

9. DBS-Pufferverwaltung

- ❑ Allgemeine Charakteristika
- ❑ Eigenschaften von DB-Referenzstrings
 - Seitenreferenzstrings
 - Lokalität, Sequentialität
 - LRU-Stacktiefenverteilung
 - Referenzdichte-Kurven
- ❑ Suche im Puffer
- ❑ Speicherzuteilung im Puffer
- ❑ Seitenersetzungsverfahren
- ❑ Ersetzungsverfahren - Einbezug von Kontextwissen
- ❑ Pufferverwaltung bei Seiten variabler Größe
 - zusätzliche Anforderungen
 - VAR-PAGE-LRU
- ❑ Seitenersetzung bei virtuellem Speicher

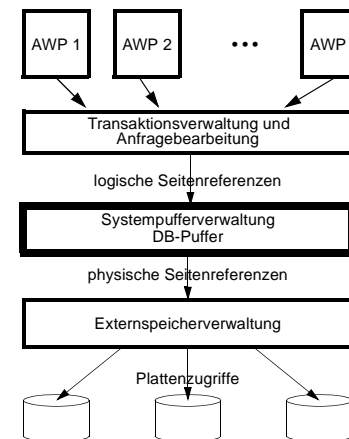
326.

Eigenschaften eines DB-Puffers

- ❑ Puffer besteht aus **Frames**
 - Annahme: ein Frame kann genau eine Seite aufnehmen.
 - Puffer liegt im tatsächlichen HSP und nicht im virtuellen HSP.
- ❑ Ein Puffer kann für
 - eine Datei
 - eine Seitengröße
 - einen Dateityp
 verwendet werden.
 - Annahme: **DBS besitzt genau einen Puffer.**
- ❑ Puffer wird gemeinsam von verschiedenen AWP's benutzt.
- ❑ Puffer liest direkt von der Platte (ohne daß der Block nochmals in einem Puffer des Betriebssystems gespeichert wird.)
- ❑ Zeitpunkt des Zurückschreibens einer modifizierten Seite wird allein durch den Pufferorganisator und nicht durch das AWP bestimmt.

328.

Funktionsweise der Pufferverwaltung in DBS



Schnittstelle:

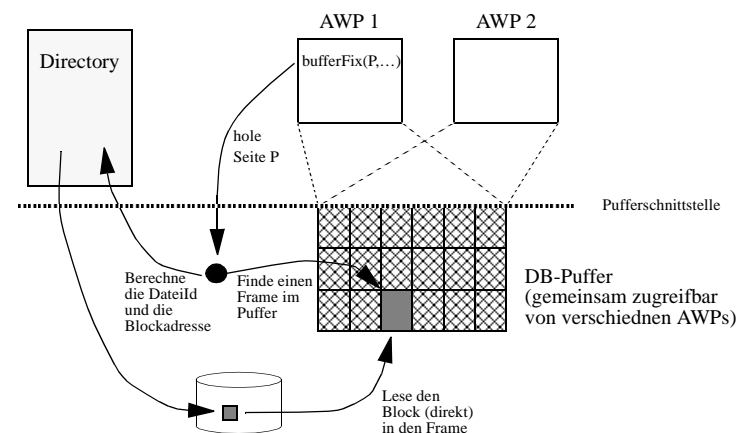
- ❑ Bereitstellen einer DB-Seite im DB-Puffer (zur exklusiven oder gemeinsamen Benutzung).
- ❑ Bereitstellen einer neuen Seite.
- ❑ Freigeben einer Seite.

intern verwendete Funktionen:

- ❑ effiziente Suche im Puffer
- ❑ Suche nach freien **Frames**
- ❑ Bestimmen einer Seite, die aus dem Puffer entfernt wird.
- ❑ Schreiben modifizierter Seiten
- ❑ Umsetzen logischer Seitenreferenzen in physische Referenzen (FileID, Block#).

327.

Ablauf einer Anforderung an den DB-Puffer

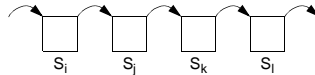


329.

