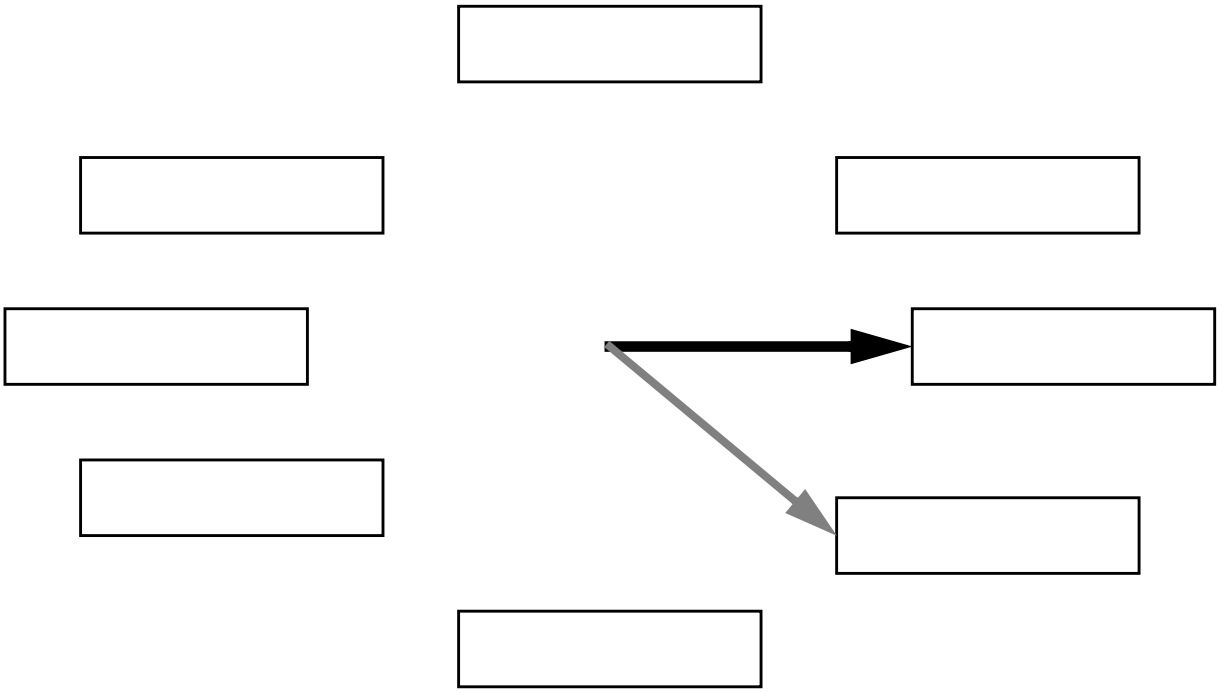
- **Führen eines Referenzzählers pro Seite im DB-Puffer**
- **Ersetzung der Seite mit der geringsten Referenzhäufigkeit**

RZ	
2	
4	
1	
3	
3	
6	
1	
3	

- **Alter einer Seite wird nicht berücksichtigt**

FIFO (First-In First-Out)

- **die älteste Seite im DB-Puffer wird ersetzt**



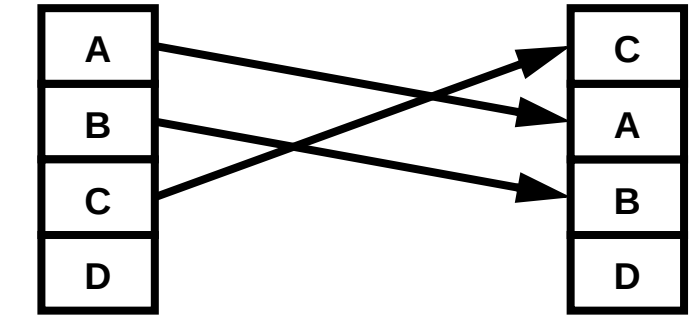
- **Referenzierungsverhalten während Pufferaufenthaltes wird nicht berücksichtigt**

  ⇒ nur für strikt sequentielles Referenzverhalten geeignet

Least-Recently-Used (LRU)

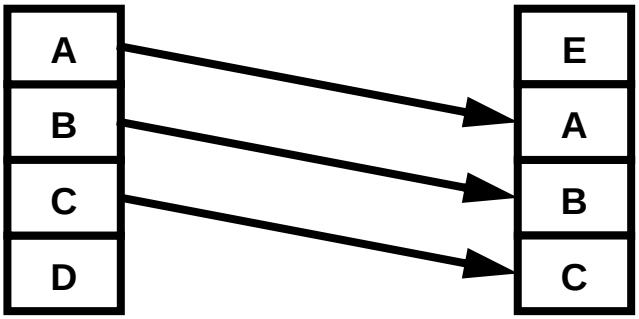
- **Beispiel (Puffergröße 4):**

  1. Referenz der Seite C



LRU-Stack

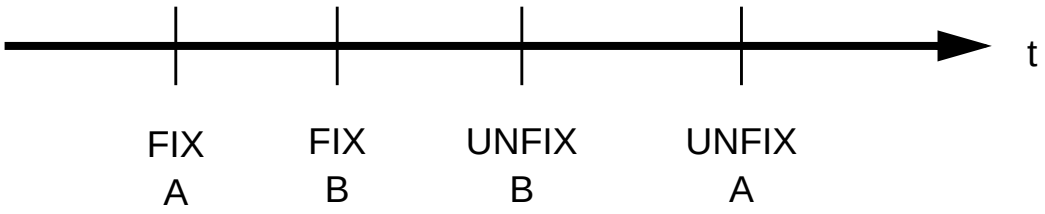
  2. Referenz der Seite E



- **Unterscheidung zwischen**

  Least-Recently-Referenced **und**

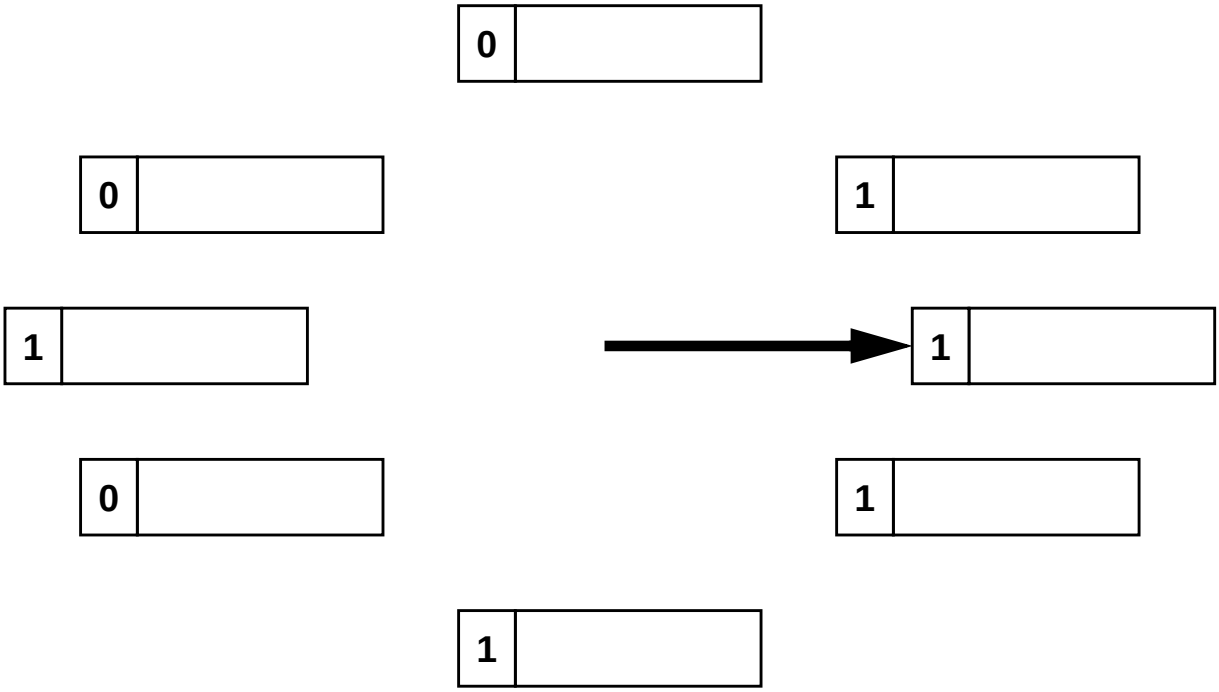
  Least-Recently-Unfixed



CLOCK (Second Chance)

- **Erweiterung von FIFO**
- **Referenzbit pro Seite, das bei Zugriff gesetzt wird**
- **Ersetzung erfolgt nur bei zurückgesetztem Bit**

  (sonst erfolgt Zurücksetzen des Bits)

