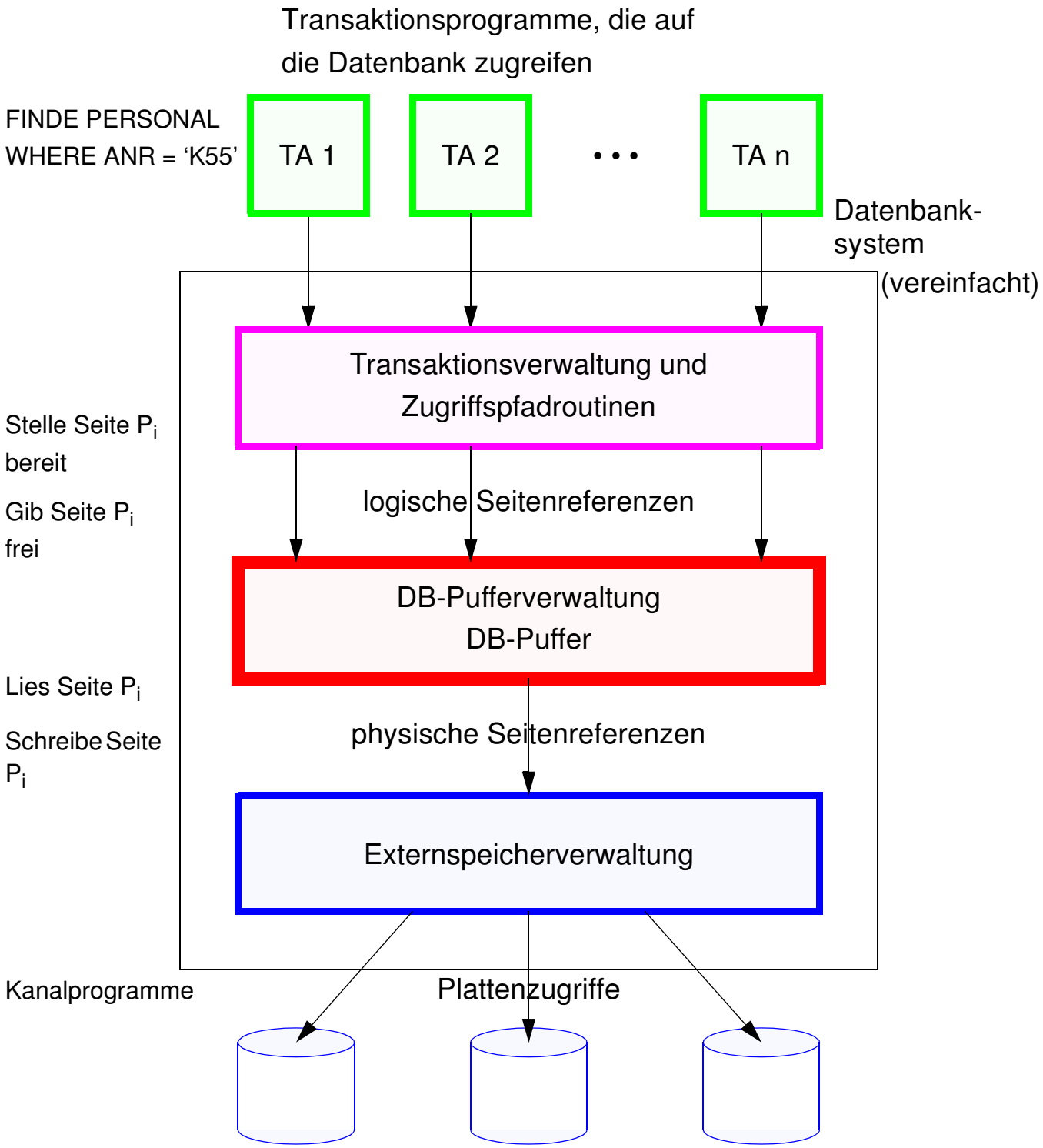
3. DB-Pufferverwaltung

- **Ziel: Realisierung einer effizienten, seitenbasierten Verarbeitungsplattform im Hauptspeicher**
  - größtmögliche Vermeidung von physischer Ein-/Ausgabe
  - Ersetzungsverfahren ohne und mit Kontextwissen
- **Rolle der DB-Pufferverwaltung**[1]
  - Ablauf des Zugriffs auf den DB-Puffer
  - Vergleich mit ähnlicher Funktionalität in Betriebssystemen (BS)
- **Lokalität**
  - Maße für Lokalität
  - Charakterisierung durch LRU-Stacktiefen-Verteilung und Referenzdichtekurven
- **Speicherzuteilung und Suche im DB-Puffer**
- **Seitenersetzungsverfahren**
  - Klassifikation von Ersetzungsverfahren
  - LRU
  - CLOCK, GCLOCK
  - LRD, LRU-K ...
- **Ersetzungsverfahren – Einbezug von Kontextwissen**
  - Hot Set Model
  - Prioritätsgesteuerte Seitenersetzung (Priority LRU, Priority Hints)

---

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



Seitenreferenzstrings

- **Jede Datenanforderung ist eine *logische Seitenreferenz***

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

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

  T_i zugreifende Transaktion

  D_i referenzierte DB-Partition

  S_i referenzierte DB-Seite

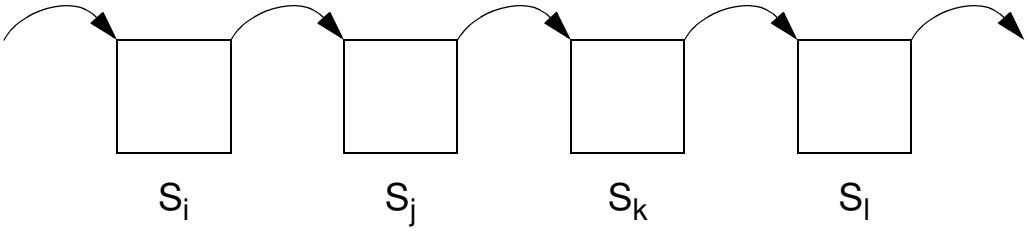
- Bestimmung von Ausschnitten aus R bezüglich bestimmter Transaktionen, Transaktions-Typen und DB-Partitionen sinnvoll zur Analyse des Referenzverhaltens

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

Eigenschaften von DB-Referenzstrings

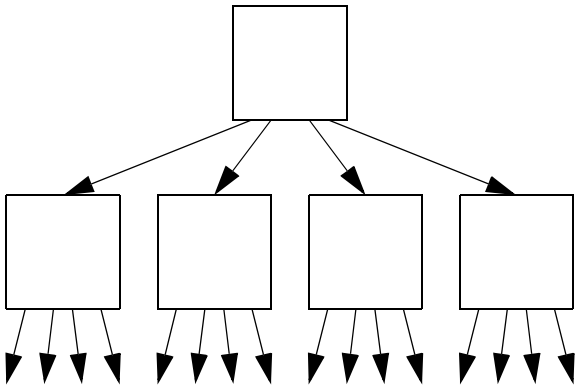
- **Typische Referenzmuster in DBS**

1. Sequentielle Suche



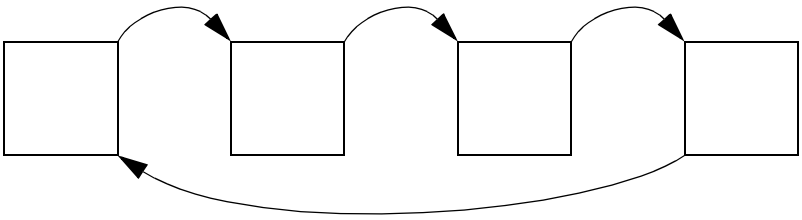
Bsp.: Durchsuchen ganzer Satztypen (Relationen)

2. Hierarchische Pfade



Bsp.: Suchen mit Hilfe von B*-Bäumen

3. Zyklische Pfade



Bsp.: Abarbeiten von Sets ((1:n)-Beziehungen),
Suchen in DBTT-/Datenseiten

Vergleich mit BS-Funktionen

- **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 BS als DB-Puffer eingesetzt werden?**

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

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

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

3. **Normale Dateisysteme bieten keine geeignete Schnittstelle für Prefetching**

   In DBMS ist aufgrund von Seiteninhalten oder Referenzmustern eine Voraussage des Referenzverhaltens (z. B. bei Tabellen-Scans) möglich; Prefetching erzielt in solchen Fällen eine enorme Leistungssteigerung

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

➥ DBMS muss eigene Pufferverwaltung realisieren,
64-Bit-Architekturen ändern daran nichts!

Sequentialität

- SRS weisen typischerweise **Phasen von Sequentialität und Lokalität** auf

- **Sequentielle Zugriffsfolge (SZ):**

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

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

  d. h., aufeinanderfolgende Zugriffe referenzieren benachbarte DB-Seiten

- **Algorithmus**

  - Seitenreferenzstring wird vollständig durchmustert; alternativ kann die Folge der ankommenden Referenzen analysiert werden
  - Solange obige Bedingung erfüllt ist,
    gehören alle aufeinanderfolgenden Referenzen zu einer SZ,
    sonst beginnt eine neue SZ

- **Länge einer sequentiellen Zugriffsfolge (LSZ):**

  - LSZ ist die Anzahl der verschiedenen in SZ referenzierten Seiten
  - Beispiel: Referenzstring
    A A B B D E E F F H enthält
    (AABB) mit LSZ(1) = 2, (DEEFF) mit LSZ(2) = 3 und (H) mit LSZ(3) = 1

- **Maß für Sequentialität:**

  - Die kumulative Verteilung der SZ-Längen LSZ(i) wird berechnet

    $$S(x) = \Pr(\text{SZ-Länge} <= x)$$

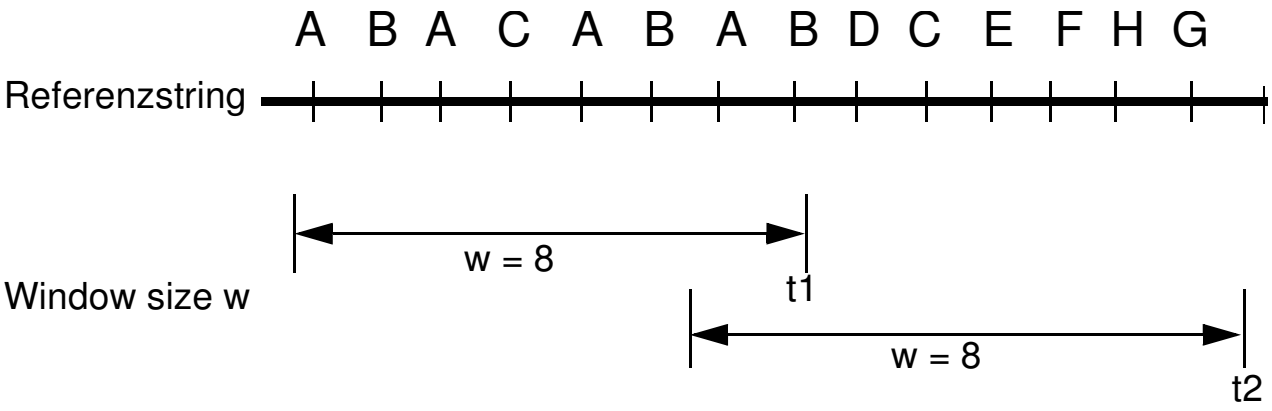
  - Für obiges Beispiel gilt: S(1)=0.33, S(2)=0.67, S(3)=1.0

