4 Systempuffer (Buffer Pool)

Arbeitsweise und Eigenschaften

Dienste eines Systempuffers

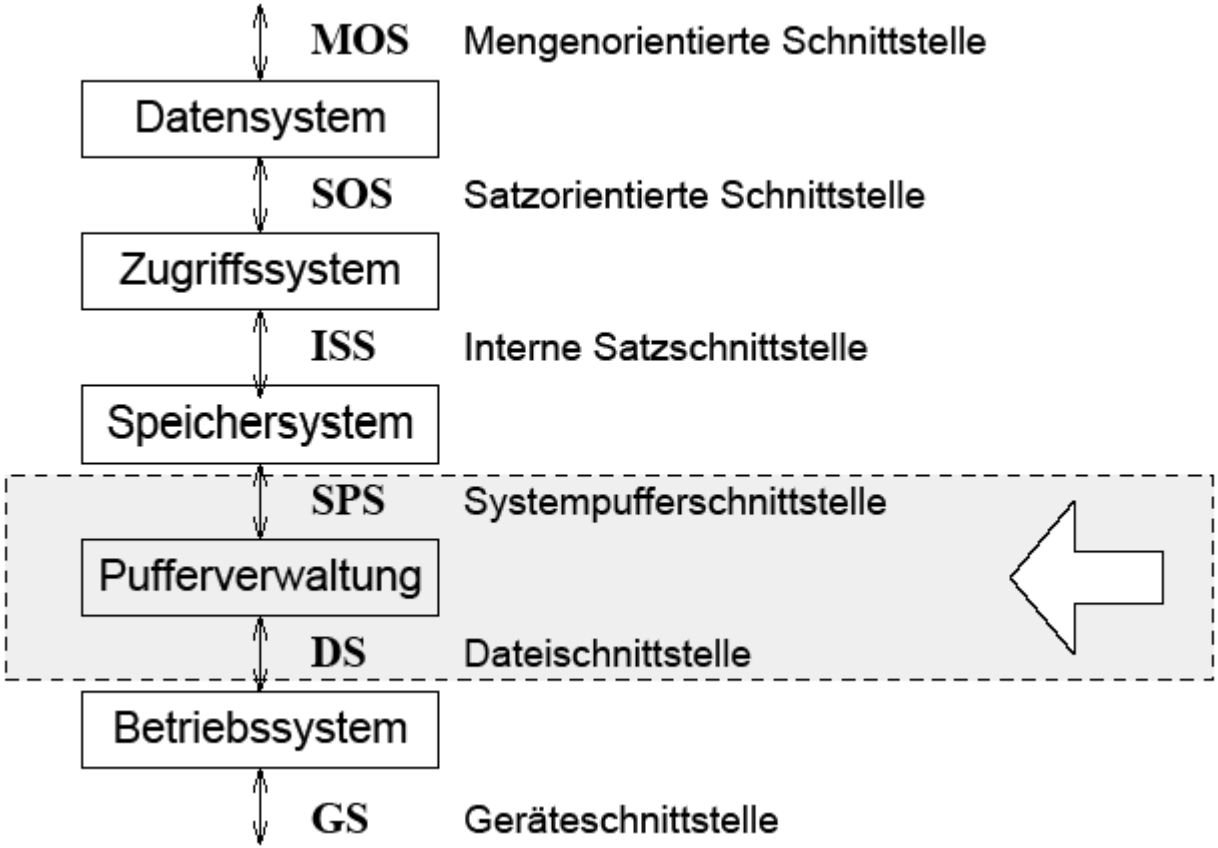
Suche im DB-Puffer

Einsatz von Seitenreferenzstrings

- Working-Set Modell
- LRU-Stacktiefenverteilung

Speicherzuteilung im Systempuffer

Seitenersetzungsstrategien


MOS
Mengenorientierte Schnittstelle
Datensystem
SOS
Satzorientierte Schnittstelle
Zugriffssystem
ISS
Interne Satzschnittstelle
Speichersystem
SPS
Systempufferschnittstelle
Pufferverwaltung
DS
Dateischnittstelle
Betriebssystem
GS
Geräteschnittstelle

Arbeitsweise und Rolle der Pufferverwaltung in einem DBS

- Lese- und Schreiboperationen werden über einen Systempuffer abgewickelt
- Puffer kann nur einen Bruchteil der gesamten Datenbank aufnehmen
- Puffer besteht aus
  - Seitenstrukturierte Segmente (Adressräume)
  - Pufferkontrollblock (Aufnahme von Verwaltungsinformation)

Eigenschaften einer DBS Systempufferverwaltung

- im Prinzip: normale Pufferverwaltung mit diversen Verdrängungsstrategien
- aber: beim DB-Puffer ist der Nutzer bekannt (die darüberliegende Schicht!), so dass Anwendungswissen (Kontextwissen) in die Pufferverwaltung einfließen kann

Lokalität als Maß für Seitenverdrängung

- LRU-Stacktiefe
- Working Set Modell

Systematik der Aufrufe

- Bereitstellen (Logische Referenz)
  - Bereitstellen der angeforderten Seite im Puffer; dabei evtl. Verdrängen einer “älteren” Seite gemäß der Ersetzungsstrategie und physisches Einlesen der neuen Seite
- FIX
  - Festhalten einer Seite im Puffer, so dass die Adressierbarkeit des Seiteninhalts gewährleistet ist (oft mit Bereitstellen automatisch durchgeführt)
- UNFIX
  - Aufheben eines FIX; macht die Seite frei für die Ersetzung
- Änderungsvermerk
  - Eintragen eines Vermerks in die Seite, dass sie beim Verdrängen aus dem Puffer physische geschrieben werden muss
  - evtl. Sicherstellen eines Before-Image, falls dies die Einbring-Strategie erfordert (der Änderungsvermerk muss VOR Ausführen der Änderung gemacht werden)
- Schreiben
  - Sofortiges Ausschreiben der Seite aus dem Puffer in die Datenbank

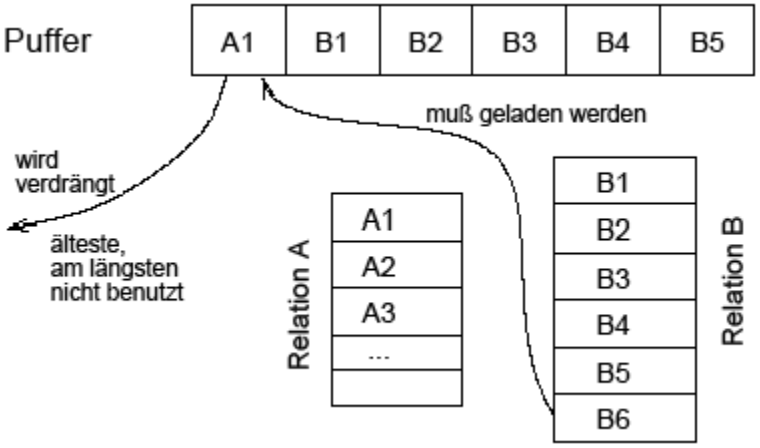
Seitenreferenz versus Adressierung

- nach einem FIX-Aufruf kann eine DB-Seite mehrfach bis zum UNFIX referenziert werden
  - unterschiedliches Seitenreferenzverhalten
  - andere Ersetzungsverfahren