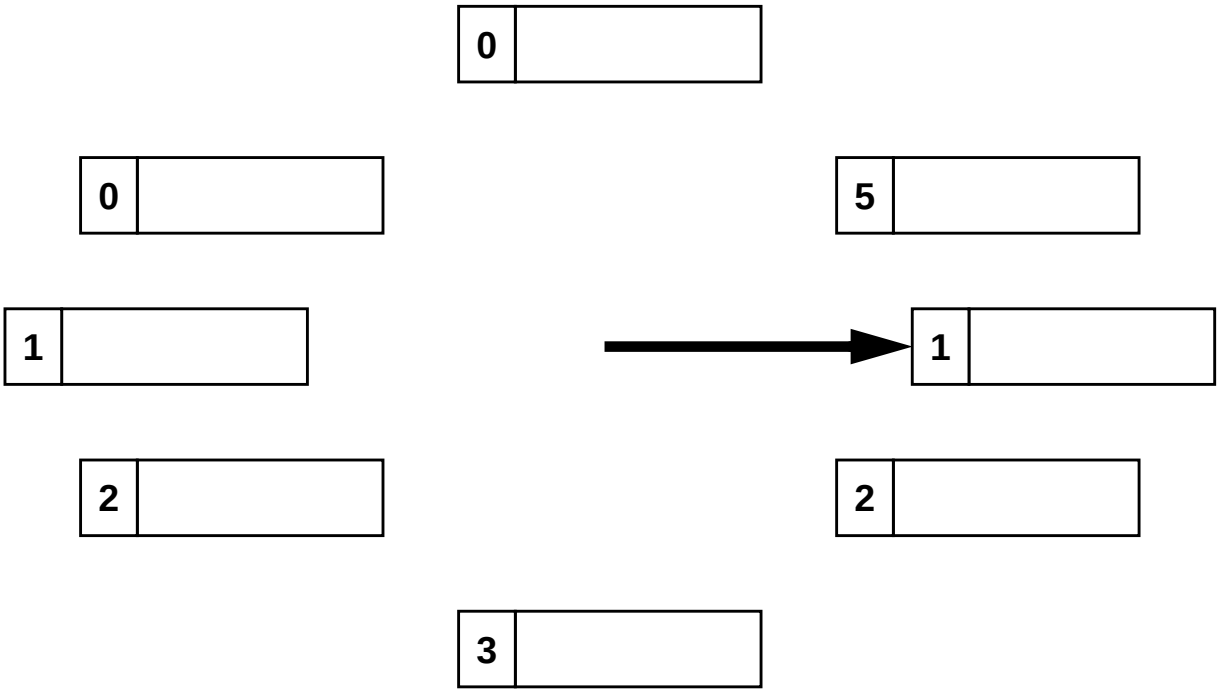


- **annähernde Berücksichtigung des letzten Referenzierungszeitpunktes**

GCLOCK (Generalized CLOCK)

- **pro Seite wird Referenzzähler geführt (statt Bit)**
- **Ersetzung nur von Seiten mit Zählerwert 0**

  (sonst erfolgt Dekrementierung des Zählers und Betrachtung der nächsten Seite)



- **Verfahrensparameter:**
  - Initialwerte für Referenzzähler
  - Wahl des Dekrementes
  - Zählerinkrementierung bei erneuter Referenz
  - Vergabe von seitentyp- oder seitenspezifischen Gewichten

Least-Reference-Density (LRD)

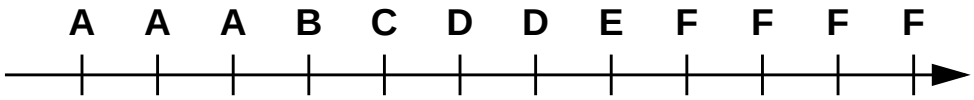
- **Referenzdichte: Referenzhäufigkeit während eines bestimmten Referenzintervalls**

- **Variante 1: Referenzintervall entspricht Alter einer Seite**

- **Berechnung der Referenzdichte:**

  Globaler Zähler GZ: Gesamtanzahl aller Referenzen
  Einlagerungszeitpunkt EZ: GZ-Wert bei Einlesen der Seite
  Referenzzähler RZ

  Referenzdichte $RD(j) = \frac{RZ(j)}{GZ - EZ(j)}$



	RZ	EZ	RD
A			
B			
C			
D			
E			
F			

Least-Reference-Density (2)

- **Variante 2: konstante Intervallgröße**

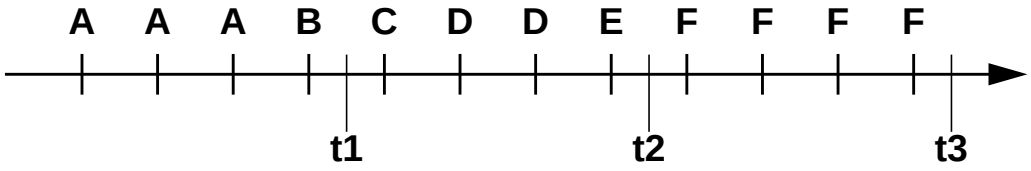
- **periodisches Reduzieren der Referenzzähler, um Gewicht früher Referenzen zu reduzieren**

  Reduzierung von RZ durch Division oder Subtraktion:

$$RZ(i) = \frac{RZ(i)}{K1} \qquad (K1 > 1)$$

  oder

$$RZ(i) = \begin{cases} RZ(i) - K2 & \text{falls} \quad RZ(i) - K2 \geq K3 \\ K3 & \text{sonst} \quad (K2 > 0, K3 \geq 0) \end{cases}$$



	t1	t2	t3
RZ(A)			
RZ(B)			
RZ(C)			
RZ(D)			
RZ(E)			
RZ(F)			

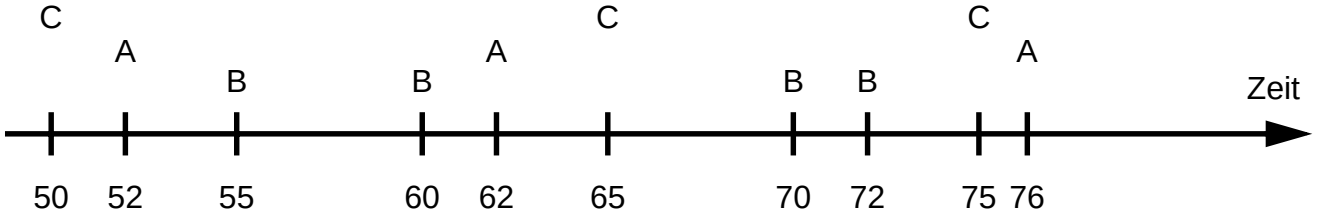
LRU-K

- **Aufzeichnung der K letzten Referenzzeitpunkte** (pro Seite im DB-Puffer)
  - Aufwendigere Aufzeichnung gewährleistet aktuelle Ersetzungsinformation; Methode benötigt kein explizites „Altern“ über Tuning-Parameter wie LRD-V2
  - Gegeben sei bis zum Betrachtungszeitpunkt t der Referenzstring $r_1, r_2, \ldots, r_t$. **Rückwärtige K-Distanz $b_t(P, K)$** ist die in Referenzen gemessene Distanz rückwärts bis zur K-jüngsten Referenz auf Seite P:

    $b_t(P, K) = x$, wenn r_{t-x} den Wert P besitzt und
    es genau K-1 andere Werte i mit $t-x < i \leq t$ mit $r_i = P$ gab.

    $b_t(P, K) = \infty$, wenn P nicht wenigstens K mal in $r_1, r_2, \ldots, r_t$ referenziert wurde

- **Beispiel (K=4)**



- **Zur Ersetzung genügt es, die $b_t(P_i, K)$ der Pufferseiten zu berücksichtigen!**
  - Sonderbehandlung für Seiten mit weniger als K Referenzen erforderlich
  - Wie hängt LRU-K mit LRD zusammen? Approximation der Referenzdichte?

- **LRU-2 (d.h. K=2) stellt i. allg. beste Lösung dar[1]**
  - ähnlich gute Ergebnisse wie für K > 2, jedoch einfachere Realisierung
  - Verfahren reagiert schneller auf Referenzschwankungen als bei größeren K

1. O‘Neil, E.J., O‘Neil, P.E., Weikum, G.: The LRU-K Page Replacement Algorithm for Database Disk Buffering. Proc. ACM SIGMOD Conf. Washington. D.C. 1993. 297–306

Ersetzungverfahren – Einbezug von Kontextwissen

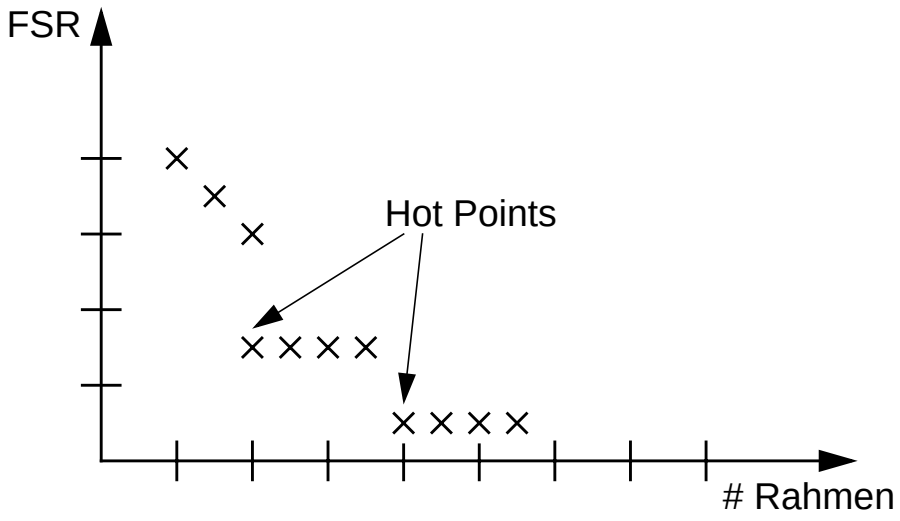
- **Ausnutzung von Kontextwissen bei mengenorientierten Anforderungen**