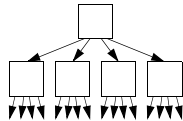
9.1 Seitenreferenzen

Typische Referenzmuster in DBS:

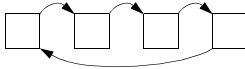
1. Sequentielle Suche (z. B. Durchsuchen ganzer Satztypen (Relationen))



2. Hierarchische Pfade (z. B. beim Verarbeiten von Selektionsanfragen mittels Indexstrukturen)



3. Zyklische Pfade (z. B. bei der Join-Verarbeitung)



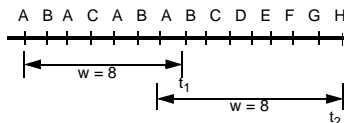
330.

Lokalität

- hohe Wiederbenutzungswahrscheinlichkeit für gerade referenzierte Seiten
- grundlegende Voraussetzung für
 - effektive Pufferverwaltung (Seitenersetzung)
 - Einsatz von Speicherhierarchien
- Wie kann man Lokalität messen?

Working-Set-Modell

Referenzstring



Working Set Size W:

$$W(t_1, w=8) = 3$$

$$W(t_2, w=8) = 8$$

$A(t, w) = \frac{W(t, w)}{w}$ aktuelle Lokalität	$L(w) = \frac{\sum_{t=1}^n A(t, w)}{n}$ durchschnittliche Lokalität (n = Länge des Referenzstrings)
--	--

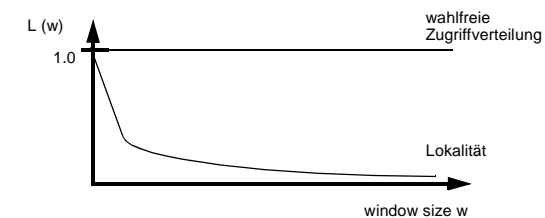
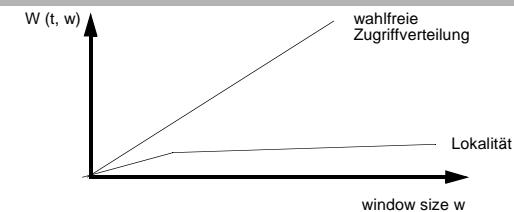
332.

Seitenreferenzstrings

- jede Datenanforderung besteht aus einer *logischen Seitenreferenz*
- Aufgabe der Pufferverwaltung:
Minimierung der *physischen Seitenreferenzen*
- Referenzstring $R = \langle r_1, r_2, \dots, r_i, \dots, r_n \rangle$ 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 ?

331.

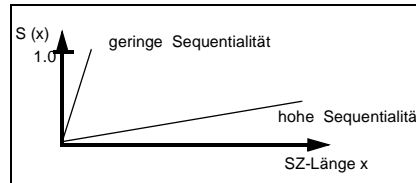
Typisches Verhalten



333.

Sequentialität

- ❑ aufeinanderfolgende Zugriffe auf 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
 $S(x) = \Pr(SZ\text{-Länge} \leq x)$
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.

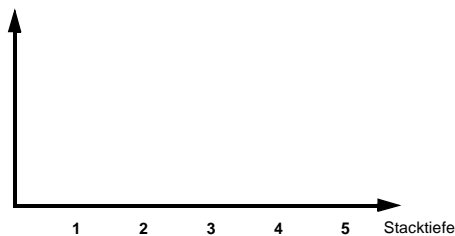


334.

Beispiel: Stacktiefenverteilung

- ❑ Referenzstring: A B A C A B C D E A

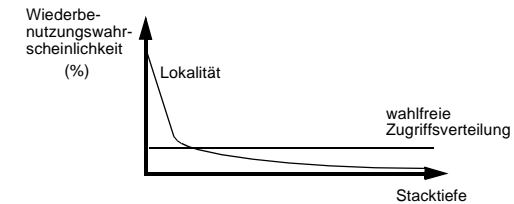
1	A								
2	B								
3	C								
4	D								
5	E								



336.

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 (*Wiederbenutzungshäufigkeit*)
 - Referenz einer Seite führt zur Zählererhöhung für die jeweilige Stackposition

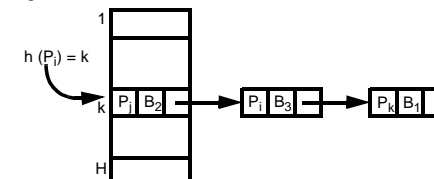


- ❑ Für LRU-Seitenersetzung kann aus der Stacktiefenverteilung für eine bestimmte Puffergröße unmittelbar die Trefferrate (bzw. Fehlseitenrate) bestimmt werden.