Dateipuffer des Betriebssystems als DB-Puffer

- Zugriff auf Dateipuffer ist teuer (SVC: supervisor call)
- DB-spezifische Referenzmuster können nicht mehr gezielt genutzt werden (z. B. zyklisch sequentielle oder baumartige Zugriffsfolgen)
- keine geeignete Schnittstelle für Pre-Fetching:
  aufgrund von Seiteninhalten oder Referenzmustern ist (teilweise) eine Voraussage des Referenzverhaltens möglich;
  durch Pre-Fetching lässt sich in solchen Fällen eine Leistungssteigerung erzielen
- Selektives Ausschreiben von Seiten zu bestimmten Zeitpunkten (z. B. für Logging) ist nicht immer möglich in existierenden Dateisystemen (globales “sync” ist zu teuer)

--> DBVS muss eigene Pufferverwaltung realisieren



Beispiel

- Natürlicher Verbund von Relationen A und B (zugehörige Folge von Seiten A_i und B_j)
- Implementierung: Nested-Loop-Join

Ablauf

- FIFO: A1 verdrängt, da älteste Seite im Puffer
- LRU: A1 verdrängt, da diese Seite im ersten Schritt beim Auslesen des Vergleichstuples benötigt wurde
- Problem
  - A1 wird im nächsten Schritt gerade wieder benötigt
  - "Aufschaukeln": um A2 laden zu können, muss B1 entfernt werden (wird aber im nächsten Schritt wieder benötigt), usw.

Anforderung

- Logische Seitenreferenz (Seite i aus Segment j)

Fall 1: Seite im Puffer

- Geringer Aufwand (» 100 Instr.), um
  - Seite zu finden
  - Wartungsoperationen im PKB durchzuführen
  - Pufferadresse an rufende Komponente zu Übergeben

Fall 2: Seite nicht im Puffer

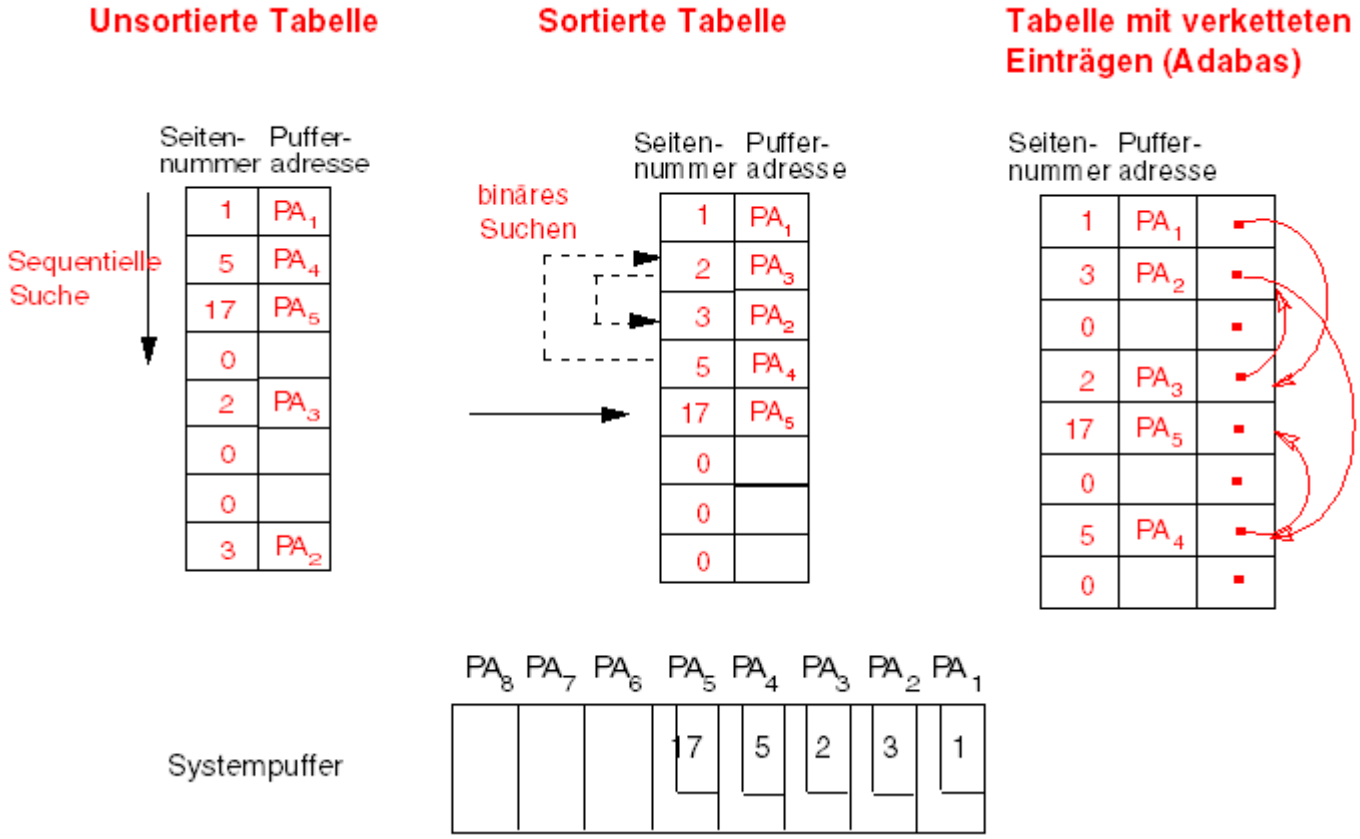
- Logische Seitenreferenz führt zu einer **physischen Seitenreferenz.**
- erfolglose Suche im Puffer
- zwei E/A-Vorgänge
  - Auswahl einer Seite zur Verdrängung und Herausschreiben der Seite bei Änderungsvermerk (nur bei Speicherknappheit)
  - neue Seite lesen

Forderung

- hoch effizient, da der Vorgang extrem häufig vorkommt !