  ➥ Verbesserung in relationalen DBS möglich

- **Zugriffspläne durch Optimizer**
  - Zugriffscharakteristik/Menge der referenzierten Seiten kann bei der Erstellung von Plänen vorausgesagt/abgeschätzt werden
  - Zugriffsmuster enthält immer Zyklen/Loops (mindestens Kontrollseite – Datenseite, *nested loop join* etc.)
  - Kostenvoranschläge für Zugriffspläne können verfügbare Rahmen berücksichtigen
  - Bei Ausführung wird die Mindestrahmenzahl der Pufferverwaltung mitgeteilt

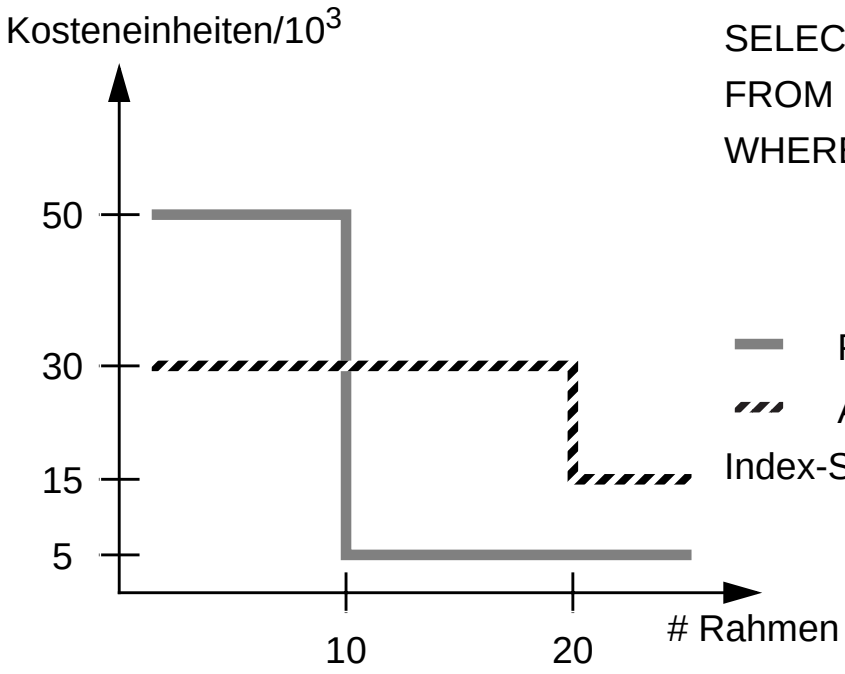
- **<u>Hot-Set</u>: Menge der Seiten im Referenzzyklus**

  Prinzipieller Verlauf der Fehlseitenrate (**FSR**) bei speziellen Operationen



„Hot-Set“-Modell

- ***Hot Point*:**
  abrupte Veränderung in der FSR, z. B. verursacht durch Schleife beim Verbund
- ***Hot Set Size* (HSS):**
  größter *Hot Point* kleiner als der verfügbare DB-Puffer
- Optimizer berechnet HSS für die verschiedenen Zugriffspläne (Abschätzung der #Rahmen)
- Beispiel:




```
SELECT   *
FROM     ABT X, PERS Y
WHERE    X.ANR = Y.ANR
         AND ...
```

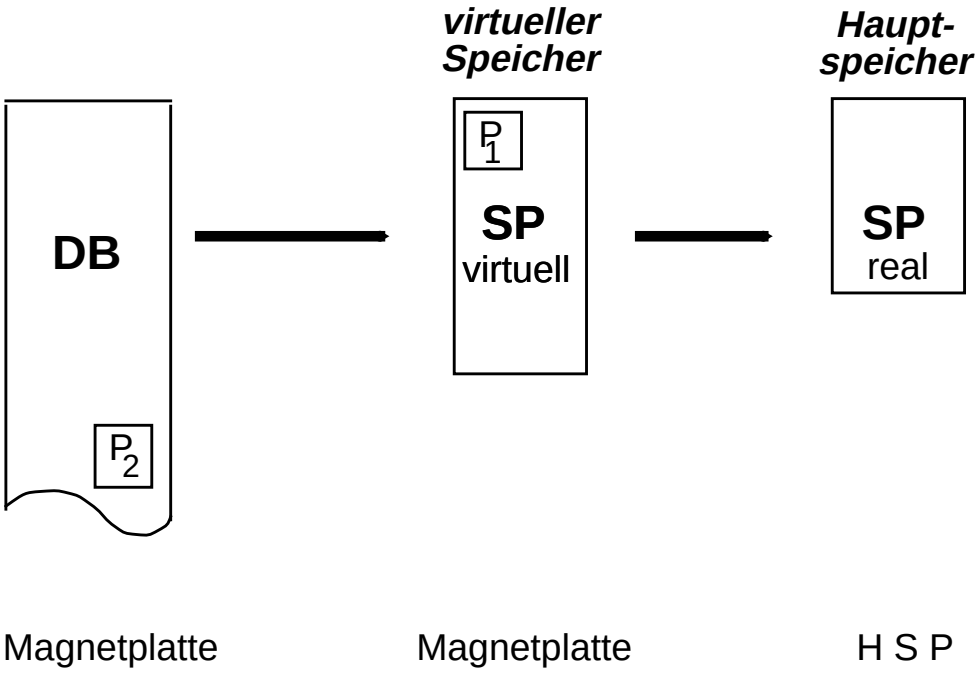
PERS in äußerer Schleife

ABT in äußerer Schleife

Index-Scan für beide Relationen

- **Anwendungscharakteristika**
  - Berücksichtigung der HSS in den Gesamtkosten
  - Auswahl abhängig von verfügbarer DB-Puffergröße
  - Bindung zur Laufzeit möglich

Seitenersetzung
bei virtuellem Speicher



- **Page Fault:**

  $P_i(P_1)$ in SP virtuell, aber nicht in SP real (HSP)

- **Database Fault:**

  $P_i(P_2)$ nicht in SP virtuell,
  Seitenrahmen für P_i jedoch in SP real

- **Double Page Fault:**

  $P_i(P_2)$ nicht in SP virtuell, ausgewählter
  Seitenrahmen nicht in SP real

Zusammenfassung

- **Referenzmuster in DBS sind Mischformen**
  - sequentielle, zyklische, wahlfreie Zugriff
  - Lokalität innerhalb und zwischen Transaktionen
  - „bekannte“ Seiten mit hoher Referenzdichte
- **Ohne Lokalität ist jede Optimierung** der Seitenersetzung sinnlos (~ RANDOM)
- **Suche im Puffer durch Hash-Verfahren**
- **Speicherzuteilung:**
  - global ⇒ alle Pufferrahmen für alle Transaktionen (Einfachheit, Stabilität ...)
  - lokal ⇒ Sonderbehandlung bestimmter Transaktionen/Anfragen/ DB-Bereiche
- **Behandlung geänderter Seiten:** NOFORCE, asynchrones Ausschreiben
- **Seitenersetzungsverfahren**
  - „zu genaue“ Verfahren sind schwierig einzustellen (⇒ instabil)
  - Nutzung mehrerer Kriterien: Alter, letzte Referenz, Referenzhäufigkeit
  - CLOCK ~ LRU, aber einfachere Implementierung
  - GCLOCK, LRD, LRU-K relativ komplex
  - LRU-2 guter Kompromiss; vorletzter Referenzzeitpunkt bestimmt „Opfer“
- **Erweiterte Ersetzungsverfahren**
  - Nutzung von Zugriffsinformationen des Query-Optimierers
  - Hot-Set-Modell
- **Double-Paging sollte vermieden werden**

Prioritätsgesteuerte Seitenersetzung[1]

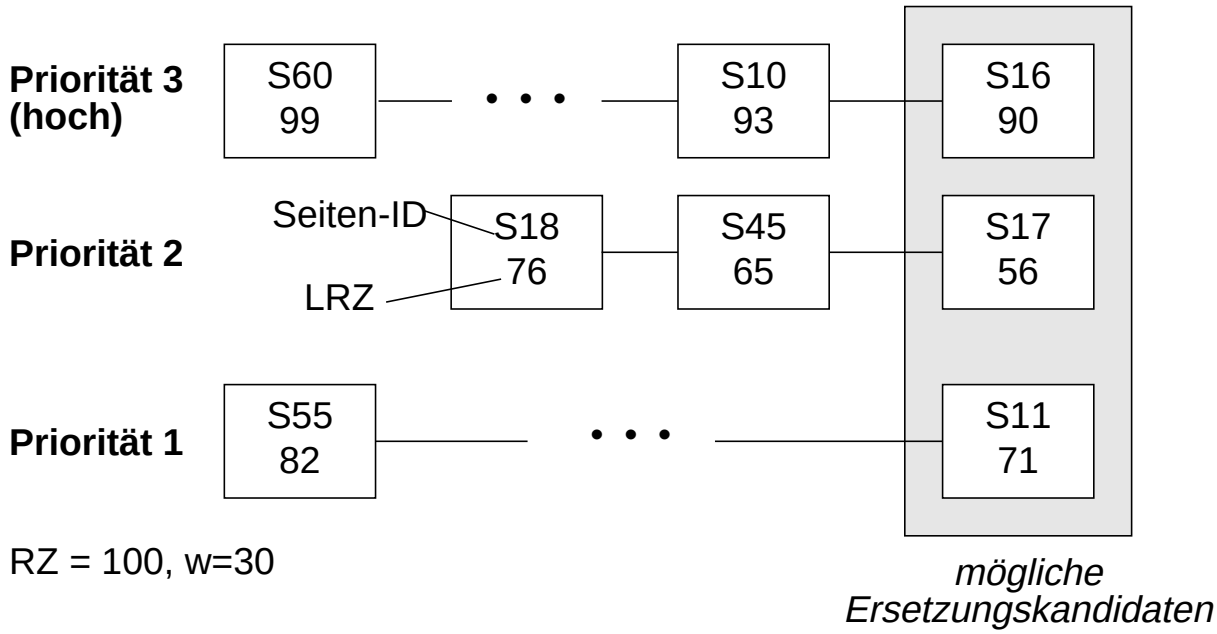
Bevorzugung bestimmter Transaktionstypen/DB-Partitionen vielfach wünschenswert (z. B. um Benachteiligungen durch sehr lange TA oder sequentielle Zugriffe zu vermeiden)