335.

9.2 Suche im Puffer

- ❑ direkt (sequentielles Durchsuchen der Pufferrahmen)
 - sehr hoher Suchaufwand
- ❑ indirekt / Nutzung von Hilfsstrukturen: Hash-Tabelle mit Überlaufketten



337.

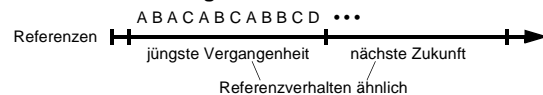
9.3 Seitenersetzungsverfahren

optimale Strategie:

Ersetze immer die Seite, die am längsten nicht gebraucht wird.

drei Näherungsmethoden:

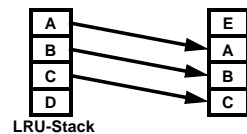
- ☐ preplaning
 - durch untersuchen der AWP ergibt sich der zukünftige Referenzstring
- ☐ prepaging
 - basiert auf der Nutzung semantischen Wissens über physische Datenstrukturen und der physischen Anfragebearbeitung
- ☐ demand paging
 - kein Blick in die Zukunft
 - Lokalitätserhaltung im Puffer
- ☐ **Grundannahme bei Ersetzungsverfahren:**



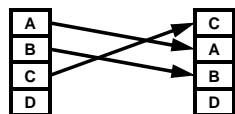
338.

9.3.1 Least-Recently-Used (LRU)

- ☐ die Seite wird ersetzt, die unter allen Seiten im Puffer am längsten nicht mehr referenziert wurde.
 - ☐ einfache und effiziente Implementierung
- Beispiel (Puffergröße 4):
- Referenz der Seite E



- Referenz der Seite C

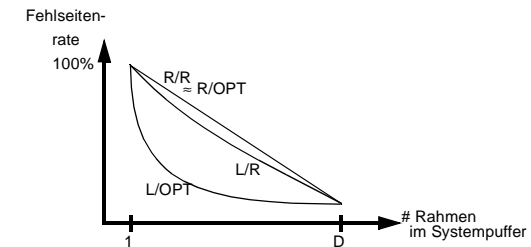


- ☐ Unterscheidung zwischen Least-Recently-Referenced und Least-Recently-Unfixed

340.

Referenzverhalten und Ersetzungsstrategien

- ☐ **typischerweise** hohe Lokalität: Optimierung durch Ersetzungsverfahren
- ☐ **manchmal** Sequentialität oder zufällige Arbeitslast (RANDOM-Referenzen)

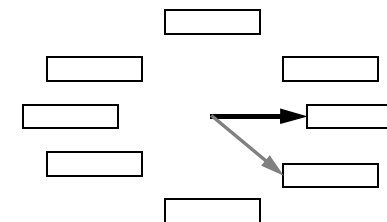


- ☐ D = Datenbankgröße (in der Anzahl der Seiten)
- | | | | |
|----------------------------|----------------|-------------------|-------------------|
| Referenzen: <u>R</u> ANDOM | <u>R</u> ANDOM | <u>L</u> okalität | <u>L</u> okalität |
| Ersetzung: <u>R</u> ANDOM | <u>O</u> PT | <u>R</u> ANDOM | <u>O</u> PT |

339.

9.3.2 FIFO (First-In First-Out)

- ☐ die älteste Seite im Puffer wird ersetzt



- ☐ Referenzierungsverhalten während Pufferaufenthaltes wird nicht berücksichtigt
- ☐ nur für strikt sequentielles Referenzverhalten geeignet

341.

9.3.3 Least-Frequently-Used

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

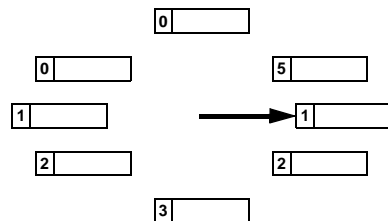
RZ	
2	
4	
1	
3	
3	
6	
1	
3	

- ❑ Alter einer Seite wird nicht berücksichtigt

342.