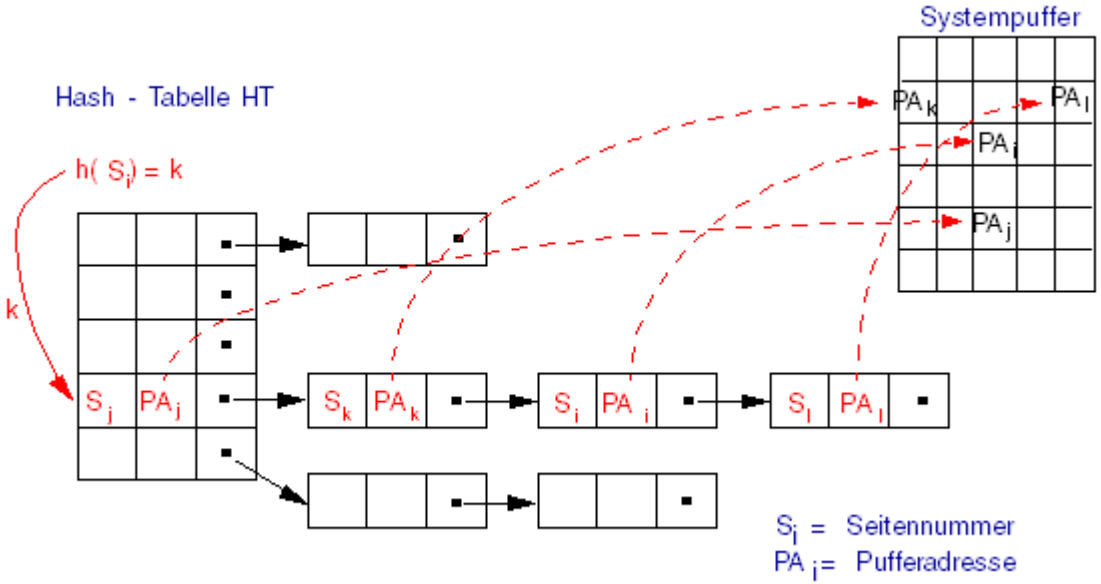
Suchstrategien

- Direkte Suche im Datenbankpuffer
  - sequentielles Durchsuchen
  - in jedem Seitenkopf wird nachgeprüft, ob die gesuchte Seite gefunden ist
  - sehr hoher Suchaufwand
  - Gefahr vieler Paging-Fehler bei virtuellen Speichern
- Indirekte Suche über Hilfsstrukturen
  (ein Eintrag pro Seite im Puffer)
  - unsortierte oder sortierte Tabelle
  - Tabelle mit verketteten Einträgen
  - Suchbäume (z. B. AVL-, m-Weg-Bäume)
  - Hash-Tabelle mit †berlaufketten


Unsortierte Tabelle
Seiten-nummer
Puffer-adresse
Sequentielle Suche
1 PA1
5 PA4
17 PA5
0
2 PA3
0
0
3 PA2
Sortierte Tabelle
Seiten-nummer
Puffer-adresse
binäres Suchen
1 PA1
2 PA3
3 PA2
5 PA4
17 PA5
0
0
0
Tabelle mit verketteten Einträgen (Adabas)
Seiten-nummer
Puffer-adresse
1 PA1
3 PA2
0
2 PA3
17 PA5
0
5 PA4
0
PA8 PA7 PA6 PA5 PA4 PA3 PA2 PA1
Systempuffer
17 5 2 3 1

Prinzip

- Seitennummern werden über Hashfunktion auf Pufferadresse abgebildet
- Synonyme werden verkettet
- Ein-/Auslagern einer Seite impliziert Ein-/Austragen des entsprechenden Eintrags



Direkte Suche

- nicht praktikabel

Unsortierte Tabelle

- Aufwand im Erfolgsfall: N/2 Einträge
- Aufwand bei Misserfolg: N Einträge

Sortierte Tabelle

- Aufwand bei Misserfolg: $\log_2 N$
- Einfüge- und Löschoperationen sehr aufwendig

Tabelle mit verketteten Einträgen

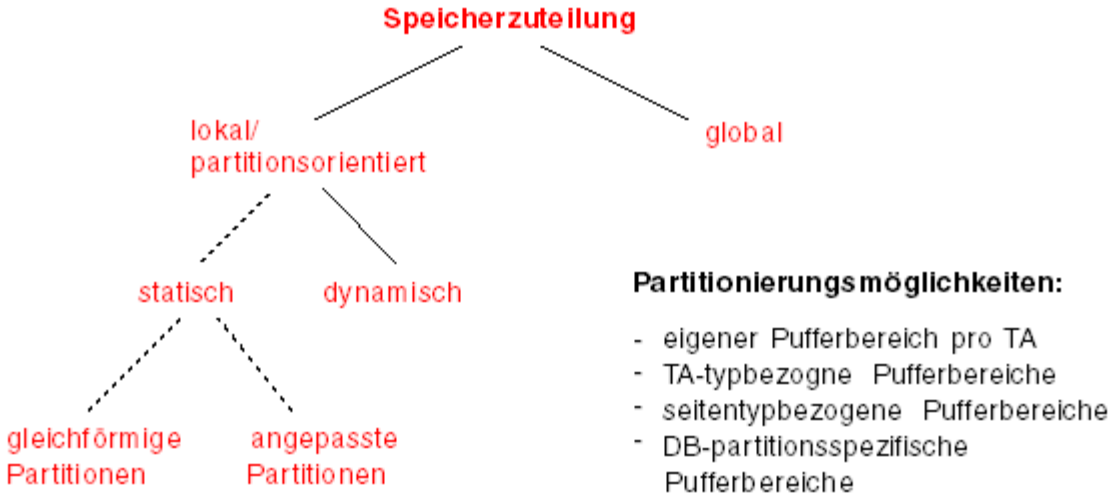
- gute Lösch- und Einfügeeigenschaften
- Aufwand bei Misserfolg: N/2
- Vorteil: kann zu LRU Reihenfolge verkettet werden

Hashverfahren

- beste Lösung

Besonderheiten

- Datenbankseiten können gemeinsam benutzt werden
  (lesende) Zugriffe auf dieselbe Seite sind durch mehrere Benutzer möglich
- Lokalität kommt nicht durch das Zugriffsverhalten eines, sondern aller Benutzer (Transaktionen) zustande
  Zugriffe einzelner sind weitgehend sequentiell
- unterschiedliche Behandlung von Datenseiten und Zugriffspfadverwaltungsseiten mit jeweils spezifischer Lokalität



Lokale Speicherzuteilung

- nur das aktuelle Referenzverhalten einer Transaktion wird berücksichtigt und Partitionen im Puffer werden gebildet

Globale Speicherzuteilung

- der gesamte Puffer steht allen aktiven Transaktionen gemeinsam zur Verfügung
- Die Speicherzuteilung wird vollständig durch die Ersetzungsstrategie bestimmt