⇒ Berücksichtigung von Prioritäten bei der DB-Pufferverwaltung

- **Verfahren PRIORITY-LRU:**
  - pro Prioritätsstufe eigene dynamische Pufferpartition
  - LRU-Kette pro Partition
  - Priorität einer Seite bestimmt durch DB-Partition bzw. durch (maximale) Priorität referenzierender Transaktionen
  - ersetzt wird Seite aus der Partition mit der geringsten Priorität

    Ausnahme: die w zuletzt referenzierten Seiten sollen (unabhängig von ihrer Priorität) nicht ersetzt werden

    ⇒ Kompromiß zwischen Prioritäts- und absolutem LRU-Kriterium möglich



1. R. Jauhari, M.J. Carey, M. Livny: Priority-Hints: An Algorithm for Priority-Based Buffer Management. Proc. 16th VLDB Conf., 1990, pp. 708-721

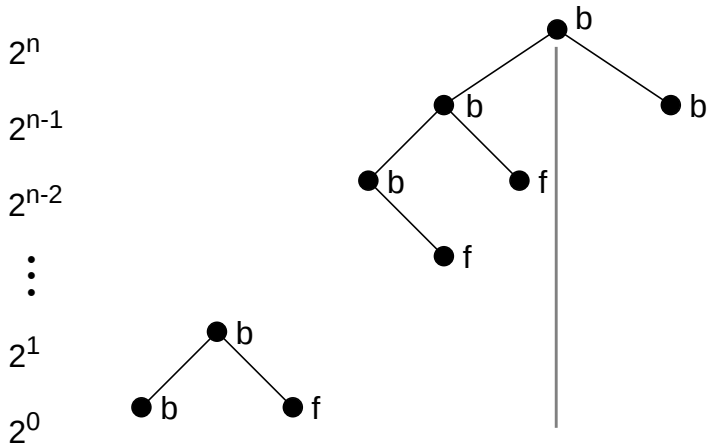
DB-Pufferverwaltung
bei Seiten variabler Größe

- **Seitengröße**
  - keine beliebigen Seiten (Fragmentierung, Abbildung auf Externspeicher)
  - Seitenlänge (SL) als Vielfaches eines Einheitsrasters (Transporteinheit, Rahmengröße im DB-Puffer)
  - Bsp.: SL = 1, 2, 4, …, 2^n · Rahmengröße $(n \leq 8)$

- **Trennung von Seite und Rahmen (Rastergröße)**

- **Schnittstellenforderung: Zusammenhängende Speicherung der Seite im DB-Puffer**

- **Buddy-System (BS-Algorithmus)**



  - Verwaltung variabler Bereiche: frei(f) / belegt(b)
    (festes Raster, hierarchischer Vergabemechnismus)
  - Suche nur in freien Bereichen (belegte Bereiche können nicht verschoben werden)
  - Zusammenfassung von freien Neffen aufwendig

Zusätzliche Anforderungen an DB-Pufferverwaltung

- **Zustände von Seiten/Rahmen**
  - frei: keine Seite vorhanden
  - unfixed: Seite ist ersetzbar/verschiebbar
  - fixed: Seite muß Pufferadresse behalten
- **Vermeidung von Seitenersetzungen**
  - Umlagern von Seiten mit Unfix-Vermerk und „hoher“ Wiederbenutzungs-wahrscheinlichkeit
- **Suche nach „bestem“ Ersetzungskandidaten**
- **Was heißt „bester“ Ersetzungskandidat?**
  - Wiederbenutzungs-WS: Anzahl der Referenzen, Alter, letzte Referenz einer Seite
  - Fragmentierung/Lückenbenutzung: *first fit*, *best fit*
  - Rahmeninhalte: Anteil ersetzbarer und freier Rahmen
  - Anzahl der zu ersetzenden Seiten zur Platzbeschaffung für eine neue Seite
- **Kombination von Ersetzung und Umlagern von Seiten**

  $\Rightarrow$ sehr komplexe Entscheidungssituation
- **Alternative: Partitionierung des DB-Puffers für Seiten gleicher Größe (und anwendungsabhängige Ersetzungsverfahren)**

DB-Pufferverwaltung

- **Belegungsbeispiel bei Seiten variabler Größe und Rahmen fester Größe**

- **Probleme:**
  - Puffer-Fragmentierung

  - Ersetzung mehrerer Seiten zur Erfüllung einer neuen Seitenanforderung

- **Ziele:**
  - Maximierung der Pufferbelegung (Speicherplatzoptimierung)
  - Minimierung der E/A durch Berücksichtigung von Lokalität im Referenzverhalten

- **Vorschlag für einen Algorithmus: VAR-PAGE-LRU[1]**

- **Neue Aufgabenstellung:**
  Caching und spekulatives Prefetching von Dokumenten in Speicherhierarchien (Tertiärspeicher)[2]

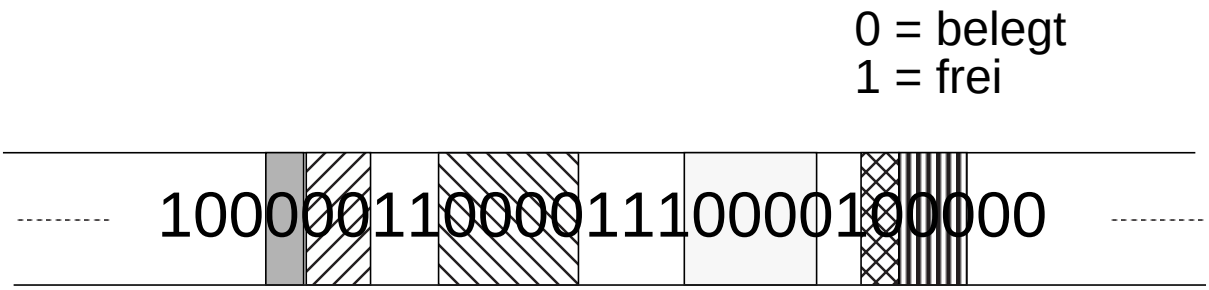
---

1. A. Sikeler: *VAR-PAGE-LRU—A Buffer Replacement Algorithm Supporting Different Page Sizes.* in Proc. Extending Database Technology, 1988, Springer, LNCS 303, pp. 336-351
2. A. Kraiss, G. Weikum: *Integrated Document Caching and Prefetching in Storage Hierarchies Based on Markov-Chain Predictions*, to appear in: VLDB Journal, 1998.

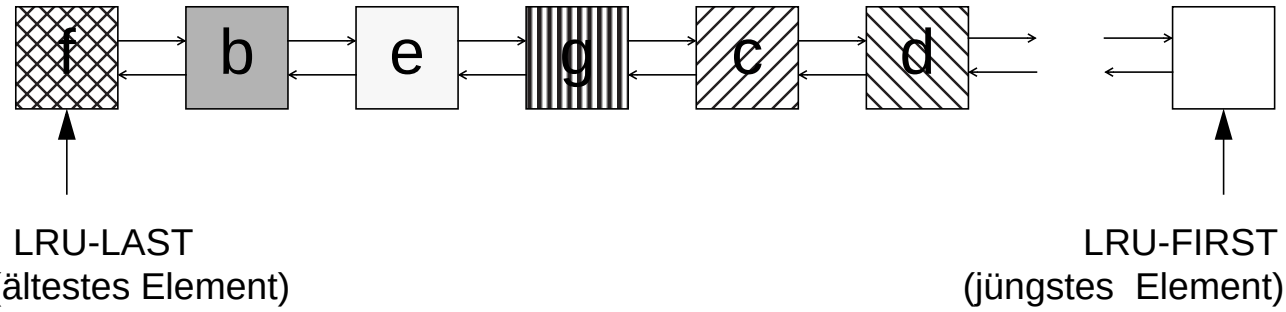
Der Algorithmus VAR-PAGE-LRU[1]

Datenstrukturen:

Freiliste:



LRU-Kette:



Algorithmus:

Kopiere Freiliste in eine Arbeitsliste

Schritt 1: Bestimme ausreichend viele, zusammenhängende Seitenrahmen zur Aufnahme der neuen Seite