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

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

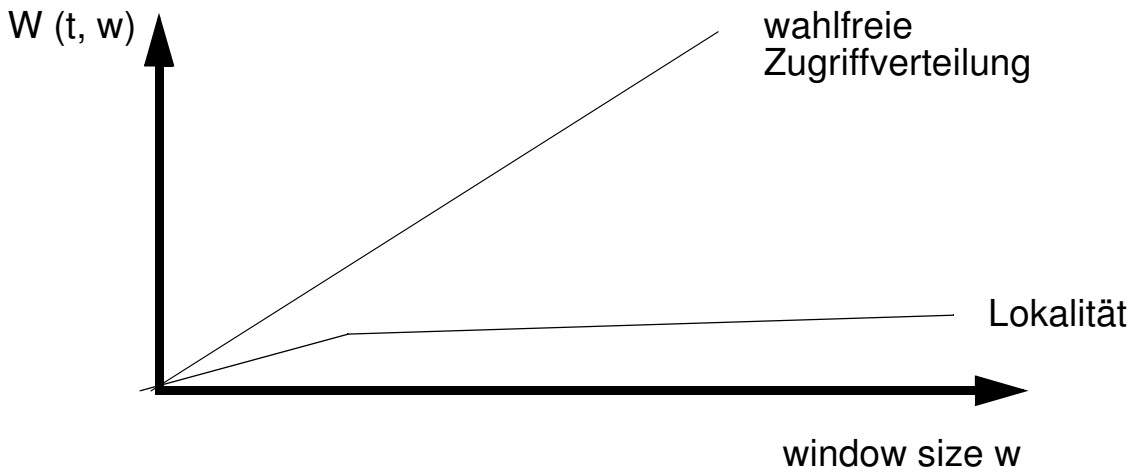


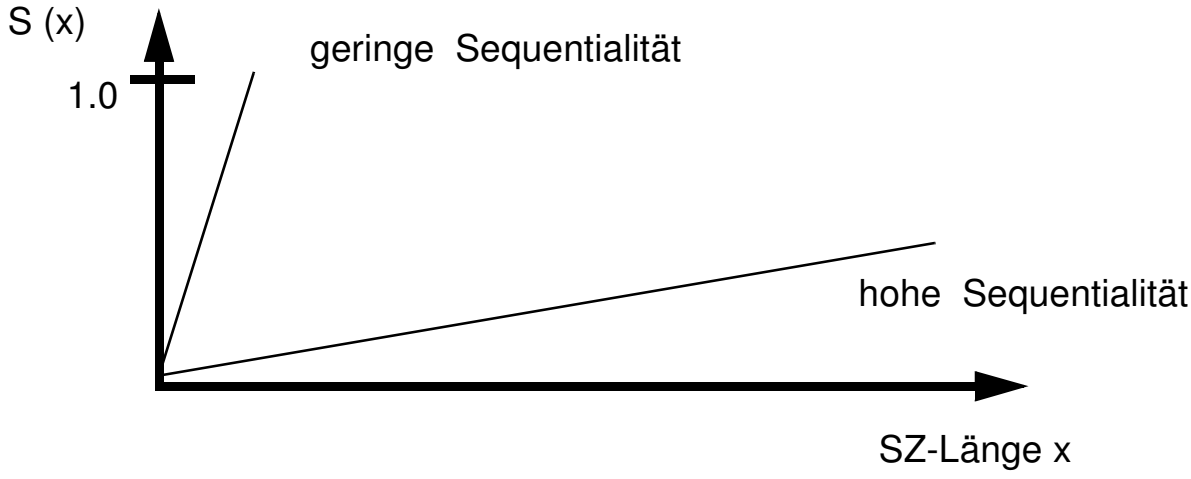
Working set size W W (t1, w=8) = 3 W (t2, w=8) = 8

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

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

Sequentialität vs. Lokalität





Relative Referenzmatrix (DOA-Last)

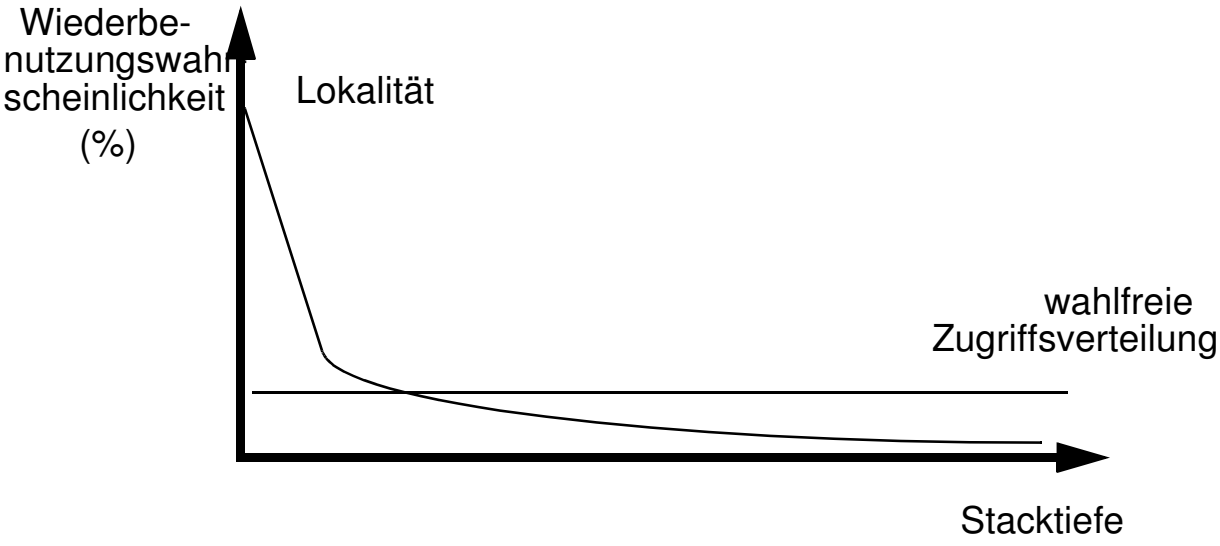
ca. 17 500 Transaktionen, 1 Million Seitenreferenzen auf ca. 66 000 verschiedene Seiten

	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

LRU-Stacktiefenverteilung

- **Wie läßt sich Lokalität charakterisieren?**
  - LRU-Stacktiefenverteilung liefert 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



➥ **Zählerwerte entsprechen der Wiederbenutzungshäufigkeit**

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

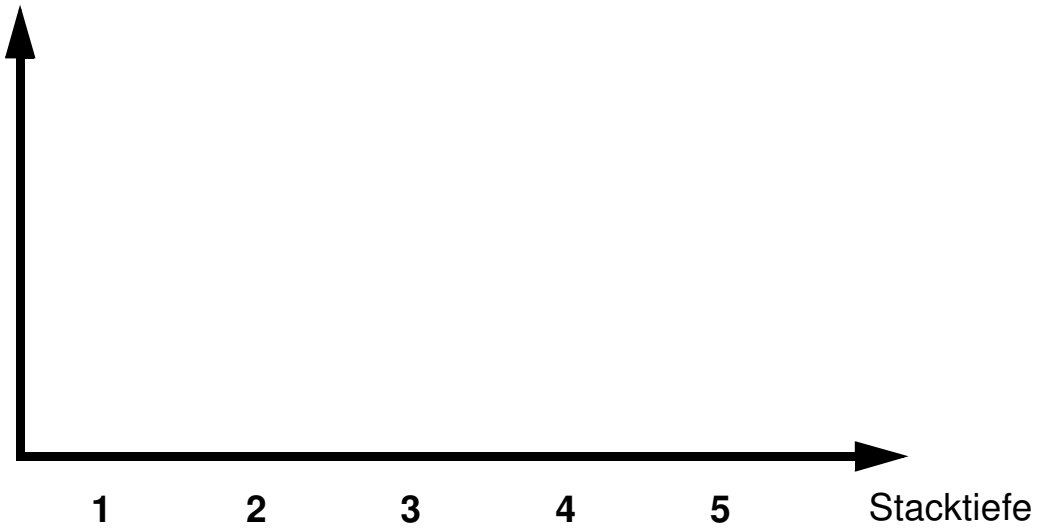
Beispiel: Ermittlung der Stacktiefen-Verteilung

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

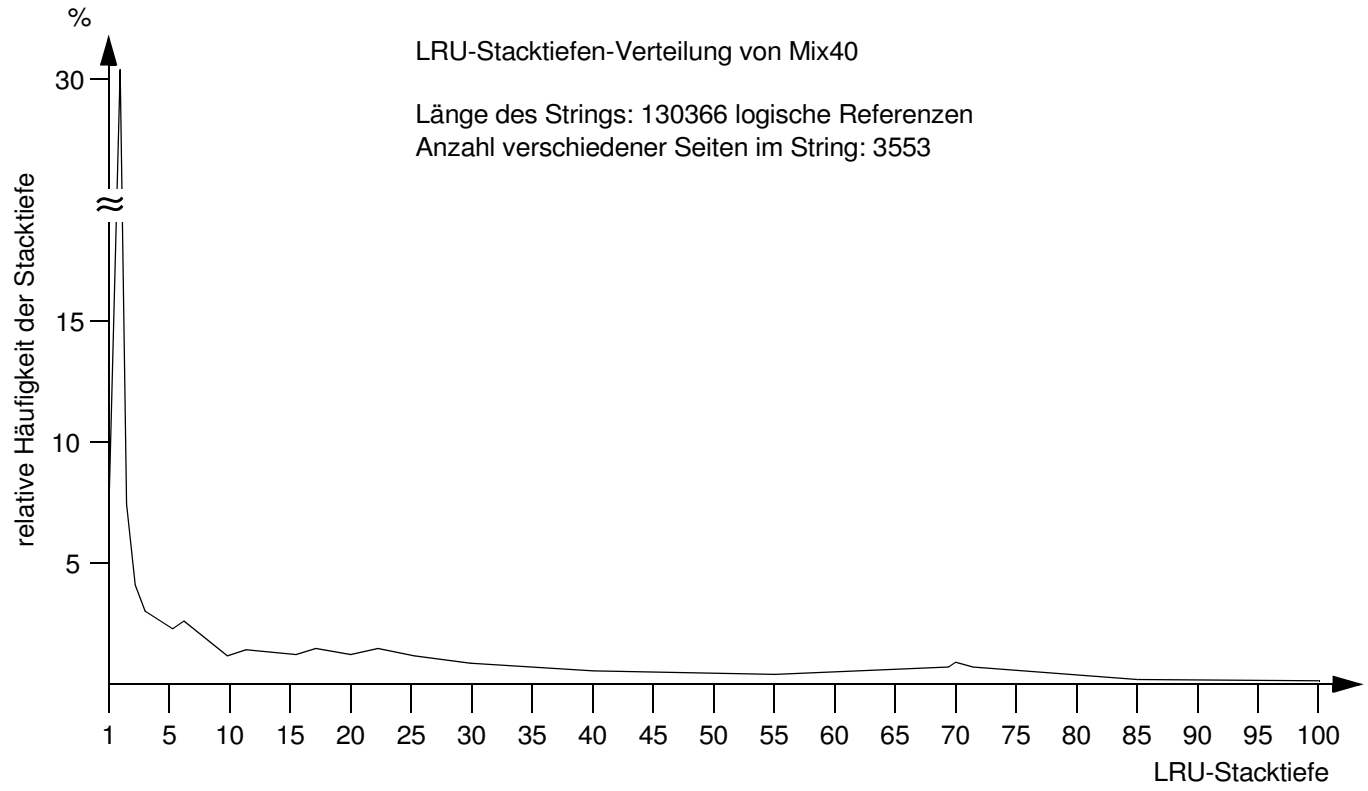
LRU-Stack:

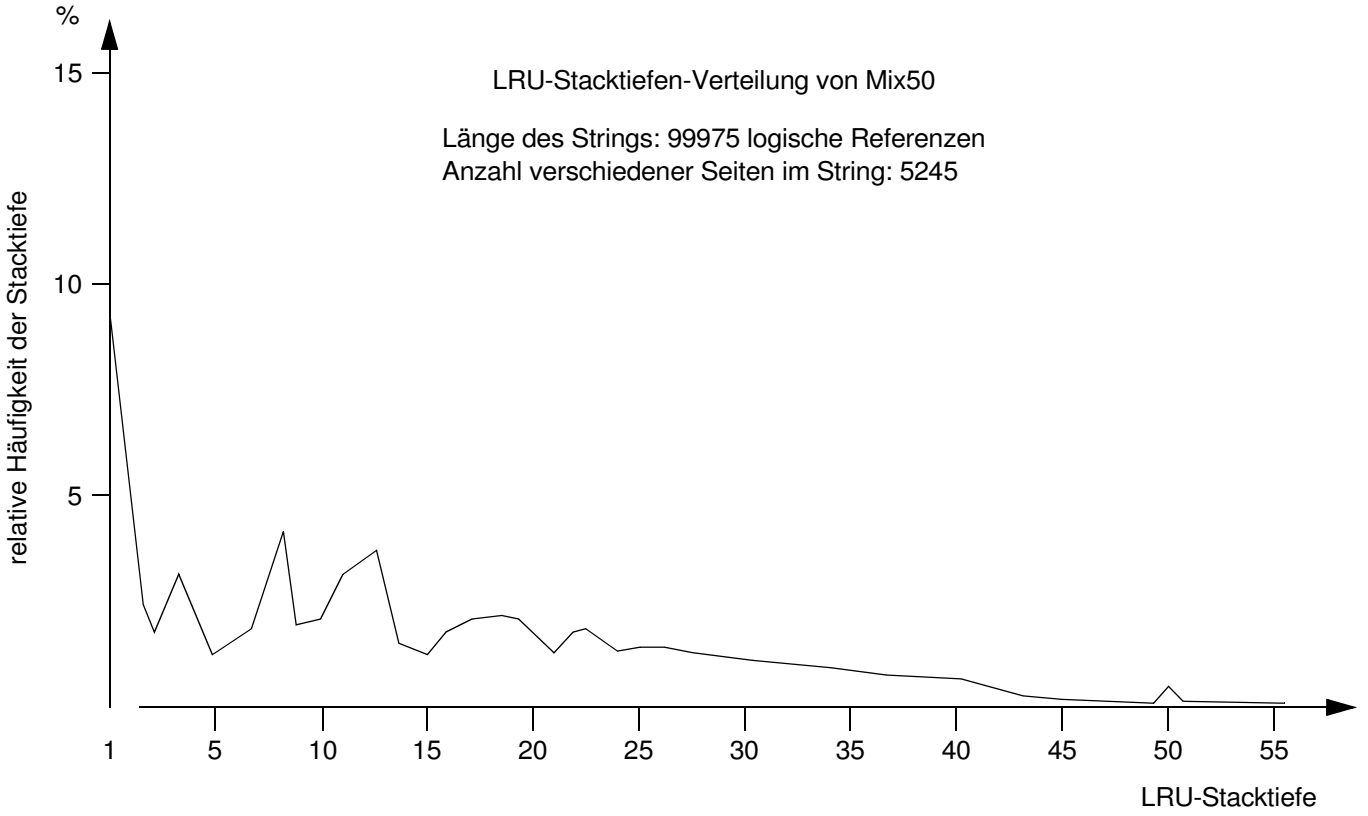
1	**A**
2	**B**
3	**C**
4	**D**
5	**E**

Stacktiefen-Verteilung

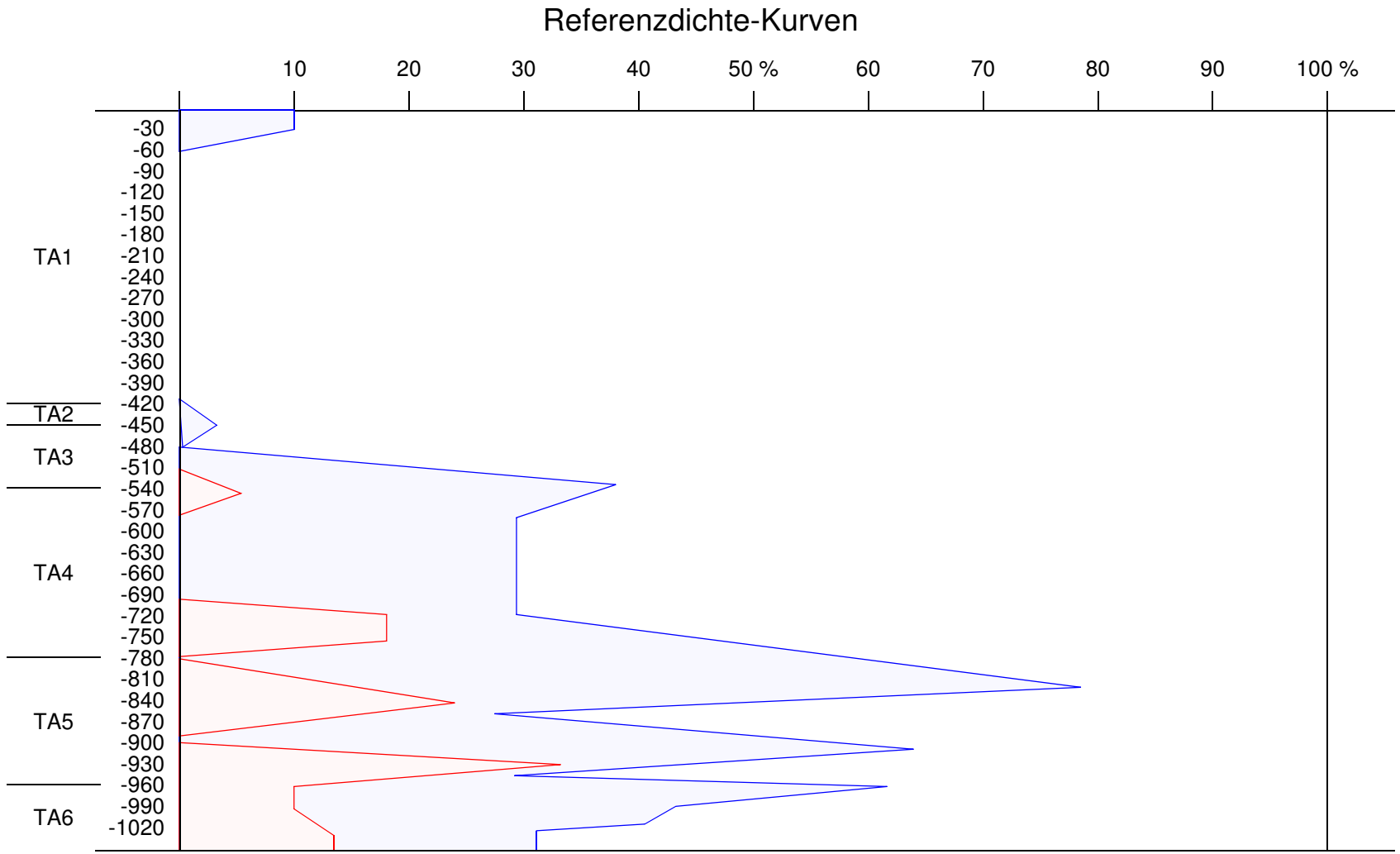


Reale LRU-Stacktiefen-Verteilungen[2]