Seitenbezogene (oder seitentypbezogene) Speicherzuteilung

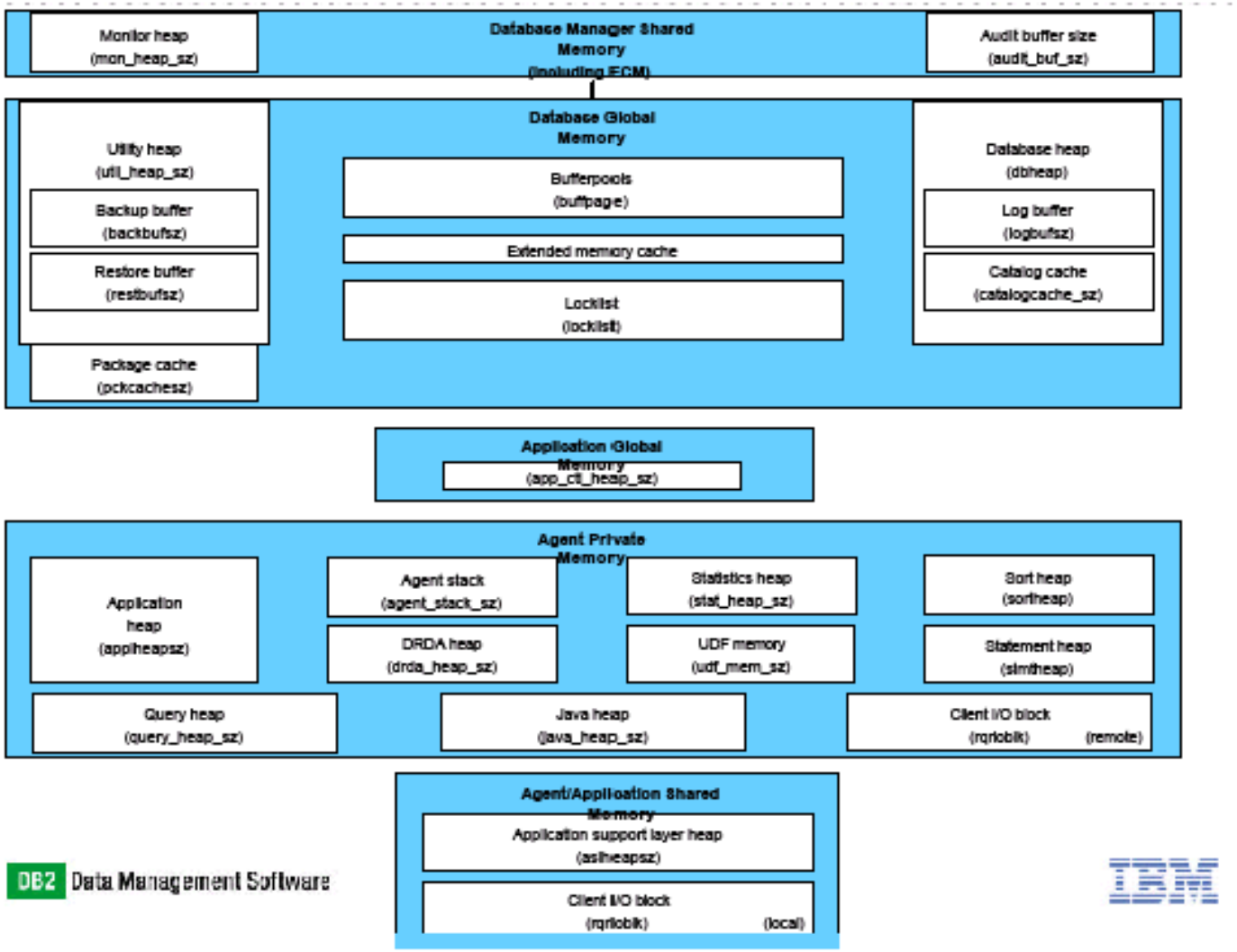
- für Datenseiten, Zugriffspfadseiten, Seiten der Freispeicherverwaltung usw. werden jeweils eigene Partitionen im Puffer gebildet

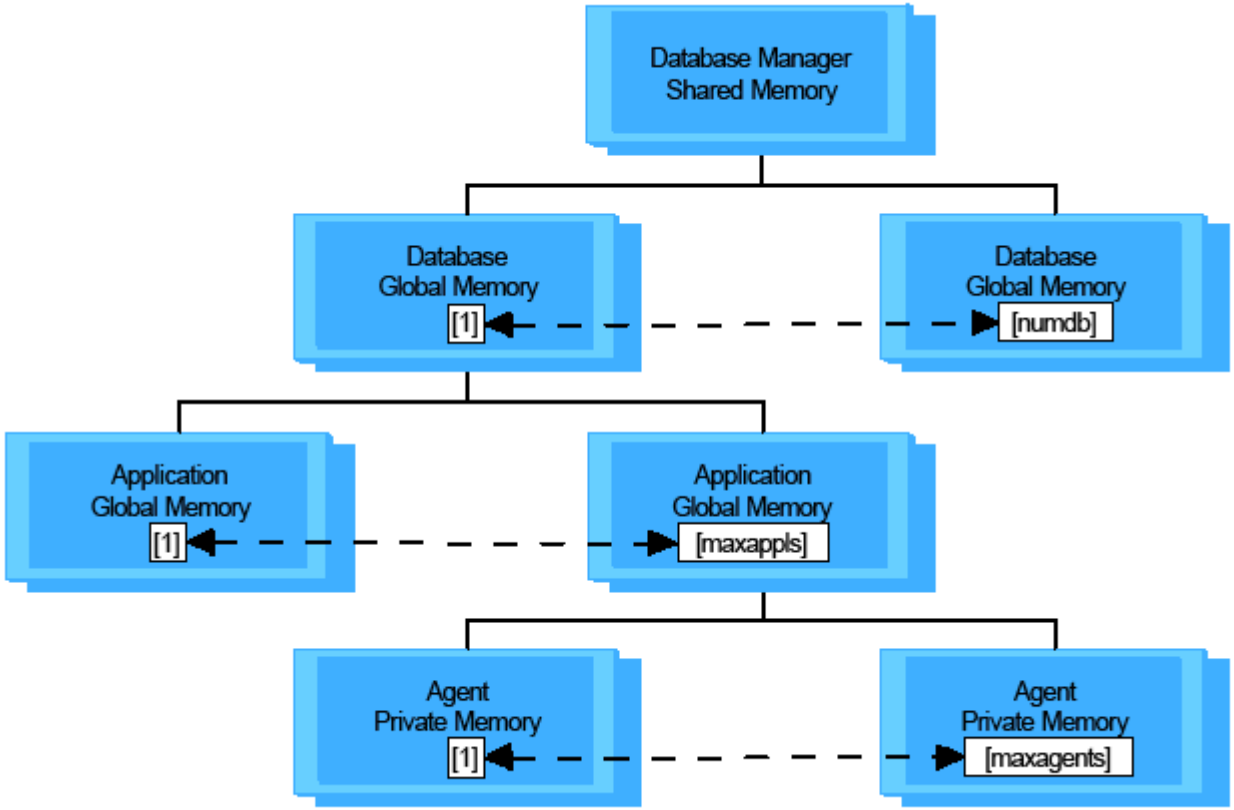
Statische Speicherzuteilung

- eine erforderliche Menge von Rahmen (Partition) muss verfügbar sein, bevor die Transaktion gestartet wird (preclaiming). **Uninteressant in DBS**.

Dynamische Speicherzuteilung

- Seiten werden zwischen Partitionen ausgetauscht (variable Partitionen)
- Beispiel Working-Set-Strategie: es wird versucht, einer Transaktion ihren Working-Set zur Verfügung zu halten (Wahl von t ist der kritische Faktor!)