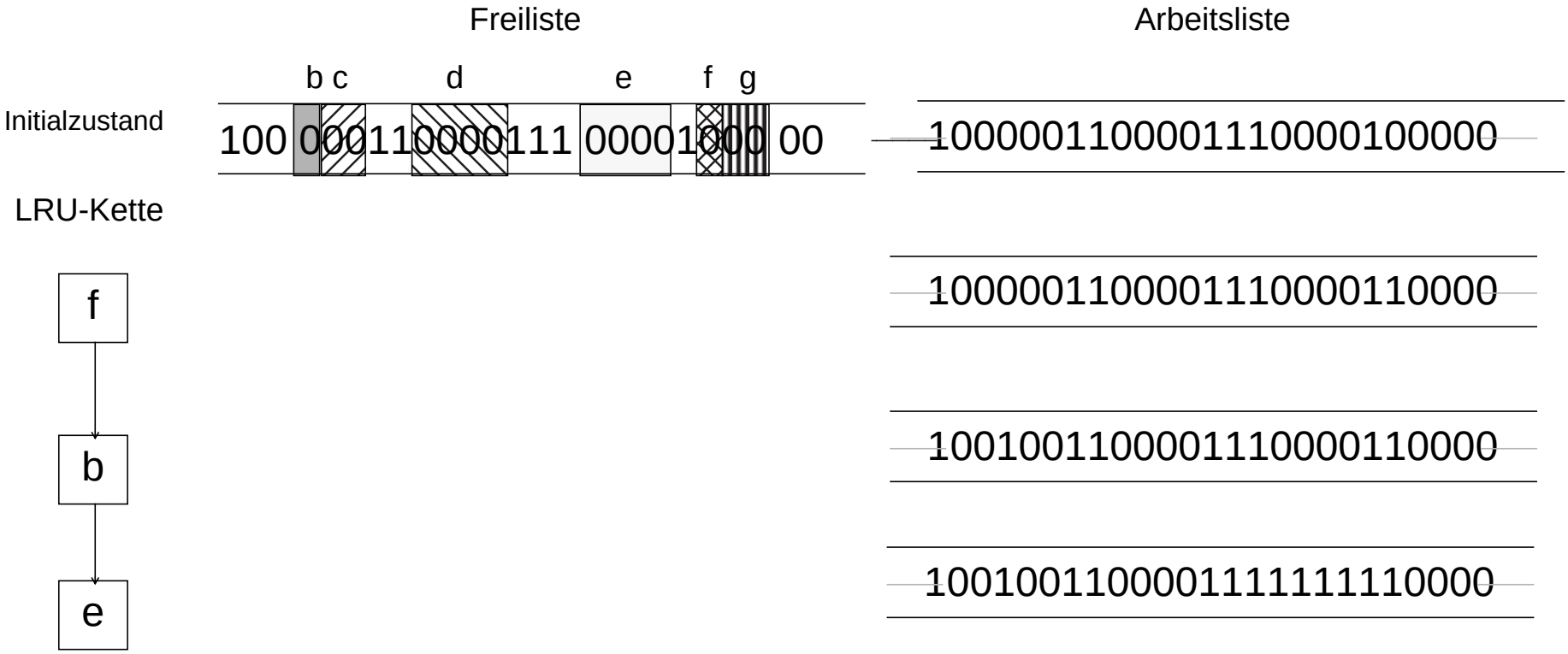
Schritt 2: Bestimme zu ersetzende Seiten und verdränge sie

Lies neue Seite in den Puffer ein

1. A. Sikeler: *VAR-PAGE-LRU - A Buffer Replacement Algorithm Supporting Different Page Sizes.* in Proc. Extending Database Technology, 1988, Springer, LNCS 303, pp. 336-351

VAR-PAGE-LRU: Schritt 1

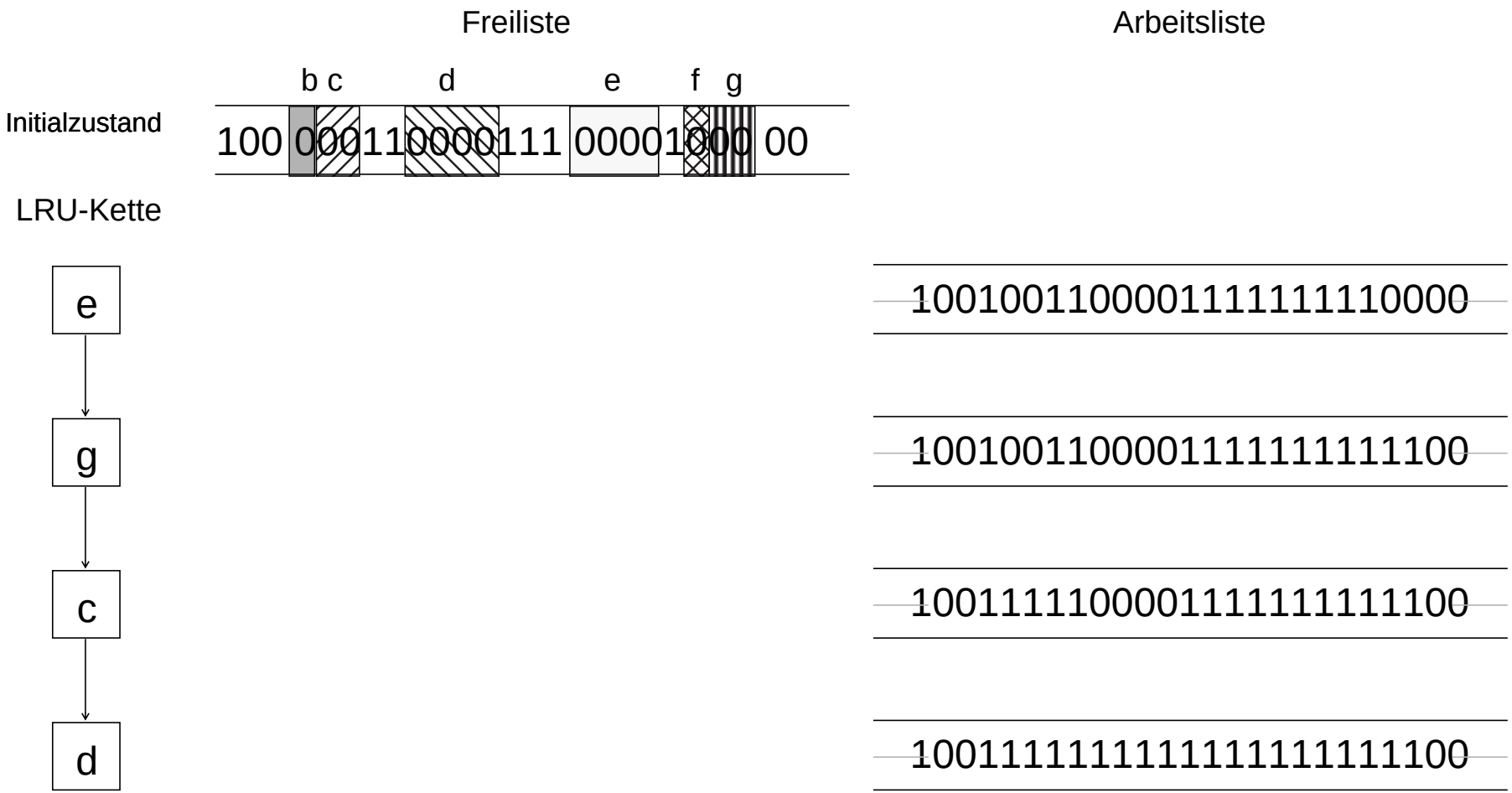
Suche nach 8 freien, zusammenhängenden Rahmen