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

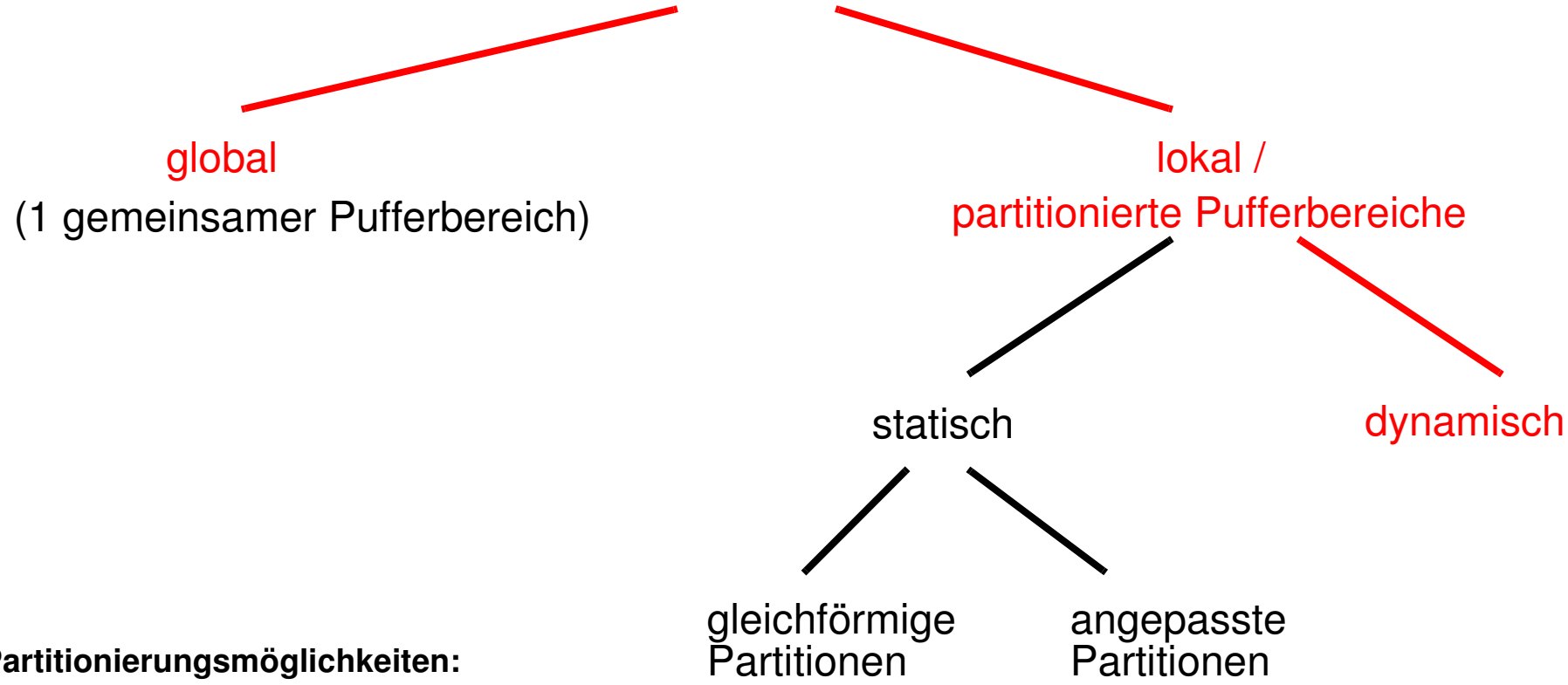


Relative Häufigkeit der Seitentypen im Beispiel

- = Daten und Indexstrukturen: 93,8 %
- = Adressumsetztabellen: 6,1 %
- = Freispeicher-Verwaltung: 0,1 %

Referenzdichte-Kurven

Speicherzuteilung im DB-Puffer



Partitionierungsmöglichkeiten:

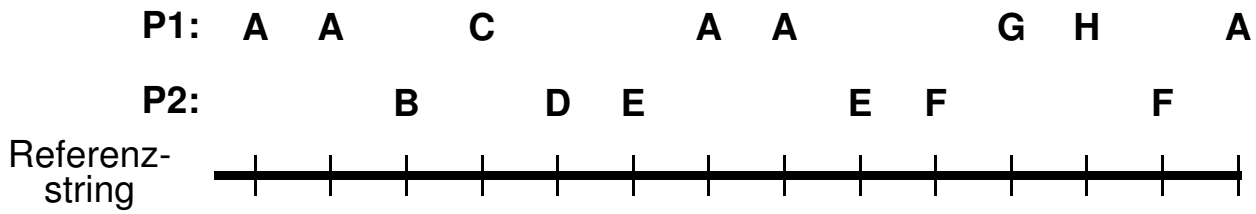
- eigener Pufferbereich pro Transaktion
- TA-Typ-bezogene Pufferbereiche
- Seitentyp-bezogene Pufferbereiche
- DB-(Partitions)spezifische Pufferbereiche

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 → Thrashing-Gefahr**



Suche im DB-Puffer

- **Sequentielles Durchsuchen der Pufferrahmen**
  - sehr hoher Suchaufwand
  - Gefahr vieler Paging-Fehler bei virtuellen Speichern

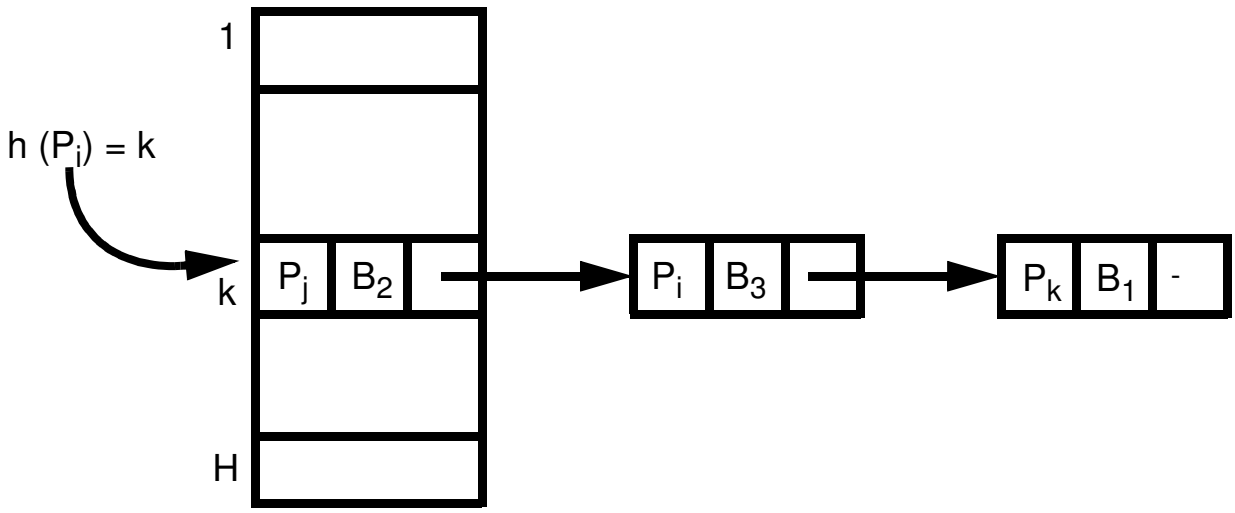
- **Nutzung von Hilfsstrukturen**
  (Eintrag pro Pufferrahmen)

  1. **unsortierte oder sortierte Tabelle**

  2. **Tabelle mit verketteten Einträgen**
     - geringere Änderungskosten
     - Anordnung in LRU-Reihenfolge möglich

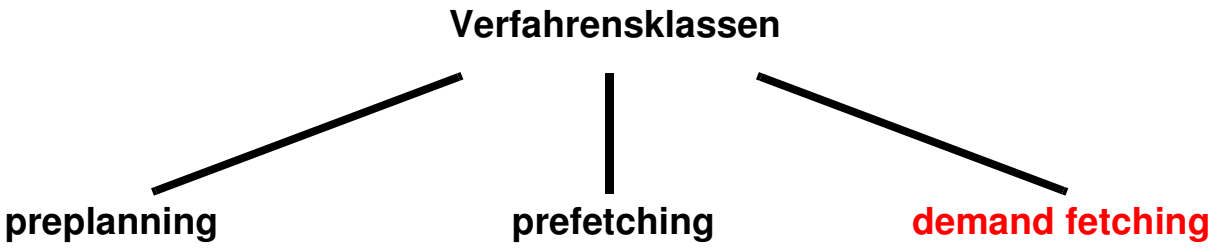
  3. **Suchbäume (z. B. AVL-, m-Weg-Bäume)**

  4. **Hash-Tabelle mit Überlaufketten**
     - beste Lösung



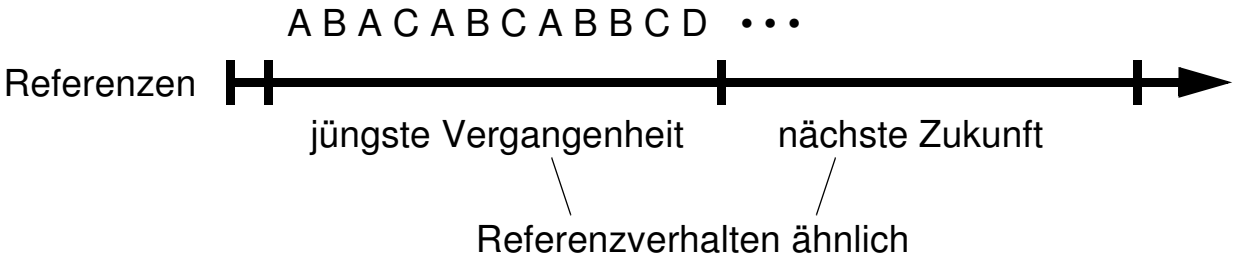
Seitenersetzungsverfahren

- **Klassifikation**



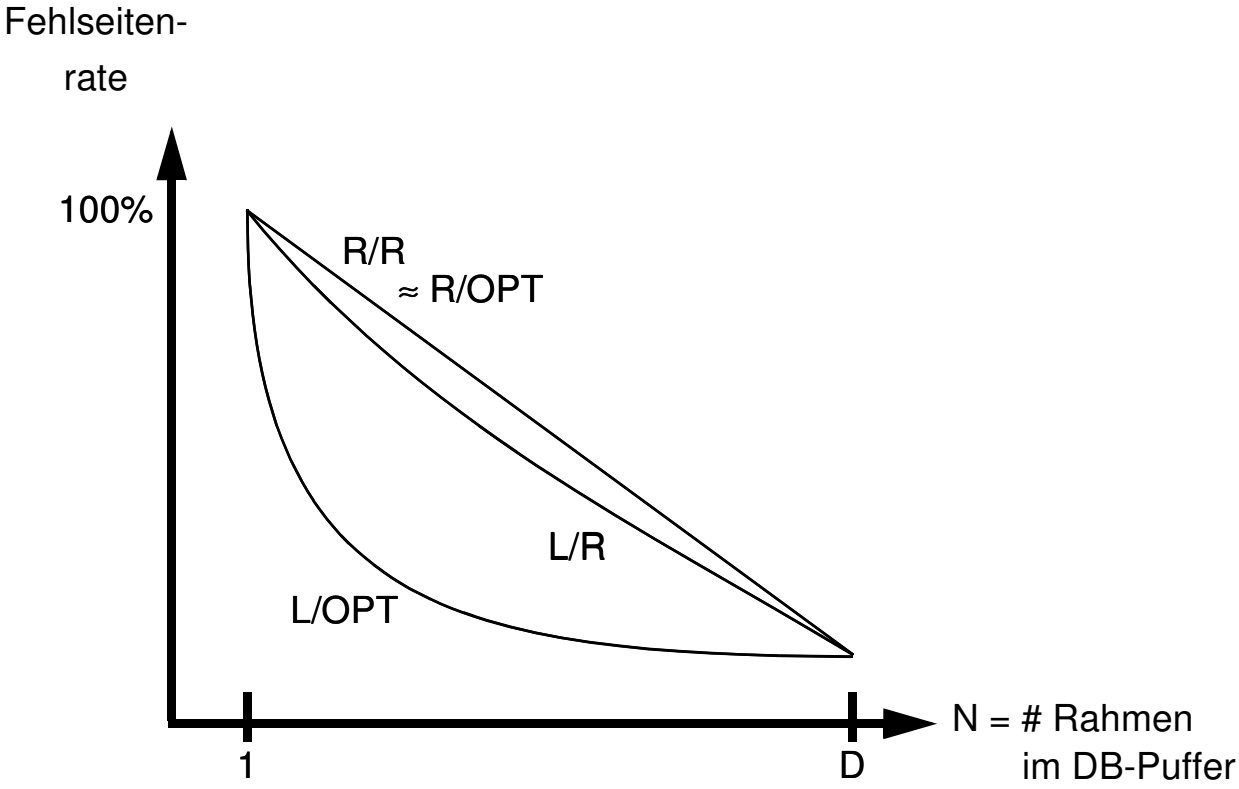
preplanning	prefetching	demand fetching
Programmanalyse, Vorabuntersuchung des Datenbedarfs	physische Daten-strukturierung, Clusterbildung, Verarbeitungswissen	keine Vorausaktionen
➥ große Fehlrate, ungenaue Ober-mengen	➥ datenmodellbezogen (hierarchisch), spekulative Entschei-dungen	➥ Lokalitätserhaltung im DB-Puffer