Monitor heap
(mon_heap_sz)
Database Manager Shared Memory
(including FCM)
Audit buffer size
(audit_buf_sz)
Database Global Memory
Utility heap
(util_heap_sz)
Backup buffer
(backbufsz)
Restore buffer
(restbufsz)
Package cache
(pckcachesz)
Bufferpools
(buffpage)
Extended memory cache
Locklist
(locklist)
Database heap
(dbheap)
Log buffer
(logbufsz)
Catalog cache
(catalogcache_sz)
Application Global Memory
(app_ctl_heap_sz)
Agent Private Memory
Application heap
(appiheapsz)
Agent stack
(agent_stack_sz)
Statistics heap
(stat_heap_sz)
Sort heap
(sortheap)
DRDA heap
(drda_heap_sz)
UDF memory
(udf_mem_sz)
Statement heap
(stmtheap)
Query heap
(query_heap_sz)
Java heap
(java_heap_sz)
Client I/O block
(rqriolblk)
(remote)
Agent/Application Shared Memory
Application support layer heap
(aslheapsz)
Client I/O block
(rqriolblk)
(local)
DB2 Data Management Software
IBM


Database Manager
Shared Memory
Database
Global Memory
[1]
Database
Global Memory
[numdb]
Application
Global Memory
[1]
Application
Global Memory
[maxappls]
Agent
Private Memory
[1]
Agent
Private Memory
[maxagents]

Database Manager Shared Memory Set

- Stores all relevant information for a particular instance, such as lists of all active connections and security information

Database Shared Memory Set

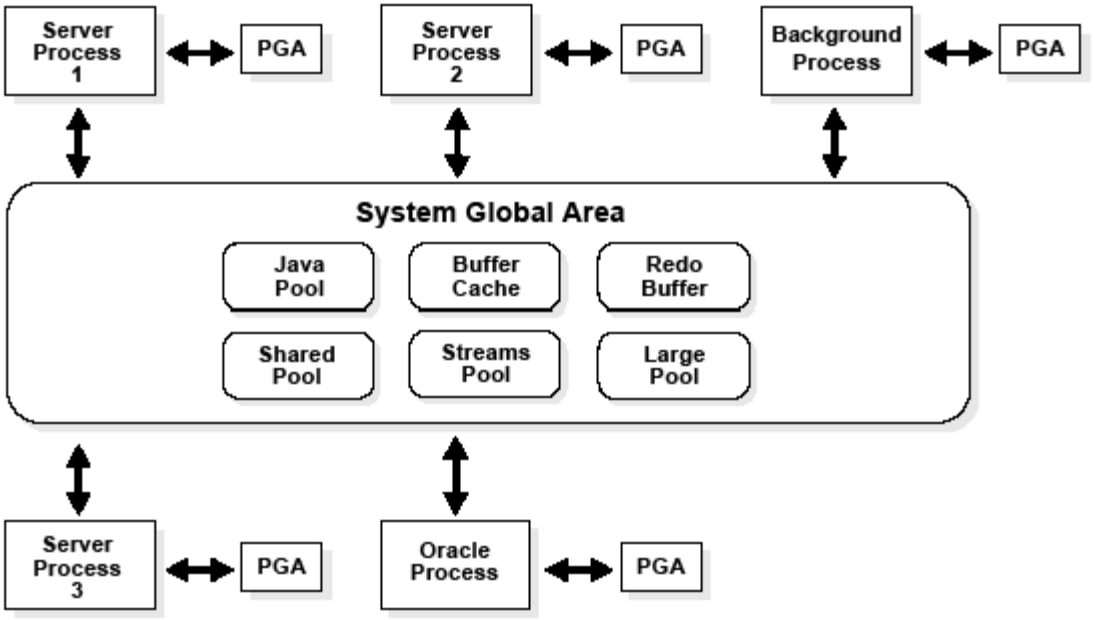
- Stores information relevant to a particular database, such as package caches, log buffers, and bufferpools

Application Shared Memory Set

- Stores information that is shared between DB2 and a particular application, primarily rows of data being passed to or from the database

Agent Private Memory Set

- Stores information that is used by DB2 to service a particular application, such as sort heaps, cursor information, and session contexts