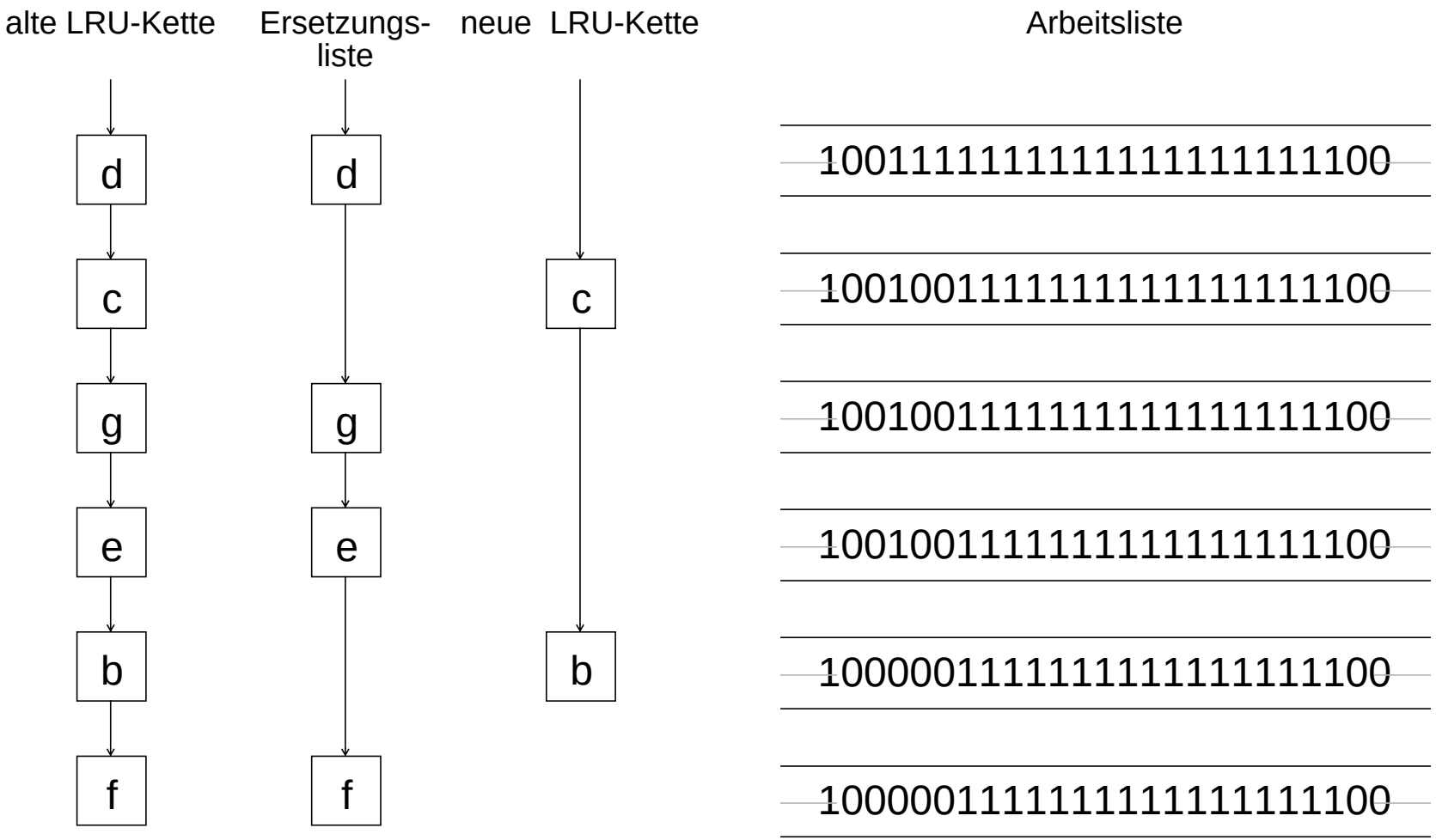
=> Ersetzung von Seite e ausreichend

VAR-PAGE-LRU: Schritt 1 (Forts.)

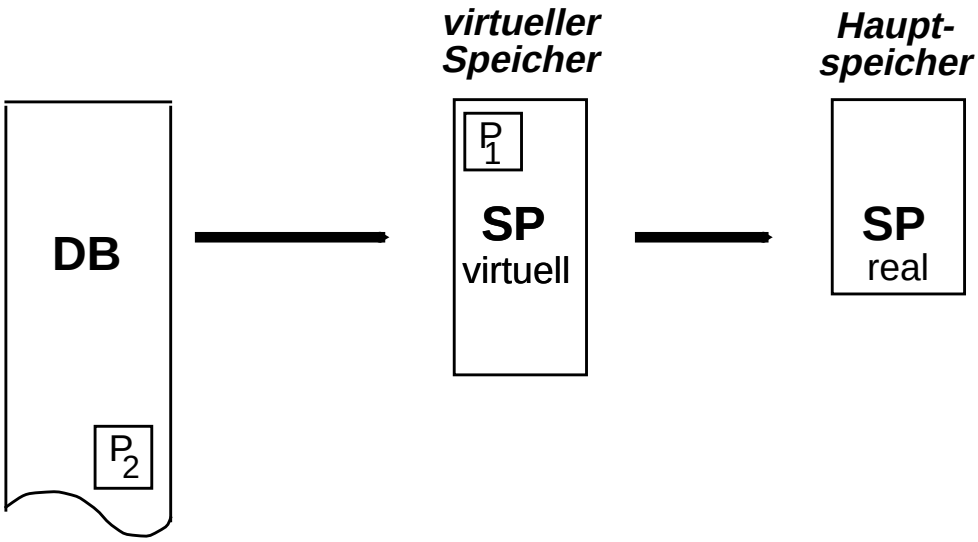
Suche nach 16 freien, zusammenhängenden Rahmen



VAR-PAGE-LRU: Schritt 2
Bestimmung der tatsächlich zu ersetzenden Seiten
(bei der Anforderung von 16 Rahmen)



Seitenersetzung bei virtuellem Speicher



- **Page Fault:**

  $P_i(P_1)$ in SP virtuell, aber nicht in SP real (HSP)

- **Database Fault:**

  $P_i(P_2)$ nicht in SP virtuell,
  Seitenrahmen für P_i jedoch in SP real

- **Double Page Fault:**

  $P_i(P_2)$ nicht in SP virtuell, ausgewählter
  Seitenrahmen nicht in SP real

Zusammenfassung

- **Referenzmuster in DBS sind Mischformen**
  - sequentielle, zyklische, wahlfreie Zugriff
  - Lokalität innerhalb und zwischen Transaktionen
  - "bekannte" Seiten mit hoher Referenzdichte
- **Ohne Lokalität ist jede Optimierung der Seitenersetzung sinnlos (~ RANDOM)**
- **Suche im Puffer durch Hash-Verfahren**
- **Speicherzuteilung:** global ⇒ alle Pufferrahmen für alle Transaktionen (Einfachheit, Stabilität ...)
- **Behandlung geänderter Seiten:** NOFORCE, asynchrones Ausschreiben
- **Seitenersetzungsverfahren**
  - "zu genaue" Verfahren sind schwierig einzustellen (⇒ instabil)
  - Nutzung mehrerer Kriterien: Alter, letzte Referenz, Referenzhäufigkeit
  - CLOCK ~ LRU, aber einfachere Implementierung
  - GCLOCK, LRD relativ komplex
- **Erweiterte Ersetzungsverfahren**
  - Nutzung von Zugriffsinformationen des Query-Optimierers (Hot-Set-Modell)
  - Berücksichtigung von Prioritäten
- **Ersetzung bei Seiten variabler Größe**
  - komplexe Algorithmen
  - pro Seitengröße ein Puffer: einfach, aber i.a. schlechtere Speichernutzung
- **Double-Paging sollte vermieden werden**

Ersetzungverfahren – Einbezug von Kontextwissen

- **Probleme bei LRU-ähnlichen Verfahren**
  - T1: langer sequentieller Scan mit sehr schneller Seitenanforderung

    Auswirkung auf T_i: Seiten der T_i werden bei „langsamer“ Anforderung verdrängt – auch bei hoher Referenzlokalität

  - Zyklisches Referenzieren (*Loop*) einer Menge von Seiten (#Seiten > #Rahmen)

    $\Rightarrow$ internes Thrashing

  - T1: zyklisches Referenzieren einer Seitenmenge (#Seiten < #Rahmen)

    Interferenz durch T_i bei schnellerer Anforderung (*stealing*)

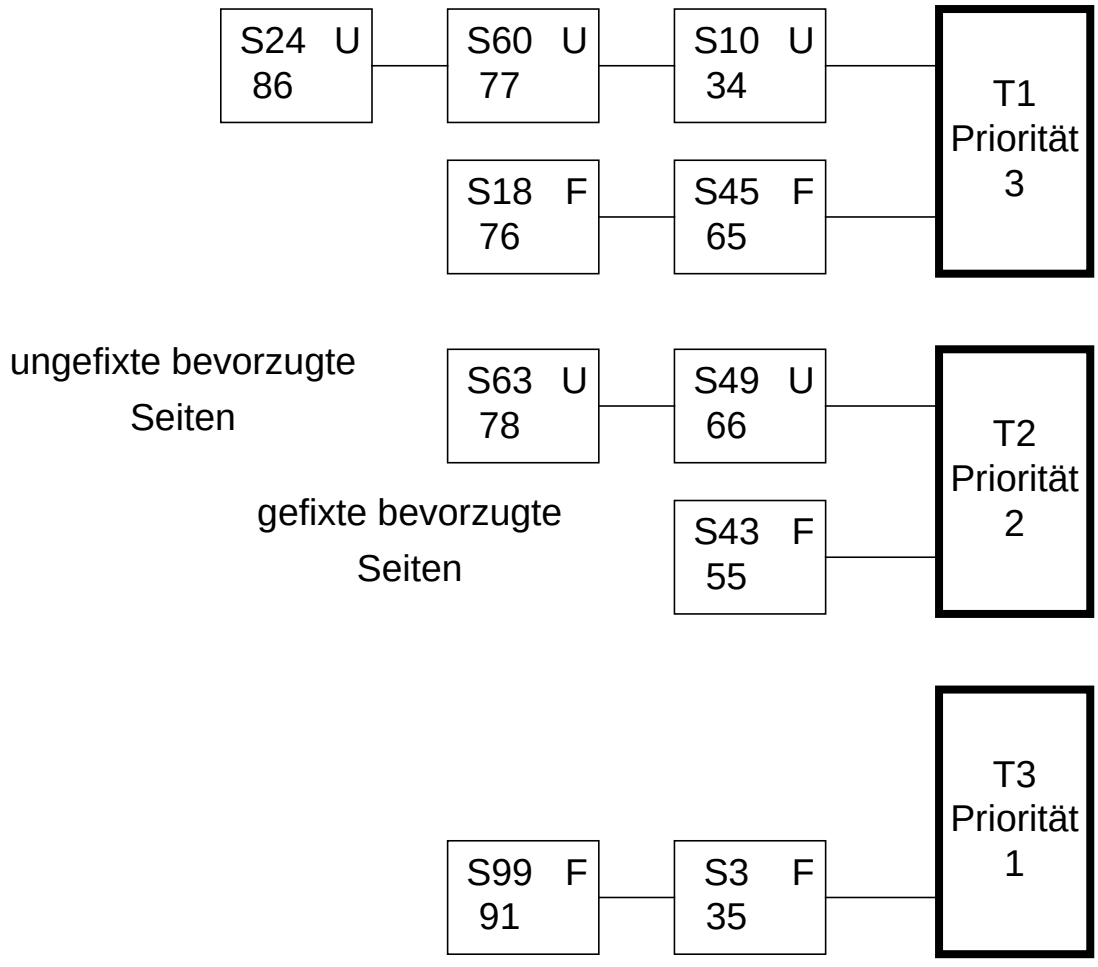
    $\Rightarrow$ externes Thrashing

- **Mechanismen gegen Thrashing: WS-Modell**
  - Scheduler versucht für T_i wenigstens $\sigma = W(t, w)$ Rahmen zu allokieren
  - Wenn Loop > w, versucht WS w Rahmen zu reservieren
  - Bei sequentiellem Scan hätte ein Rahmen genügt
  - WS-Modell: teure Implementierung