- **Grundannahme bei Ersetzungsverfahren**:



Referenzverhalten und Ersetzungsverfahren

- **Referenzverhalten in DBS**
  - **typischerweise** hohe Lokalität: Optimierung durch Ersetzungsverfahren
  - manchmal Sequentialität oder zufällige Arbeitslast (RANDOM-Referenzen)
- **Prinzipielle Zusammenhänge**, welche 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. Allg. 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 DB-Schreibvorgänge 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	-	-	X	---
FIFO	X	-	-	---
LRU				
CLOCK				
GCLOCK				
LRD (V1)				
LRD (V2)				
LRU-K				

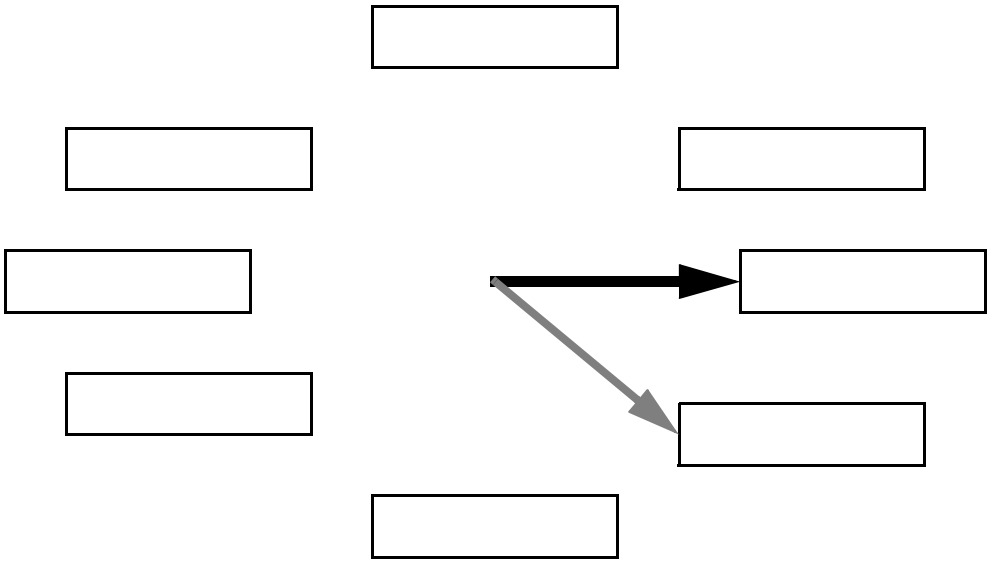
Least Frequently Used und First-In First-Out

- **Algorithmus LFU**
  - Referenzzähler pro Seite wird bei jeder Seitenreferenz inkrementiert
  - Ersetzung der Seite mit der geringsten Referenzhäufigkeit

RZ	
2	
4	
1	
3	
3	
6	
1	
3	

➥ **Alter einer Seite wird nicht berücksichtigt!**

- **Algorithmus FIFO**
  - Die älteste Seite im DB-Puffer wird ersetzt
  - Referenzen während des Pufferaufenthaltes werden nicht berücksichtigt

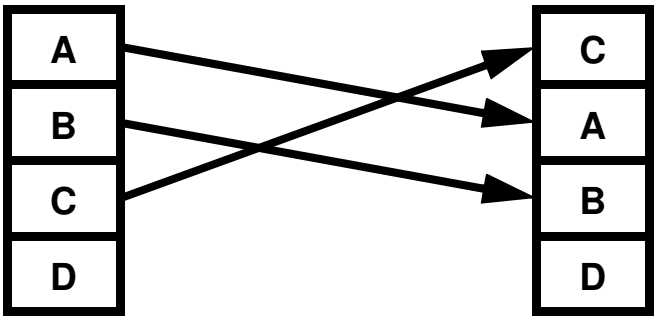


➥ **Nur für strikt sequentielles Referenzierungsverhalten geeignet**

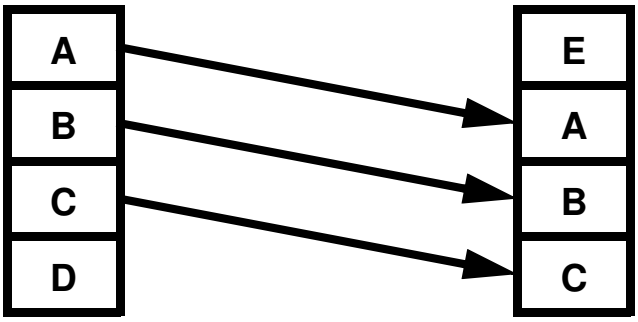
Least Recently Used (LRU)

- **Beispiel (Puffergröße 4):**

  1. Referenz der Seite C



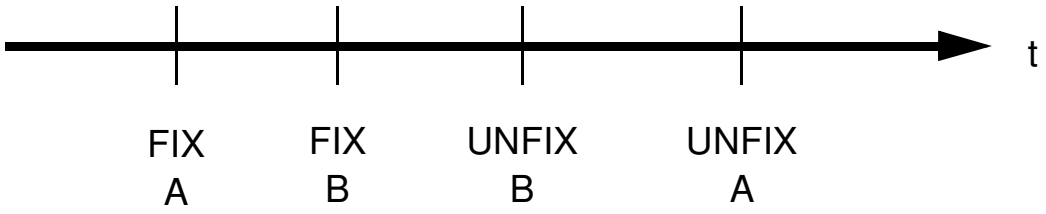
  2. Referenz der Seite E



- **Unterscheidung zwischen**

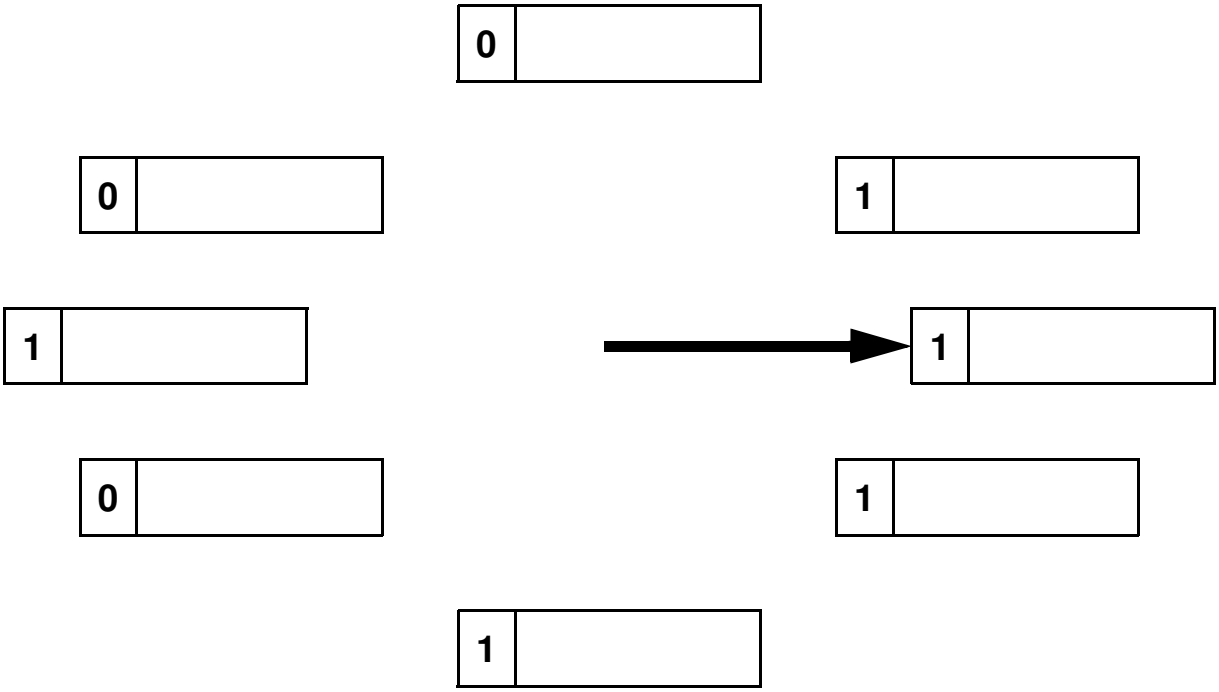
  Least Recently Referenced **und**

  Least Recently Unfixed



CLOCK (Second Chance)

- **Algorithmus**
  - Erweiterung von FIFO
  - Referenzbit pro Seite, das bei Zugriff gesetzt wird
  - Ersetzung erfolgt nur bei zurückgesetztem Bit,
    sonst erfolgt Zurücksetzen des Bits