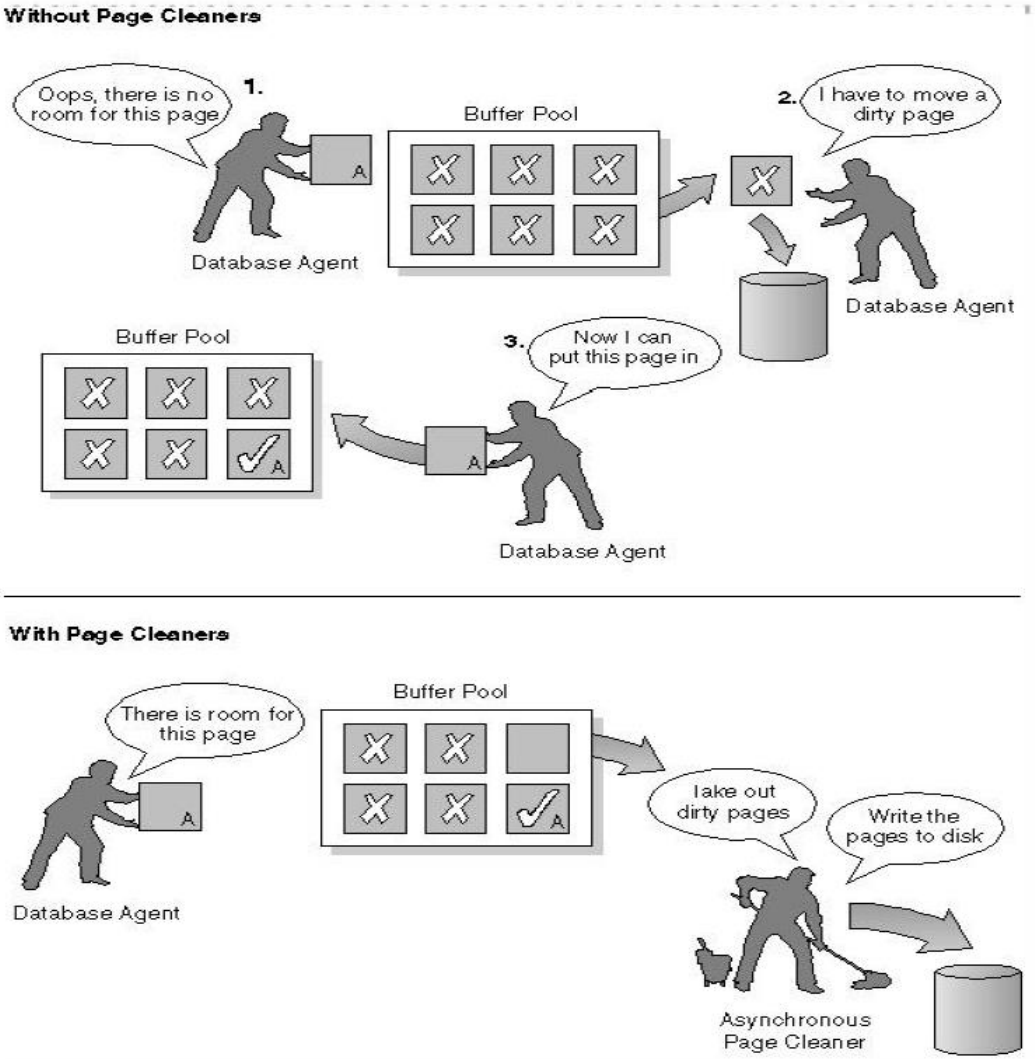


PGA (Program Global Area): individuell für jeden Datenbankprozess

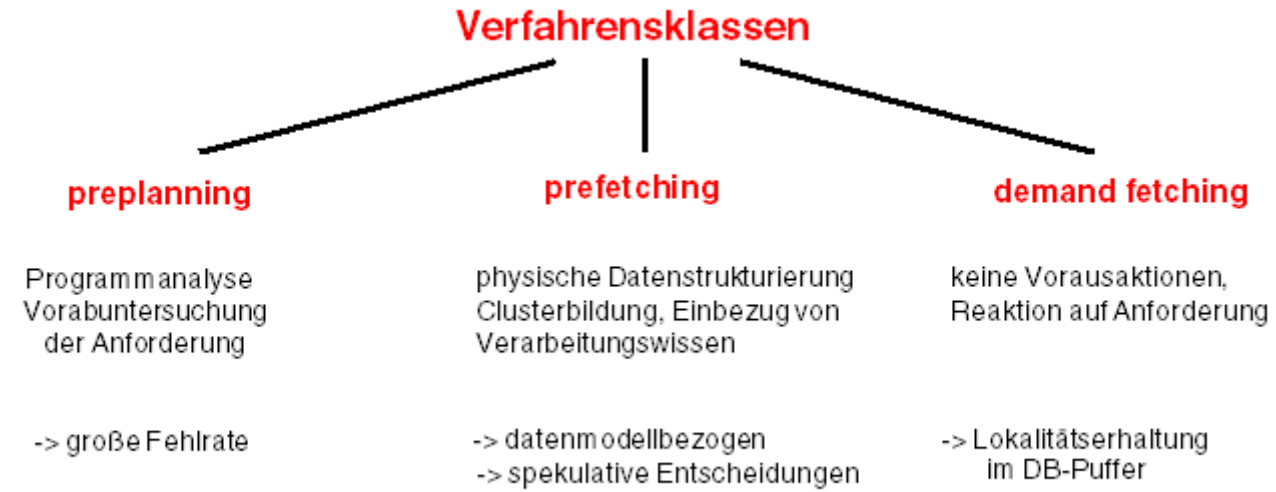
SGA (System Global Area): gemeinsamer Bereich für alle Prozesse bestimmt durch SGA_MAX_SIZE

Idee

- spezifische Threads, die Seiten laden bzw. verdrängen
- Parameter
  - Anteil an dirty pages im Systempuffer
  - Protokollierungsaufwand für Wiederherstellung nach System-Crash