9.3.5 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)

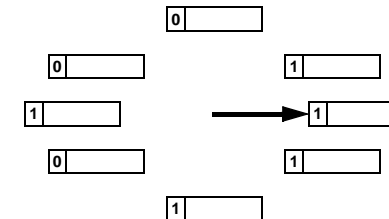


- ❑ Verfahrensparameter:
 - Initialwerte für Referenzzähler
 - Wahl des Dekrementes
 - Zählerinkrementierung bei erneuter Referenz
 - Vergabe von seitentyp- oder seitenspezifischen Gewichten

344.

9.3.4 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)

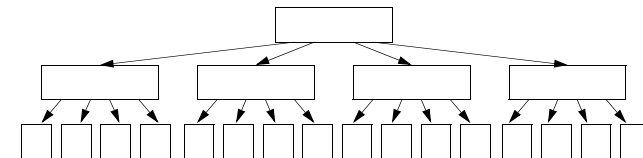


- ❑ annähernde Berücksichtigung des letzten Referenzierungszeitpunkts

343.

9.3.6 LRU-k

- ❑ Verallgemeinerung der LRU Pufferstrategie (LRU-1 = LRU)
- ❑ Problemfälle der ursprünglichen LRU-Strategie:
 1. hierarchische Zugriffsreferenzen:



Zugriff auf jeweils 1 Pfad und vier Blätter (typisches Profil einer DB-Anfrage)

Annahme: Wurzel ist stets im Puffer

Was passiert für Puffergröße 5 und was für Puffergröße 8?

2. Puffergröße: 5000,
 - “nahezu alle” Anfragen: 95% der referenzierten Seiten sind im Puffer
 - seltene (aber große) Anfragen: keine der Seiten liegt im Puffer
 - LRU tauscht komplett alle Seiten im Puffer aus.

345.

Idee

- ❑ Wenn eine der Seiten ersetzt werden muß, schätze für jede ersetzbare Seite im Puffer den Erwartungswert, wann die Seite als nächstes referenziert wird.
- ❑ Betrachte nicht nur die letzte Referenz einer Seite, sondern die letzten k . Genauer: Sei (r_1, \dots, r_n) ein Referenzstring und p eine Seite, dann ist

$$b_n(p, k) = x, \quad \text{falls } S_{n-x} = p \text{ und es gibt genau } k-1 \text{ Referenzen der Seite } p \text{ in dem Referenzstring } (r_{n-x+1}, \dots, r_n).$$

$$b_n(p, k) = \infty, \quad \text{wenn } p \text{ nicht mindestens } k\text{-mal in } (r_1, \dots, r_n) \text{ auftritt.}$$
- ❑ LRU- k entfernt die Seite aus dem Puffer mit höchstem Wert für b_n .

Probleme

1. korrelierte Referenzen: eine Referenz wird nur dann gezählt, wenn sie zeitlich einen Mindestabstand zur vorhergehenden Referenz hat.
2. Bestimmen der letzten k Referenzen von den Seiten, die nicht mehr im Puffer sind.
 Speichere die letzten k Referenzen zusätzlich für Seiten, die kurz zuvor bereits ausgelagert wurden.

346.

IDBS

Universität Marburg

- ❑ Folgender Algorithmus wird bei einer Referenz auf Seite p zum Zeitpunkt t ausgeführt:

```

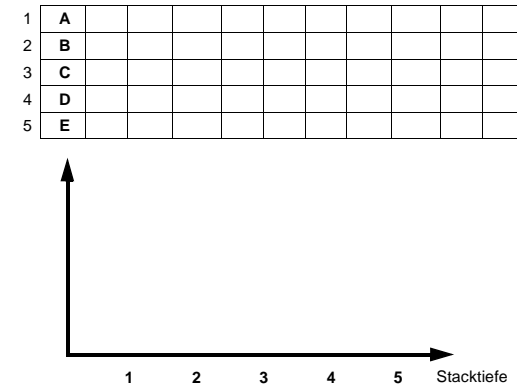
IF (p is already in the buffer) THEN
  IF (t - LAST(p) > KorRefZeit) THEN
    KorSkew = LAST(p) - HIST(p,1);
    FOR i := k DOWNTO 2 DO
      HIST(p,i) := HIST(p,i-1) + KorSkew;
    END;
    HIST(p,1) := t;
    LAST(p) := t;
  ELSE
    LAST(p) := t;
  END;
ELSE
  min := t;
  FOR (all pages q in the buffer) DO
    IF (t - LAST(q) > KorRefZeit) AND (HIST(q,k) < min) THEN
      victim := q;
      min := HIST(q,k);
    END;
  END;
  IF (victim is dirty) THEN
    write victim back into the database;
  END;
  Fetch p into the buffer frame previously held by victim;
  IF (HIST(p) does not exist) THEN
    ALLOCATE space;
    FOR i := 2 TO k DO HIST(p,i) := 0; END;
  ELSE
    FOR i := 2 TO k DO HIST(p,i) := HIST(p,i-1) END;
  END;
  HIST(p,1) := t;
  LAST(p) := t;
END;