Prioritätsgesteuerte Seitenersetzung (2)

- **Verfahren PRIORITY-HINTS:**
  - Unterscheidung zwischen **bevorzugten** und **normalen** Seiten (z.B. zyklisch referenzierte Seiten sollen bevorzugt werden)
  - bei FIX-Aufruf wird angegeben, ob Seite bevorzugt werden soll
  - normale Seiten werden vorrangig ersetzt (z.B. gemäß globaler LRU-Strategie)
  - bevorzugte Seiten werden transaktionsspezifisch verwaltet (TA-bezogene, dynamische Pufferpartitionen)
  - Ersetzung von bevorzugten Seiten erfolgt:
    - prioritätsgesteuert
    - gemäß MRU (most recently used) innerhalb einer Prioritätsstufe

Sind keine bevorzugten Seiten geringerer Priorität ersetzbar, wird eine Seite aus dem TA-spezifischen Puffer verdrängt



Seitenreferenzstrings

- **jede Datenanforderung ist eine *logische Seitenreferenz***

- **Aufgabe der DB-Pufferverwaltung:**
  Minimierung der *physischen Seitenreferenzen*

- **Referenzstring $R = \langle r_1, r_2, \ldots r_i, \ldots r_n \rangle$**
  mit $r_i = (T_i, D_i, S_i)$

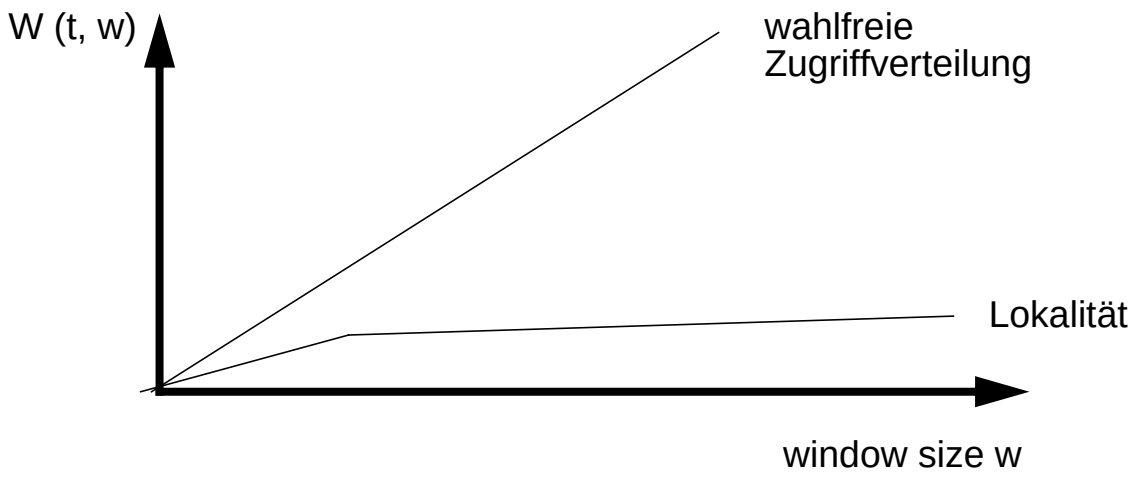
  T_i zugreifende Transaktion

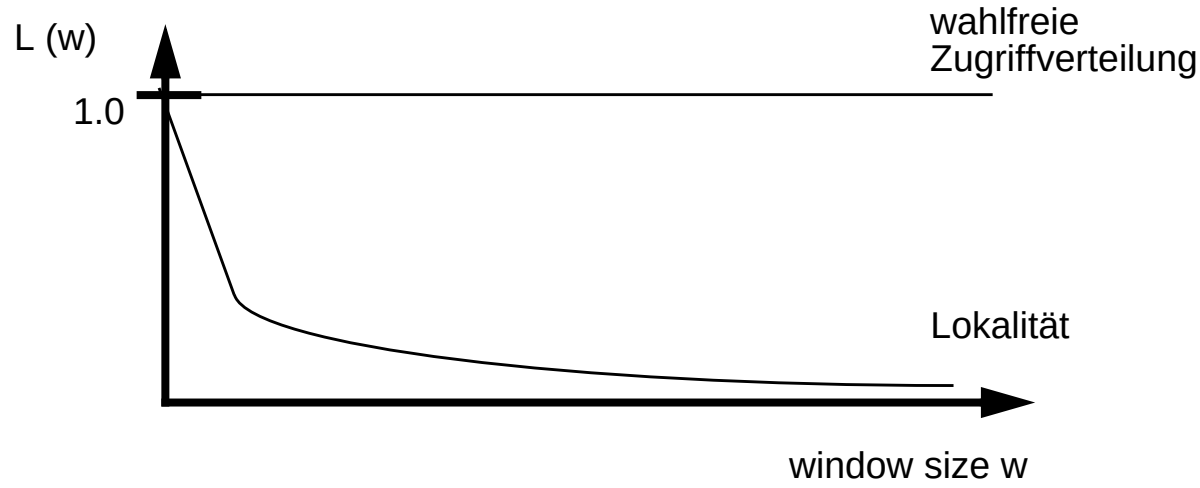
  D_i referenzierte DB-Partition

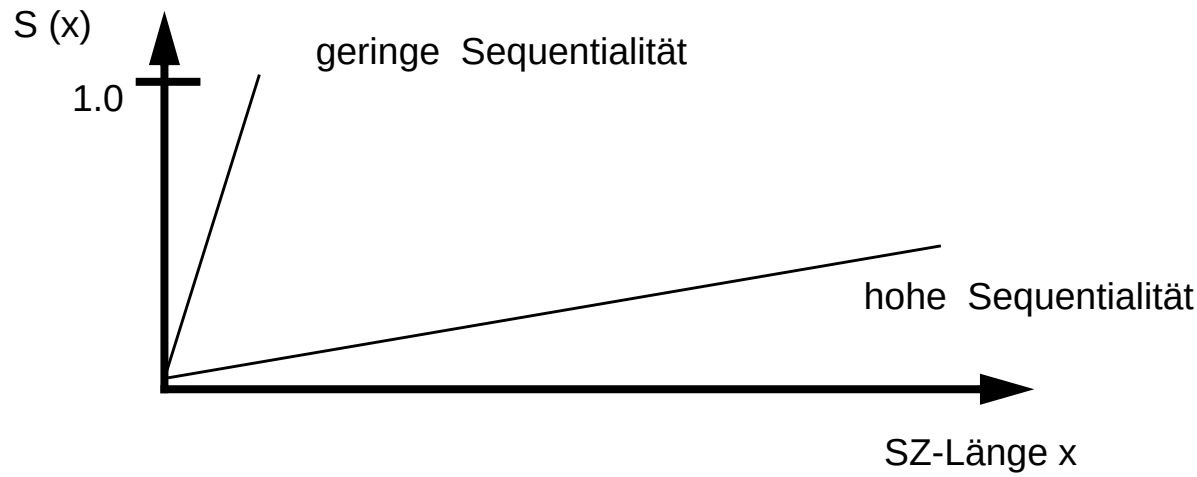
  S_i referenzierte DB-Seite

- Bestimmung von Ausschnitten aus R bezüglich bestimmter Transaktionen, Transaktions-Typen und DB-Partitionen sinnvoll zur Analyse des Referenzverhaltens

- **Wie kann Referenzstring-Information verwendet werden für**
  - Charakterisierung des Referenzverhaltens ?
  - Bestimmung von Lokalität und Sequentialität ?
  - Unterstützung einer effektiven Seitenersetzung ?