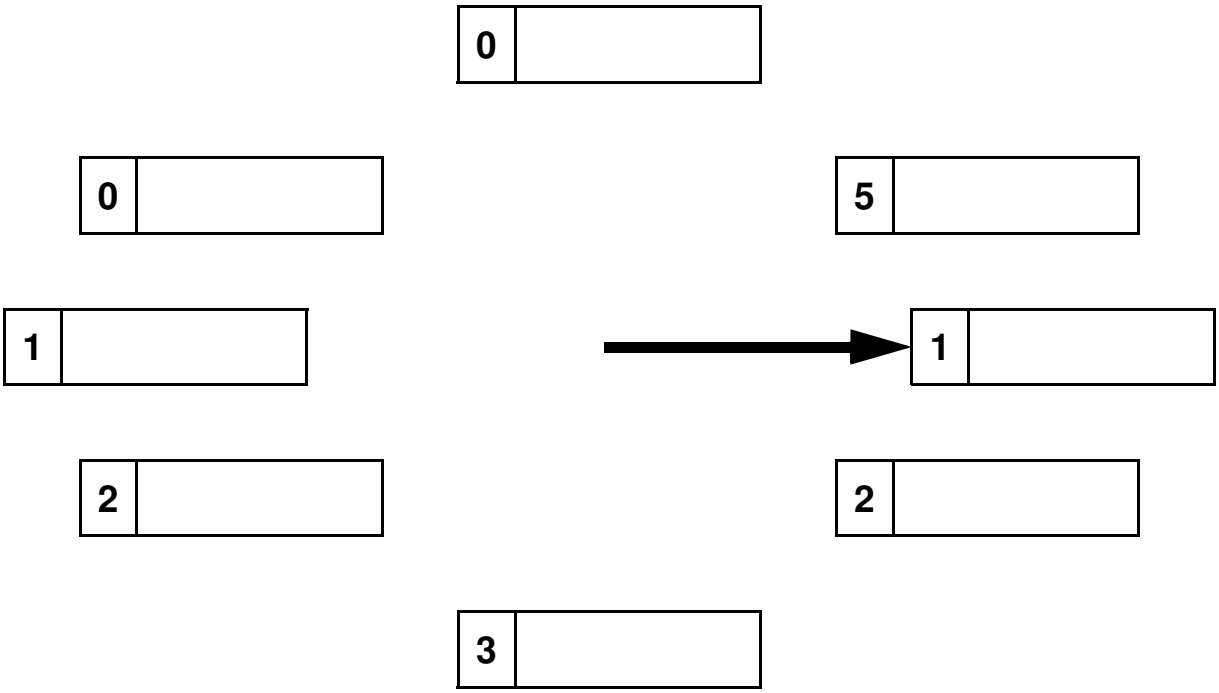
➥ **annähernde Berücksichtigung des letzten Referenzierungszeitpunkts**

Seitenersetzungsverfahren – Beispiel

Seitenreferenzfolge	2	3	2	1	5	2	4	5	3	2	5	2
OPT	2	2 3	2 3	2 3 1	2 3 5 F	2 3 5	4 3 5 F	4 3 5	4 3 5	2 3 5 F	2 3 5	2 3 5
LRU	2	2 3	2 3	2 3 1	2 5 1 F	2 5 1	2 5 4 F	2 5 4	3 5 4 F	3 5 2 F	3 5 2	3 5 2
FIFO	2	2 3	2 3	2 3 1	5 3 1 F	5 2 1 F	5 2 4 F	5 2 4	3 2 4 F	3 2 4	3 5 4 F	3 5 2 F
CLOCK →	2*	2* 3*	2* 3*	2* 3* 1*	5* 3 1 F	5* 2* 1 F	5* 2* 4* F	5* 2* 4*	3* 2 4 F	3* 2* 4	3* 2 5* F	3* 2* 5*

GCLOCK (Generalized CLOCK)

- **Algorithmus**
  - 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
  - Zähler-Inkrementierung bei erneuter Referenz
  - Vergabe von seitentyp- oder seitenspezifischen Gewichten

Least Reference Density (LRD)

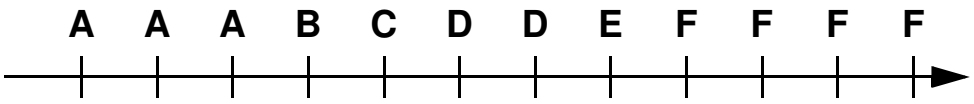
- **Algorithmus**
  - Wenn eine Seite ersetzt werden muss, wird die Referenzdichte aller Seiten im DB-Puffer bestimmt
  - Referenzdichte = Referenzhäufigkeit in einem bestimmten Referenzintervall
  - Ersetzungskandidat ist Seite mit geringster Referenzdichte

- **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			

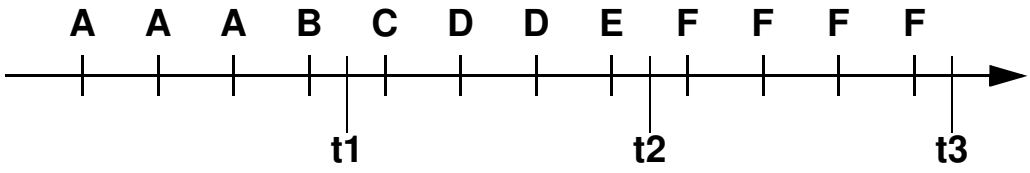
Least Reference Density (2)

- **Variante 2: konstante Intervallgröße**
  - Künstliches Altern von Seiten: Ältere Referenzen werden bei der Bestimmung der Referenzdichte geringer bewertet
  - **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

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



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

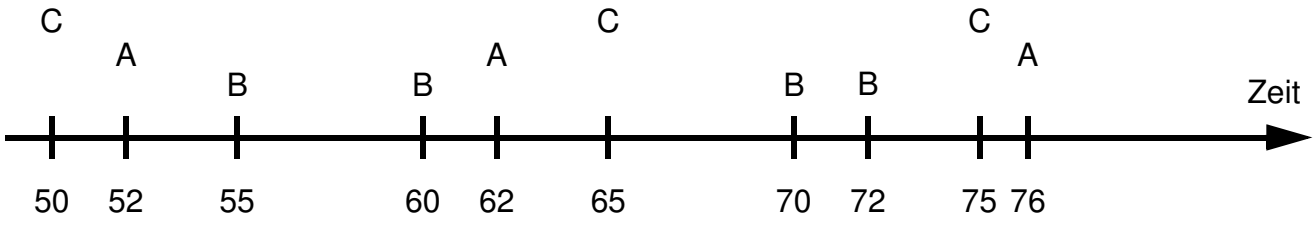
LRU-K

- **Aufzeichnung der K letzten Referenzzeitpunkte** (pro Seite im DB-Puffer)
  - Aufwendigere Aufzeichnung, Methode benötigt kein explizites „Altern“
  - 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=1, 2, 3)**



- **Bei Ersetzung** werden die $b_t(P_i, K)$ der Pufferseiten benötigt!
  - Sonderbehandlung für Seiten mit $b_t(P, K) = \infty$
  - Wie hängt LRU-K mit LRD zusammen? Approximation der Referenzdichte?

- **LRU-2 (d.h. K=2) stellt i. Allg. beste Lösung dar[3]**
  - ähnlich gute Ergebnisse wie für K > 2, jedoch einfachere Realisierung
  - schnellere Reaktion auf Referenzschwankungen als bei größeren K

3. 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

Simulation von Seitenersetzungsverfahren

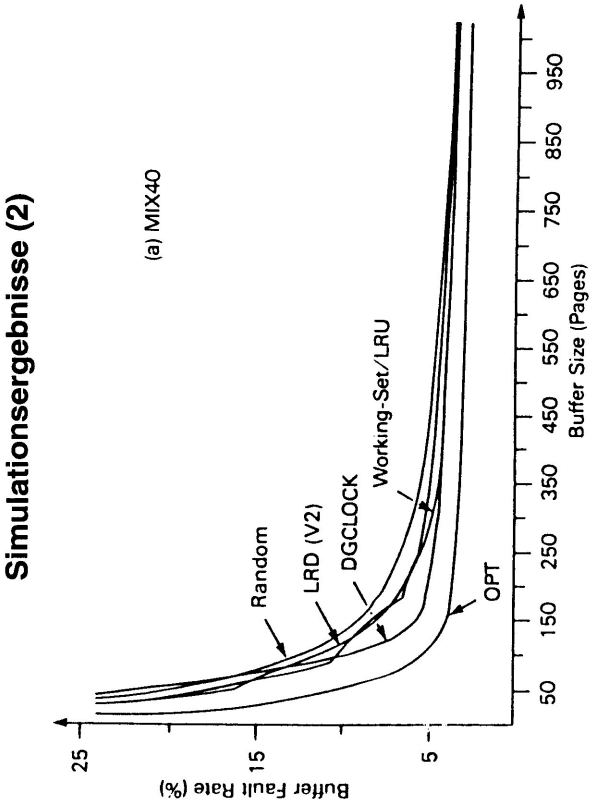
- Charakteristika von DB, Transaktionslast und logischen Seitenreferenzstrings

		MIX40	MIX50
Total number of pages (school-DB)		30,000	
Number of different pages in the string		3,553	5,245
Number of logical references		130,366	99,975
Number of page modifications		9,378	2,865
Number of pages being fixed*	maximum	11	10
	average	4.61	6.26
Percentage of references with FIX-duration	= 1	71%	41%
	2-10	22%	41%
	>10	7%	18%
FIX-duration (in logical references)	maximum	1,786	-
	average	4.62	6.26
Percentage of pages of a given type	FPA	0.1%	
	DBTT	6.1%	
	USER	93.8%	
Percentage of references to page-types	FPA	0.9%	0.1%
	DBTT	9.4%	21.7%
	USER	89.7%	78.2%
Percentage of references to most frequently referenced pages (hot spot pages)	1.	12.6%	3.3%
	2.	9.8%	0.5%
	3.	4.8%	0.4%
Relative frequency distribution of references to the other pages	>0.9%	5	1
	0.9%-0.1%	195	293
	0.1%-0.03%	1,606	2,741
	< 0.03%	1,747	2,208
Number of references to shared pages (concurrently fixed)*		1,175	359
cold start buffer fault rate in %		2.72	5.24
number of executed transactions	total	262	39
	parallel:max.	8	8
	avg.	6.31	6.29

*measured with a buffer size of 128 pages

Fig. 12: Characteristics of the DB, transaction load and logical reference strings.