```

348.

Beispiel: Stacktiefenverteilung (LRU-2)

- ❑ Referenzstring: A B A C A B C D E A



347.

IDBS

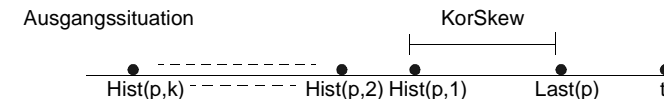
Universität Marburg

Erläuterungen zum Algorithmus

- ❑ Folgende Datenstrukturen werden im Algorithmus für eine Seite p benötigt:
 - Last(p): Zeitpunkt des letzten Zugriffs auf die Seite p
 - Hist(p,i): Zeitpunkte der k letzten Zugriffe, die als nicht korreliert akzeptiert wurden.

Es gilt stets $\text{Last}(p) \geq \text{Hist}(p,1) > \text{Hist}(p,2) > \dots > \text{Hist}(p,k)$

- ❑ Parameter:
 - KorRefZeit: Zeitdauer mit der festgestellt wird, ob ein Zugriff korreliert ist.
- ❑ Besonderheit des Algorithmus
- ❑ Wenn ein Zugriff als nicht korreliert akzeptiert wird, verändert sich Hist folgendermaßen:



$\text{Hist}(p,1) = t$

$\text{Hist}(p,2) = \text{Last}(p)$

$\text{Hist}(p,i) = \text{Hist}(p,i-1) + \text{KorSkew}, 2 \leq i \leq k$

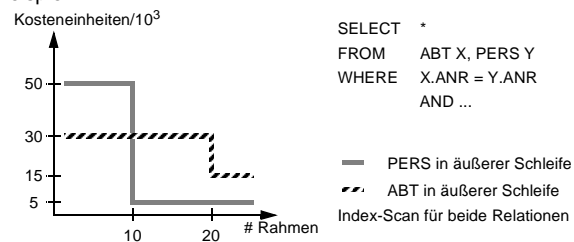
349.

9.3.7 Ersetzungsverfahren mit Kontextwissen

- ❑ Probleme bei LRU-ähnlichen Verfahren
 - T_1 : 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 ($\# \text{Seiten} > \# \text{Rahmen}$)
→ internes Thrashing
 - T_1 : zyklisches Referenzieren einer Seitenmenge ($\# \text{Seiten} < \# \text{Rahmen}$)
Interferenz durch T_i bei schnellerer Anforderung (stealing)
→ externes Thrashing
- ❑ Beachte: diese Probleme treten zum Teil nicht mehr bei LRU-K auf, welche?
- ❑ Mechanismen gegen Thrashing: WS-Modell
 - Scheduler versucht für T_i wenigstens $\sigma = W(t, w_i)$ Rahmen zu reservieren
 - Wenn Loop $> w$, versucht WS w Rahmen zu reservieren
 - Bei sequentiellem Scan genügt ein Rahmen (dann aber kein Prefetching !)
 - WS-Modell: teure Implementierung

350.