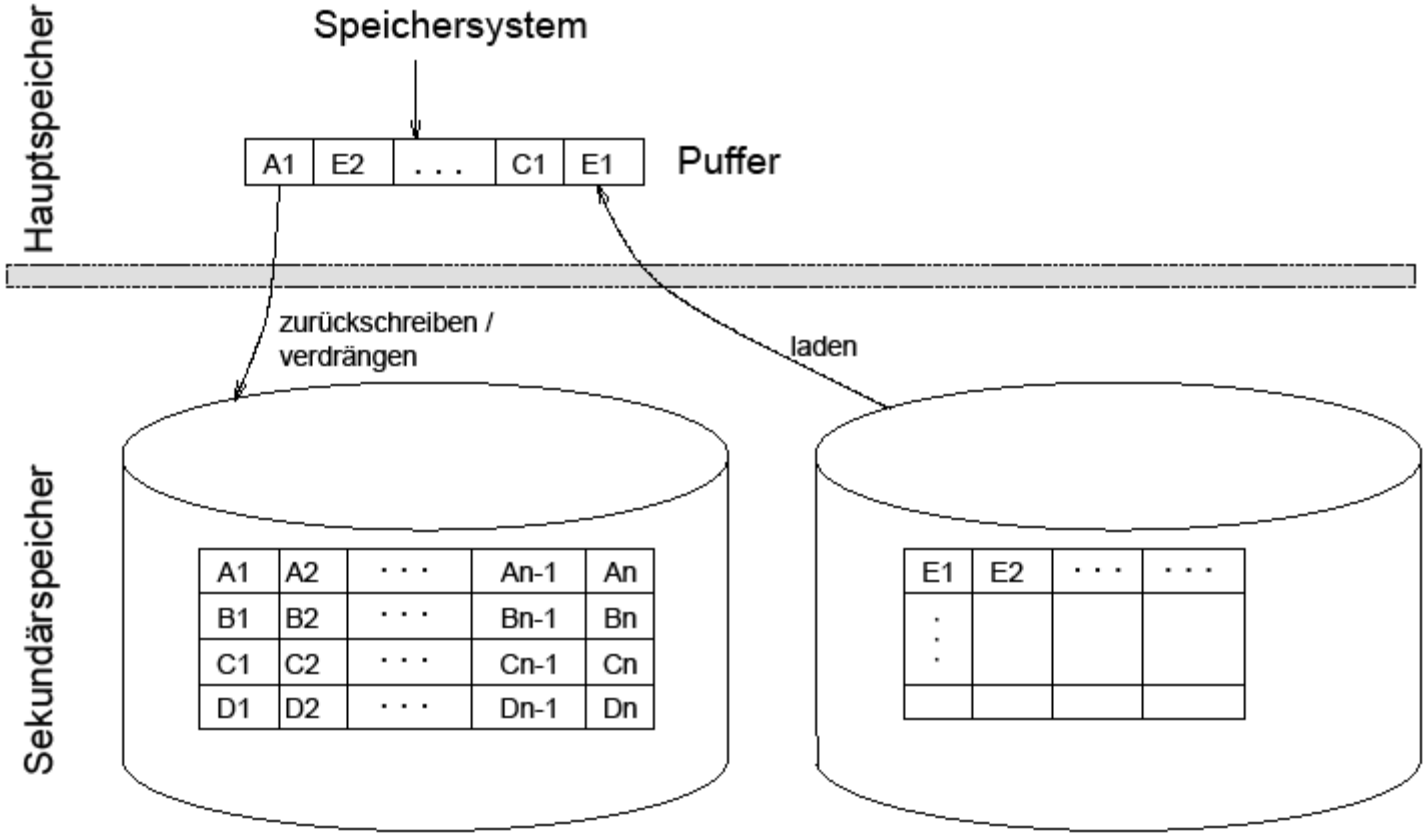


Klassifikation



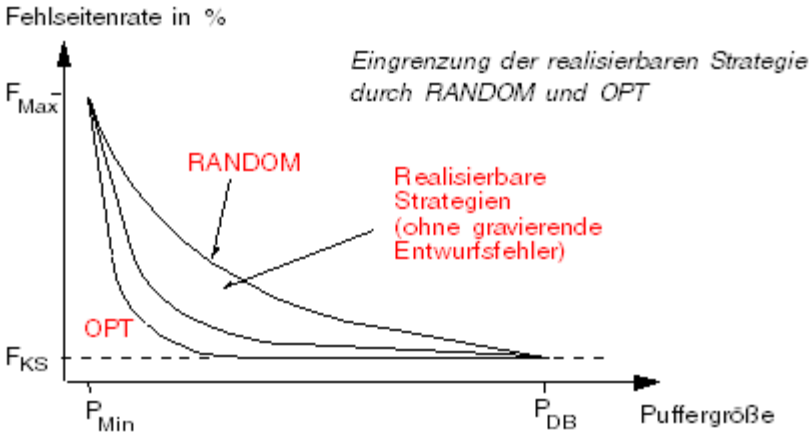
Grundannahme

- Referenzverhalten der nächsten Zukunft ist ähnlich dem der jüngsten Vergangenheit
- Prefetching: Index Prefetch, Sequential Prefetch, List Prefetch of data pages


Hauptspeicher
Speichersystem
A1
E2
. . .
C1
E1
Puffer
zurückschreiben / verdrängen
laden
Sekundärspeicher
A1
A2
· · ·
An-1
An
B1
B2
· · ·
Bn-1
Bn
C1
C2
· · ·
Cn-1
Cn
D1
D2
· · ·
Dn-1
Dn
E1
E2
· · ·
· · ·

Eingrenzung

- obere Grenze: RANDOM
- untere Grenze: OPT (nach Belady)



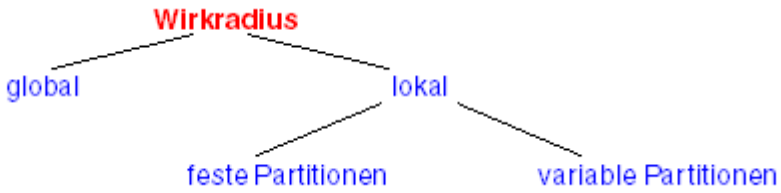
P_{Min} = minimale Größe des Systempuffers

P_{DB} = Datenbankgröße

F_{KS} = Fehlseitenrate bei Kaltstart (jede angesprocheneSeite muß zumindest einmal eingelesen werden)

Wirkradius

- abhängig von der Speicherallokation



Alter einer Seite

- gemessen in logischen Referenzen, nicht in Zeiteinheiten

Entscheidungskriterien für eine Ersetzungsstrategie

- Alter einer Seite
  - Alter seit der Einlagerung (globale Strategie)
  - Alter seit dem letzten Referenzzeitpunkt (Strategie des jüngsten Verhaltens)
  - Alter wird nicht berücksichtigt
- Referenzierung einer Seite
  - Berücksichtigung aller Referenzen (globale Strategie)
  - Berücksichtigung der letzten Referenzen (Strategie des jüngsten Verhaltens)
  - Berücksichtigung keiner Referenz

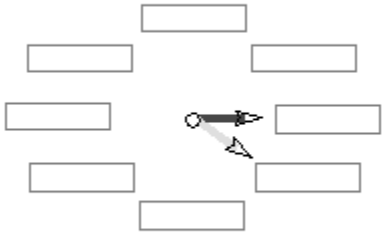
Ziel

- Annäherung an optimale Strategie

Verfahren	Prinzip	Alter	Anzahl
FIFO	älteste Seite ersetzt	G	–
LFU (least frequently used)	Seite mit geringster Häufigkeit ersetzen	–	G
LRU (least recently used)	Seite ersetzen, die am längsten nicht referenziert wurde (System R)	J	J
DGCLOCK (dyn. generalized clock)	Protokollierung der Ersetzungshäufigkeiten wichtiger Seiten	G	JG
LRD (least reference density)	Ersetzung der Seite mit geringster Referenzdichte	JG	G

Ersetzung der ältesten Seite im DB-Puffer

- Veranschaulichung durch kreisförmig umlaufenden Uhrzeiger
  - Zeiger zeigt auf älteste Seite
  - Bei Fehlseitenbedingung wird diese Seite ersetzt und der Zeiger auf die nächste Seite fortgeschaltet



Merkmale

- Unabhängigkeit vom Referenzverhalten
  (nur das Alter seit der Einlagerung ist entscheidend)

Bewertung

+ bei strikt sequentiellem Zugriffsverhalten
- bei Direktzugriff
  (häufig benutzte Seiten sollen ja gerade im Puffer bleiben und dort “alt” werden)

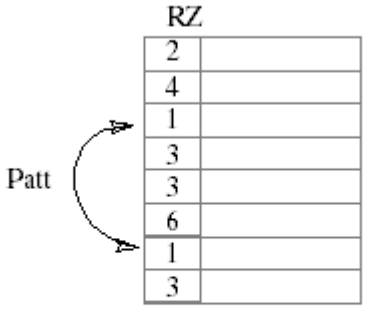
Erweiterung: CLOCK, GCLOCK und DGCLOCK

Ersetzung der Seite mit niedrigster Referenzhäufigkeit

- Führen eines Referenzzählers pro Seite im DB-Puffer

Merkmale

- nur Referenzverhalten geht ein, nicht das Alter!
- mögliche Pattsituationen müssen durch eine Sekundärstrategie aufgelöst werden
- beim sequentiellen Lesen wird jede Seite einmal referenziert Strategie nicht anwendbar

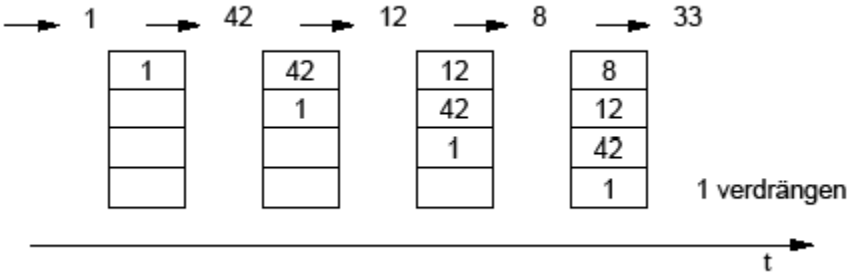


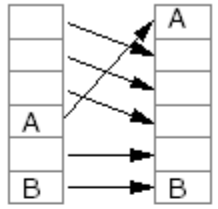
Bewertung

+ häufig benutzte Seiten werden im Puffer gehalten
- Seiten, die punktuell sehr intensiv benutzt werden und dann nicht mehr, sind praktisch nicht zu verdrängen

Ersetzung der Seite, die am längsten nicht mehr referenziert wurde

- Seiten werden als LRU-Stack verwaltet
- bei jeder Referenz kommt die Seite in die oberste Position

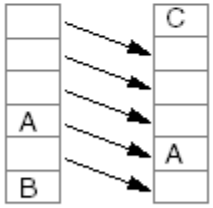




Referenz auf Seite A,
die im Puffer
gefunden wird.