Simulationsergebnisse



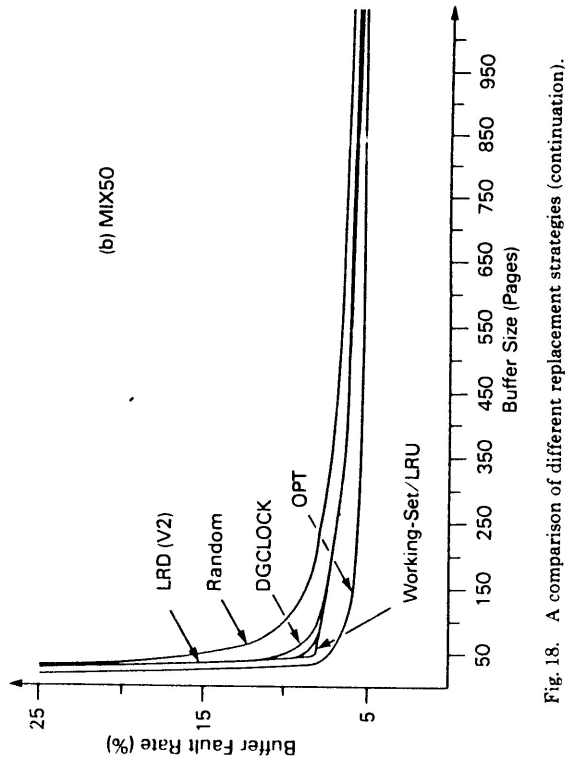
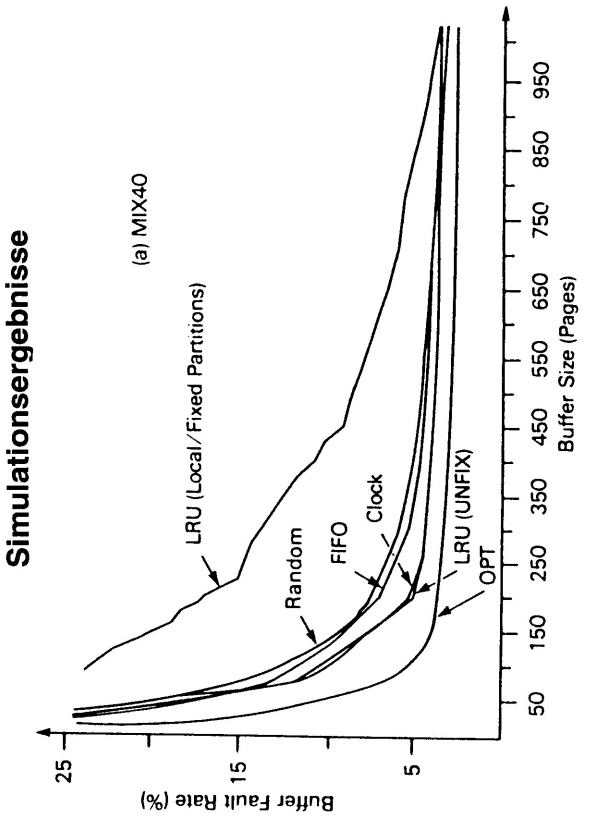


Fig. 18. A comparison of different replacement strategies (continuation).



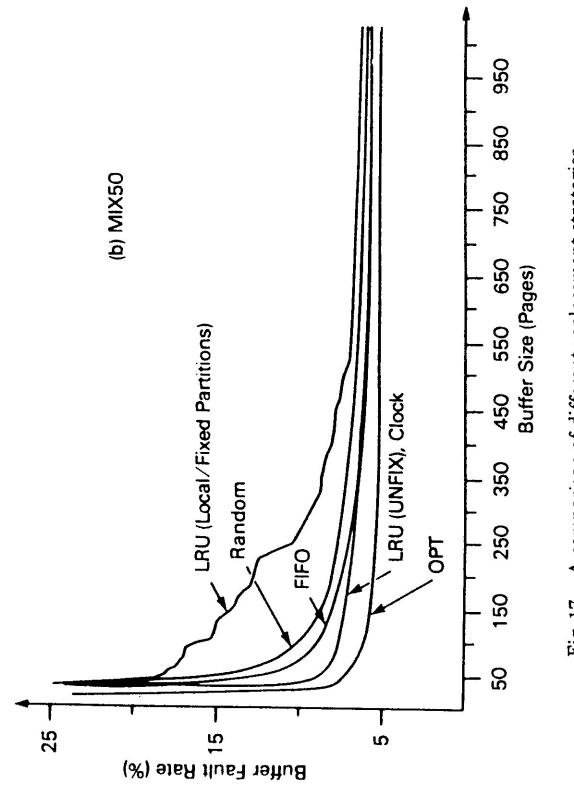


Fig. 17. A comparison of different replacement strategies.

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)

    ➥ **internes Thrashing**

  - T1: zyklisches Referenzieren einer Seitenmenge (#Seiten < #Rahmen)
    Interferenz durch T_i bei schnellerer Anforderung (*stealing*)

    ➥ **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

Ersetzungverfahren – Einbezug von Kontextwissen (2)

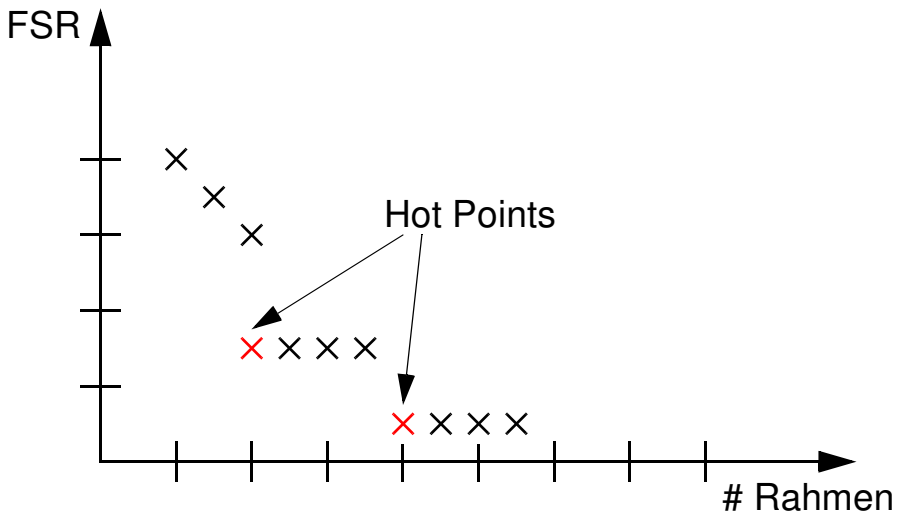
- **Ausnutzung von Kontextwissen bei mengenorientierten Anforderungen**

  ➥ Verbesserung in relationalen DBS möglich

- **Zugriffspläne durch Anfrage-Optimierer**
  - 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

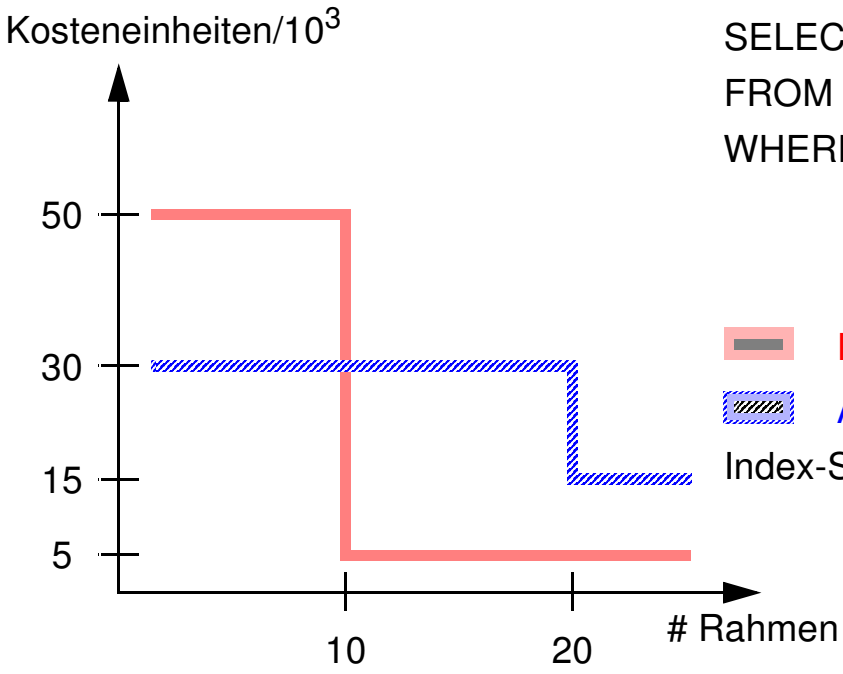
- **Hot Set: Menge der Seiten im Referenzzyklus**

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



Hot Set Model

- ***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
- Anfrage-Optimierer 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

Prioritätsgesteuerte Seitenersetzung

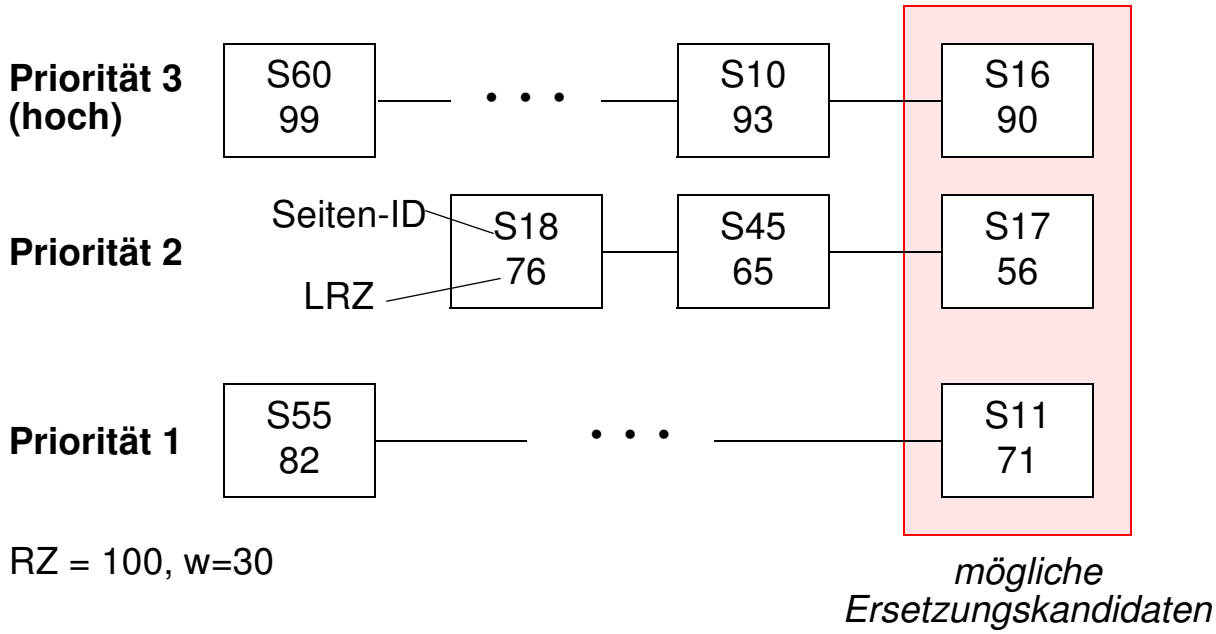
Bevorzugung bestimmter Transaktionstypen/DB-Partitionen vielfach wünschenswert (z. B. um Benachteiligungen durch sehr lange TA oder sequentielle Zugriffe zu vermeiden)

➥ Berücksichtigung von Prioritäten bei der DB-Pufferverwaltung

- **Verfahren PRIORITY LRU[4]:**
  - 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