- ❑ **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 Puffer
- ❑ Optimierer berechnet HSS für die verschiedenen Zugriffspläne (Abschätzung der #Rahmen)
- ❑ Beispiel:

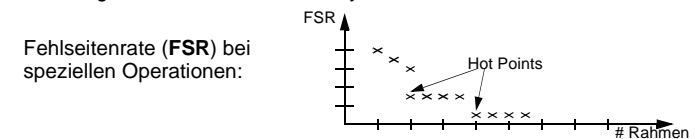


- ❑ Anwendungscharakteristika
 - Berücksichtigung der HSS in den Gesamtkosten
 - Auswahl abhängig von verfügbarer Puffergröße
 - Bindung zur Laufzeit möglich

352.

“Hot-Set”-Modell

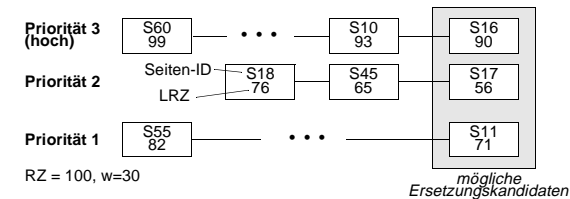
- ❑ Ausnutzung von Kontextwissen bei mengenorientierten Anforderungen
 - Verbesserung in relationalen DBS möglich
- ❑ Zugriffspläne durch Anfrageoptimierer
 - Zugriffsscharakteristik/Menge der referenzierten Seiten kann bei der Erstellung von Plänen vorausgesagt/abgeschätzt werden.
 - Zugriffsmuster 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 Referenzyklus



351.

Prioritätsgesteuerte Seitenersetzung

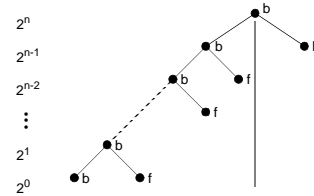
- ❑ Bevorzugung bestimmter Transaktionstypen/DB-Partitionen oft erwünscht
 - ❑ Berücksichtigung von Prioritäten bei der Pufferverwaltung
 - ❑ Verfahren PRIORITY-LRU:
 - pro Prioritätsstufe eigene dynamische Pufferpartition (z. B. LRU-Puffer)
 - Priorität einer Seite bestimmt durch DB-Partition bzw. durch (maximale) Priorität referenzierender Transaktionen
 - ersetzt wird Seite aus der Partition mit der geringsten Priorität
- Ausnahme: die zuletzt referenzierten Seiten sollen (unabhängig von ihrer Priorität) nicht ersetzt werden



353.

9.4 Pufferverwaltung für Seiten variabler Größe

- Seitengröße
 - keine beliebigen Seiten (Fragmentierung, Abbildung auf Externspeicher)
 - Seitenlänge (SL) als Vielfaches eines Einheitsrasters (Transporteinheit, Rahmengröße im Puffer), Bsp.: $SL = 1, 2, 4, \dots, 2^n$ Rahmengröße ($n \leq 8$)
- Trennung von Seite und Rahmen (Rastergröße)
- Schnittstellenforderung: Zusammenhängende Speicherung der Seite im Puffer
- Buddy-System
 - Verwaltung variabler Bereiche: frei(f) / belegt(b) (festes Raster, hierarchischer Vergabemechanismus)
 - Suche nur in freien Bereichen (belegte Bereiche können nicht verschoben werden)
 - Zusammenfassung von freien Nachbarzellen aufwendig

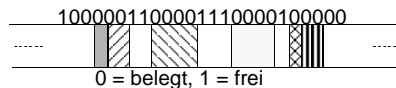


354.

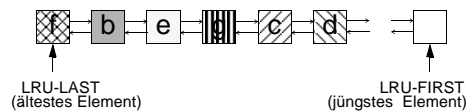
Der Algorithmus VAR-PAGE-LRU

Datenstrukturen:

- Freiliste:



- LRU-Kette:



- Algorithmus:

Kopiere Freiliste in eine temporäre 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

356.

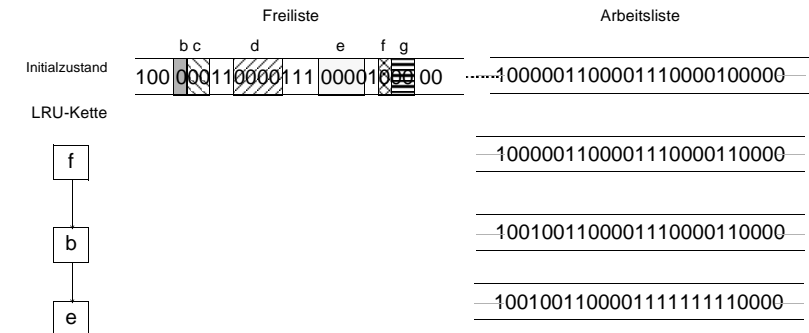
Anforderungen an DB-Pufferverwaltung

- 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?
 - 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: Partitionierung des DB-Puffers für Seiten gleicher Größe (und

355.