W (t, w)
wahlfreie
Zugriffverteilung
Lokalität
window size w


L (w)
wahlfreie
Zugriffverteilung
1.0
Lokalität
window size w


S (x)
geringe Sequentialität
1.0
hohe Sequentialität
SZ-Länge x

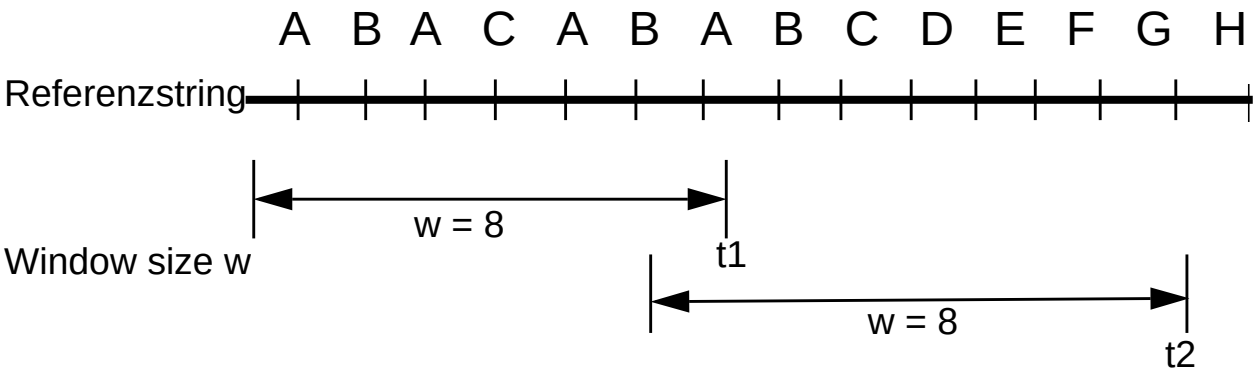
Lokalität

- **erhöhte Wiederbenutzungswahrscheinlichkeit für gerade referenzierte Seiten (gradueller Begriff)**

- **grundlegende Voraussetzung für**
  - effektive DB-Pufferverwaltung (Seitenersetzung)
  - Einsatz von Speicherhierarchien

- **Wie kann man Lokalität messen ?**

Working-Set-Modell



Working set size W W (t1, w=8) = 3 W (t2, w=8) = 8

Aktuelle Lokalität: $AL(t, w) = \frac{W(t, w)}{w}$

Durchschnittliche Lokalität: $L(w) = \frac{\sum_{t=1}^{n} AL(t, w)}{n}$

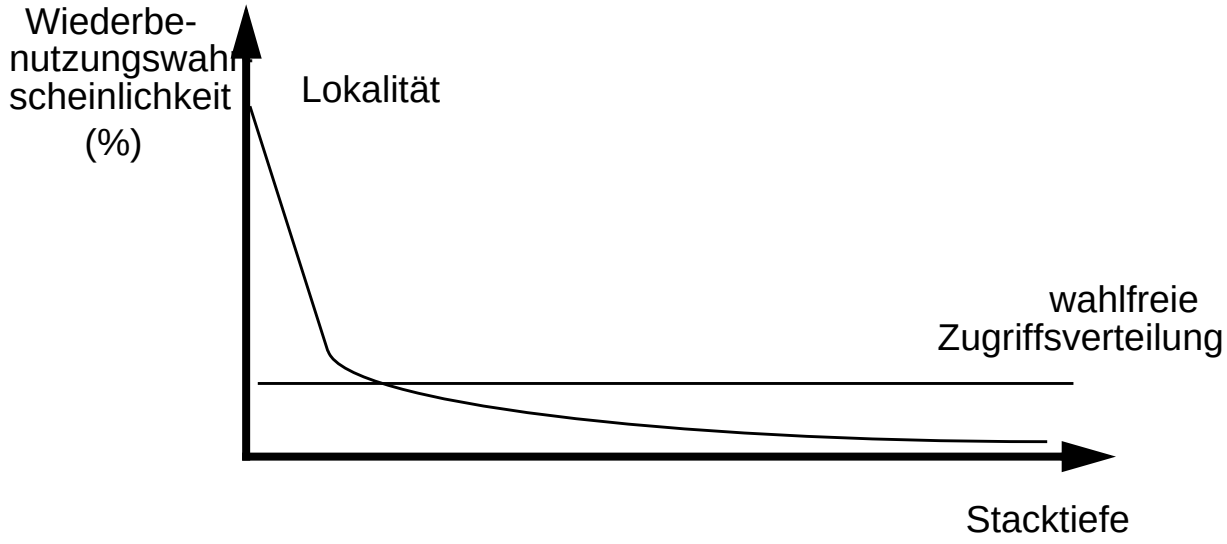
LRU-Stacktiefenverteilung

- **Maß für die Lokalität (präziser als Working-Set-Ansatz)**

- **LRU-Stack enthält alle bereits referenzierten Seiten in der Reihenfolge ihres Zugriffsalters**

- **Bestimmung der Stacktiefenverteilung:**
  - pro Stackposition wird Zähler geführt
  - Rereferenz einer Seite führt zur Zählererhöhung für die jeweilige Stackposition

  $\Rightarrow$ Zählerwerte entsprechen der Wiederbenutzungshäufigkeit



Für LRU-Seitenersetzung kann aus der Stacktiefenverteilung für eine bestimmte Puffergröße unmittelbar die Trefferrate (bzw. Fehlseitenrate) bestimmt werden

LRU-Stacktiefen-Verteilungen für reale Seitenreferenzstrings[1]

1. W. Effelsberg, T. Härder: Principles of Database Buffer Management, ACM Transactions on Database Systems, Vol. 9, No. 4, Dec. 1984, pp. 560-595.

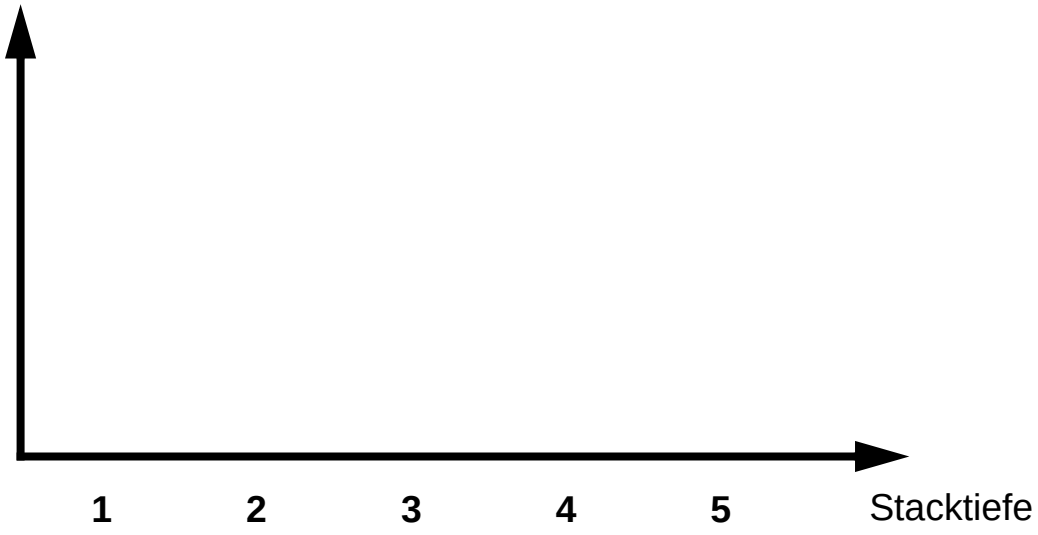
Beispiel: Ermittlung der Stacktiefenverteilung

Referenzstring: A B A C A A A B B B C D E A E

LRU-Stack:

1	**A**
2	**B**
3	**C**
4	**D**
5	**E**

Stacktiefenverteilung



Referenzdichte-Kurven

Relative Häufigkeit der Seitentypen im Beispiel:
(n=Länge des Referenzstrings)
FPA = 0.1 %
DBTT = 6.1 %
USER = 93.8 % (Zugriffspfaddaten und Sätze)

Sequentialität

- **aufeinanderfolgende Zugriffe referenzieren benachbarte DB-Seiten**

- **sequentielle Zugriffsfolge (SZ):**

  zwei aufeinanderfolgende Referenzen r_i und r_{i+1} gehören zu einer sequentiellen Zugriffsfolge, falls

  $S_{i+1} - S_i = 0$ oder 1

- **Länge einer sequentiellen Zugriffsfolge:**

  Anzahl der in der SZ referenzierten Seiten

  Bsp.: Referenzstring A A B B D E E F F H
  enthält SZ der Länge 2 (AABB), der Länge 3 (DEEFF) und 1 (H)

- **Maß für Sequentialität:**

  kumulative Verteilung der SZ-Längen

  $S(x) = Pr$ (SZ-Länge <= x)

  für obiges Bsp. gilt: S(1)=0.33, S(2)=0.67, S(3)=1.0

- **bei Sequentialität Optimierung durch (asynchrones) Prefetching von DB-Seiten möglich**

Dynamische Pufferallokation (2)

2. Page-Fault-Frequency-Ansatz (PFF)

- **Vorgabe einer Soll-Fehlseitenrate F**

- **bei Fehlseitenbedingung wird aktuelle Fehlseitenrate FA (P) ausgewertet:**

  gilt FA (P) > F, wird Partition um eine Seite erweitert
  bei FA (P) < F wird eine Seite abgegeben

- **entspricht einer speziellen WS-Strategie mit Fenstergröße**

  w' = 1/F

  falls Abstand a zwischen zwei Fehlseitenbedingungen > w':

  - wird Working-Set beibehalten
  - Seiten, die vor der letzten Fehlseitenbedingung referenziert wurden, werden freigegeben

ansonsten (a < w') werden zusätzliche Rahmen bereitgestellt

Simulation von Seitenersetzungsverfahren[1]

- Charakteristika von DB, Transaktionslast und logischen Seitenreferenzstrings

1. Effelsberg, W., Härder, T.: Principles of Database Buffer Management, in: ACM Transactions on Database Systems 9:4, Dec. 1984, pp. 560-595.

Simulationsergebnisse