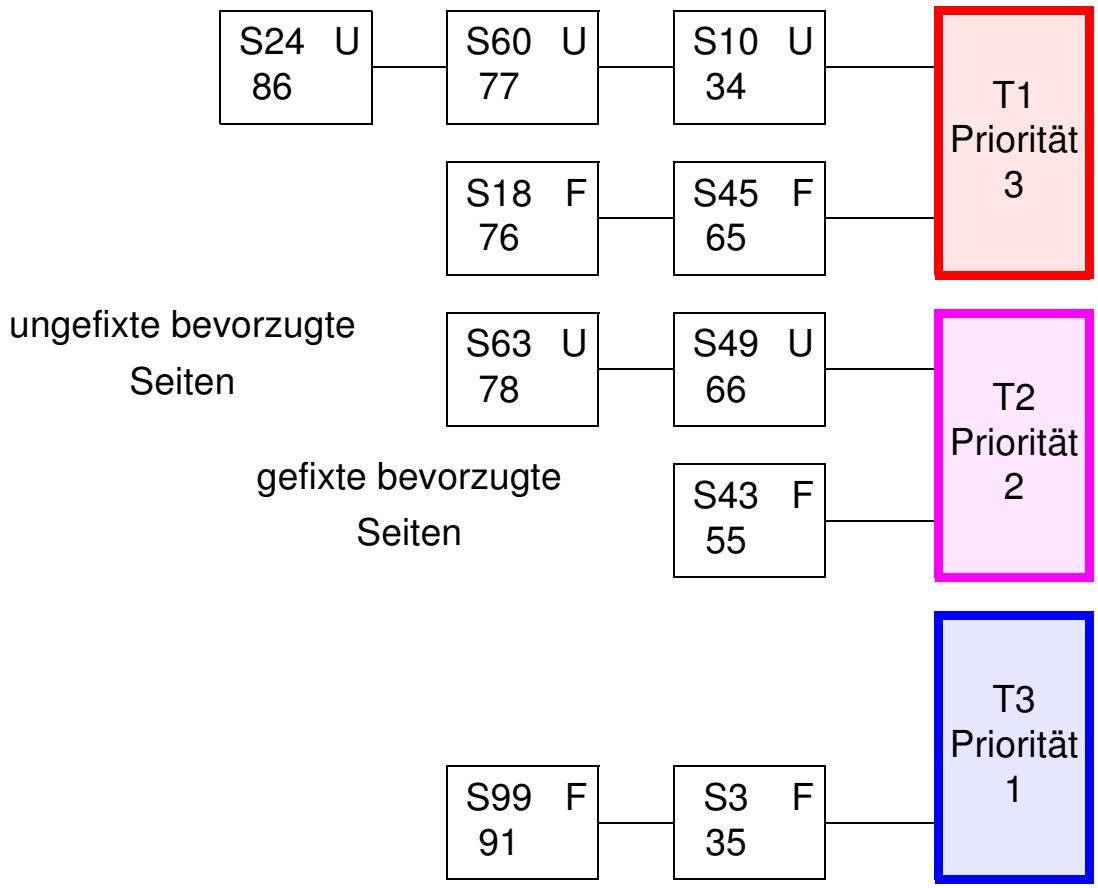
➥ Kompromiss zwischen Prioritäts- und absolutem LRU-Kriterium möglich



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

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



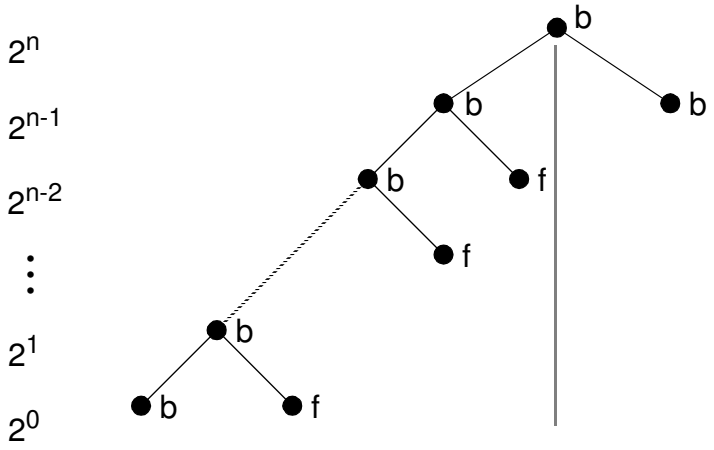
DB-Pufferverwaltung
bei Seiten variabler Größe

- **Seitengröße**
  - keine beliebigen Seiten (Fragmentierung, Abbildung auf Externspeicher)
  - Seitenlänge (SL) als Vielfaches eines Einheitsrasters (Transporteinheit, Rahmengröße im DB-Puffer)
  - Beispiel: 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

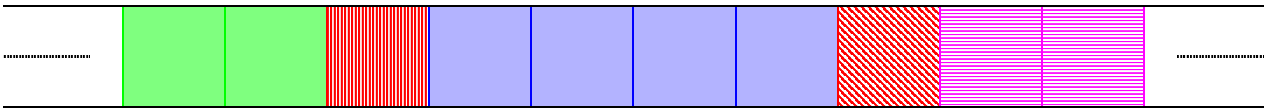
- **Zustände von Seiten/Rahmen**
  - frei: keine Seite vorhanden
  - unfixed: Seite ist ersetzbar/verschiebbar
  - fixed: Seite muss Pufferadresse behalten
- **Vermeidung von Seitenersetzungen**
  - Umlagern von Seiten mit Unfix-Vermerk und „hoher“ Wiederbenutzungswahrscheinlichkeit
- **Suche nach „bestem“ 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**

  ➥ sehr komplexe Entscheidungssituation

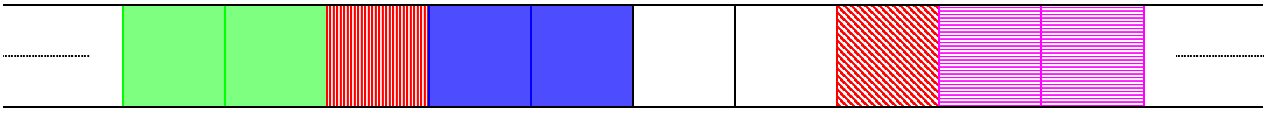
- **Alternative: Paritionierung**
  - Konfigurierung von mehreren DB-Puffern jeweils für Seiten gleicher Größe
  - Einsatz anwendungs- und typabhängiger Ersetzungsverfahren
  - DB2 erlaubt bis zu 80 Puffer

DB-Pufferverwaltung — Seiten variabler Länge

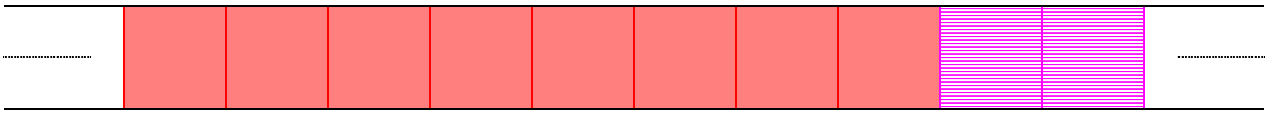
- **Belegungsbeispiel bei Seiten variabler Größe und Rahmen fester Größe**



- **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

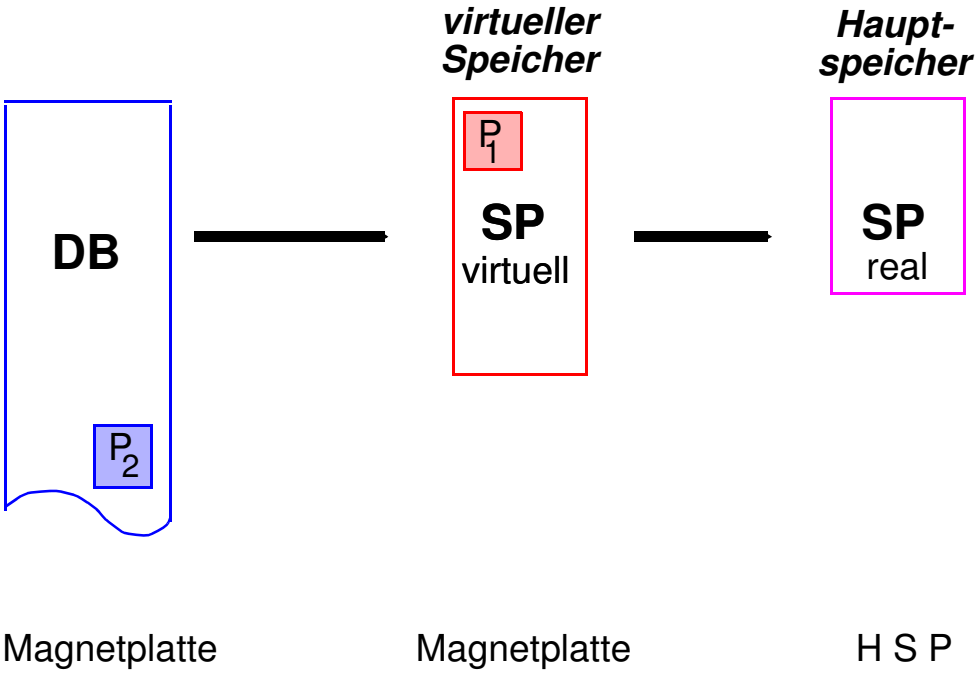
- **Vorschlag für einen Algorithmus: VAR-PAGE-LRU[5]**
  - Was heißt hier LRU?

- **Neue Aufgabenstellung:**
  Caching und spekulatives Prefetching von Dokumenten in Speicherhierarchien (Tertiärspeicher)[6]

---

5. 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
6. A. Kraiss, G. Weikum: *Integrated Document Caching and Prefetching in Storage Hierarchies Based on Markov-Chain Predictions*, in: VLDB Journal 7:3, pp. 141-162, 1998.

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
  - Erkennen von Scan-basierter Verarbeitung
- **Ohne Lokalität ist jede Optimierung** der Seitenersetzung sinnlos (~ RANDOM)
- **Suche im Puffer durch Hash-Verfahren**
- **Speicherzuteilung:**
  - global $\Rightarrow$ alle Pufferrahmen für alle Transaktionen (Einfachheit, Stabilität, ...)
  - lokal $\Rightarrow$ Sonderbehandlung bestimmter TAs/Anfragen/ DB-Bereiche
- **Behandlung geänderter Seiten:**
  NOFORCE, asynchrones Ausschreiben
- **Seitenersetzungsverfahren**
  - Prefetching und Pipelining mit Wechselpuffer
  - „zu genaue“ Verfahren sind schwierig einzustellen ($\Rightarrow$ 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 Anfrage-Optimierers
  - Hot Set Model
  - Einsatz vieler Puffer zur Separierung von Lasten verschiedenen Typs, optimiert für spezifische Datentypen
- **Double-Paging sollte vermieden werden**