VAR-PAGE-LRU (Schritt 1)

- Suche nach 8 freien, zusammenhängenden Rahmen

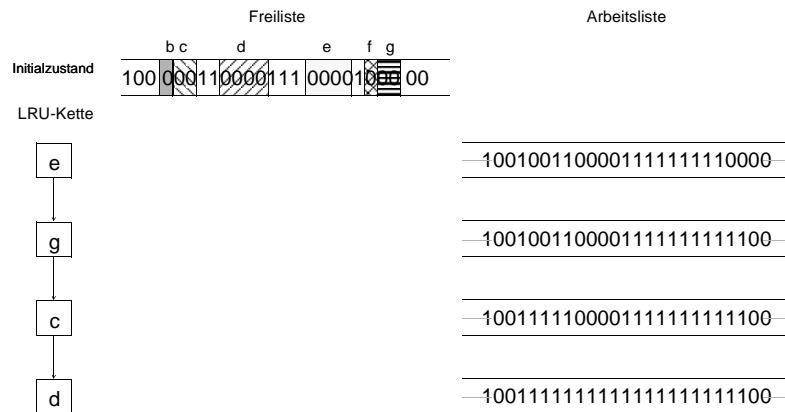


=> Ersetzung von Seite e ausreichend

357.

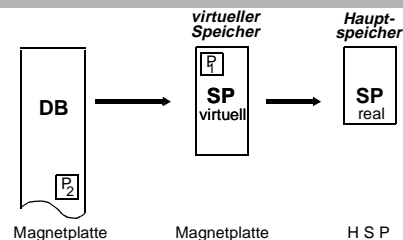
VAR-PAGE-LRU (Schritt 1)

- ☐ Suche nach 16 freien, zusammenhängenden Rahmen



358.

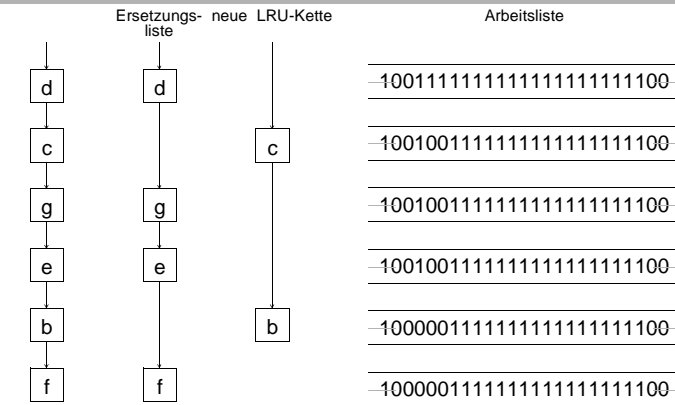
Seitenersetzung bei virtuellem Speicher



- ❑ 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

360.

VAR-PAGE-LRU (Schritt 2)



- ☐ Laufe die Liste der zu ersetzenden Seiten zurück und markiere die in der Liste unnötig aufgenommene Seiten.

359.

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!
- ❑ Suche im Puffer durch Hash-Verfahren
- ❑ Speicherzuteilung: global \Rightarrow alle Pufferrahmen für alle Transaktionen (Einfachheit, Stabilität ...)
- ❑ Behandlung geänderter Seiten: NOFORCE, asynchrones Ausschreiben
- ❑ Seitenersetzungsverfahren
 - LRU, LRU-k,...
- ❑ Erweiterte Ersetzungsverfahren
 - Nutzung von Zugriffsinformationen des Anfrageoptimierers (Hot-Set-Modell)
 - Berücksichtigung von Prioritäten
- ❑ Ersetzung bei Seiten variabler Größe
 - VAR-PAGE-LRU relativ komplex

361.