Merkmale

- bewertet das Alter seit der letzten Referenz, nicht seit dem Einlagern
- geht bei sequentiellem Zugriff in FIFO über



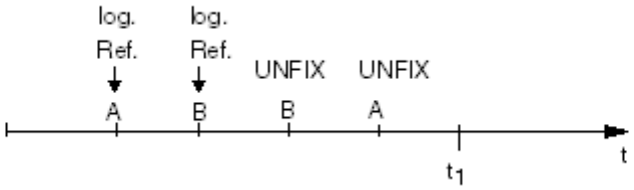
Referenz auf Seite C,
die nicht im Puffer gefunden wird;
Seite B wird ersetzt

Beachte Besonderheit der "letzten Referenz" beim DB-Puffer

- logische Referenz
- unfix-Operation

Zwei Varianten bei der "Realisierung" von LRU

- Least Recently Unfixed (LRUN)
- Least Recently Referenced (LRR)



Beispiel

- Zum Zeitpunkt t_1 wird ersetzt bei
  - Least Recently Unfixed: Seite B
  - Least Recently Referenced: Seite A

Idee

- Verbesserung durch Berücksichtigung der letzten K Referenzierungszeitpunkte
- Bestimmung des mittleren Zeitabstands zwischen den letzten K Referenzen
- K-Distanz $b_t(p,K)$: "LRU-K-Alter"
  - Zeit t, Referentierungsfolge r_1, r_2, ..., r_t
  - $b_t(p,K)$ ist Rückwärtsdistanz von t zur K-ten Referenz

$$b_t(p,K) = \begin{cases} g & \text{wenn rt-g die Seite p zum K-ten Mal referenziert} \\ \infty & \text{wenn p nicht mind. K mal in } r_1,\ldots r_t \text{ vorkommt} \end{cases}$$

- Ersetzung der Seite p mit $b_t(p,K)$ ist maximal

Bewertung

- berücksichtigt aktuelle Referenzierungen häufiger als ältere
- LRU-1 entspricht LRU
- typisch: LRU-2

Zeitliche Abfolge von Seitenanforderungen

- Logische Seitenanforderungen -> Logische Seitenreferenzstrings
- Physische Seitenanforderungen -> Physische Seitenreferenzstrings

Problem

- Erzeugung eines optimalen physischen Seitenreferenzstrings
  (minimale Anzahl von Zugriffen)
- nicht alle Seiten im Datenbankpuffer sind auch im physischen Hauptspeicher !

Typische Referenzmuster

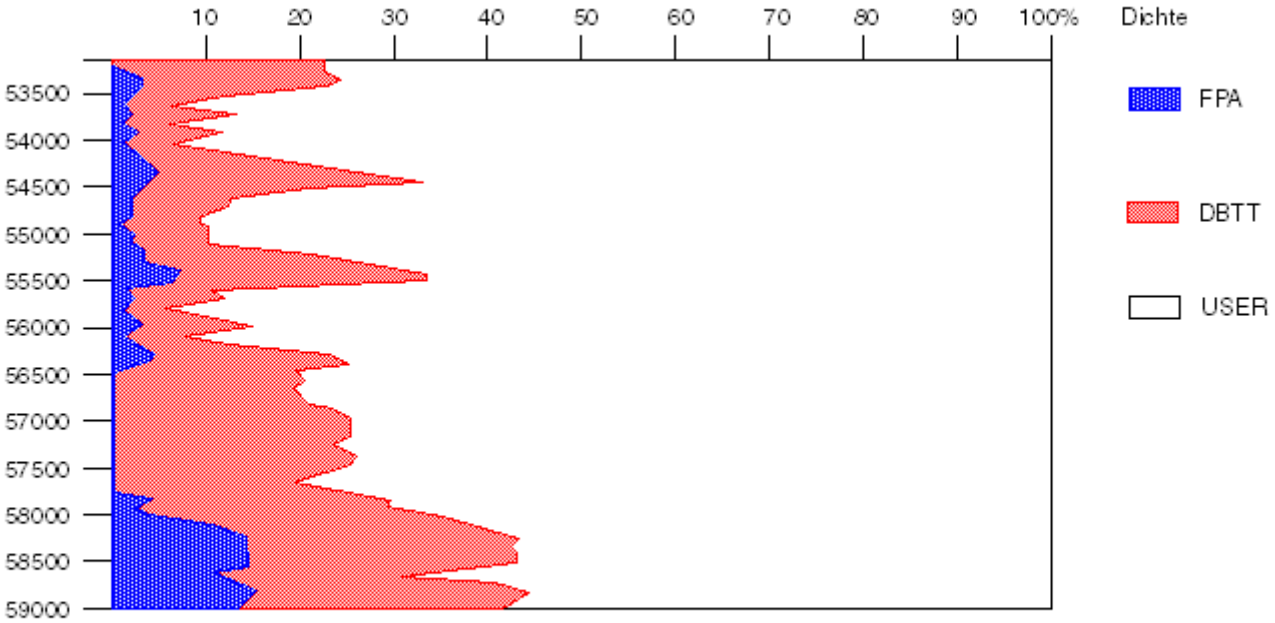
- Sequentielle Suche
  (Bsp.: Durchsuchen ganzer Datenbestände)
- Hierarchische Pfade
  (Bsp.: Suchen in baumstrukturierten Zugriffspfaden)
- Zyklische Pfade
  (Bsp.: Abarbeiten von Mengen)

Relative Referenzmatrix (Seitenbenutzungsstatistik)

- 12 TransaktionsTypen (TT1-TT12) auf 13 DB-Partitionen (P1-P13)
- ca. 17500 Transaktionen, 1Million Seitenreferenzen auf ca. 66.000 versch. 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
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
% referenced	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

Prozentuale Verteilung der Referenzen auf Seiten unterschiedlichen Typs



Anteil der Seitentypen:

FPA = 0,1%, DBTT = 6,1%, USER = 93,8%

Prinzip der Lokalität

- erhöhte Wiederbenutzungswahrscheinlichkeit für schon einmal referenzierte Seiten
- Zugriff immer nur auf eine kleine Untermenge der im Adressraum vorhandenen Seiten
- grundlegende Voraussetzung für
  - effektive DB-Pufferverwaltung (Seitenersetzung)
  - Einsatz von Speicherhierarchien

unterschiedliche Lokalitätsmaße

- Working-Set-Modell
- LRU-Stacktiefe

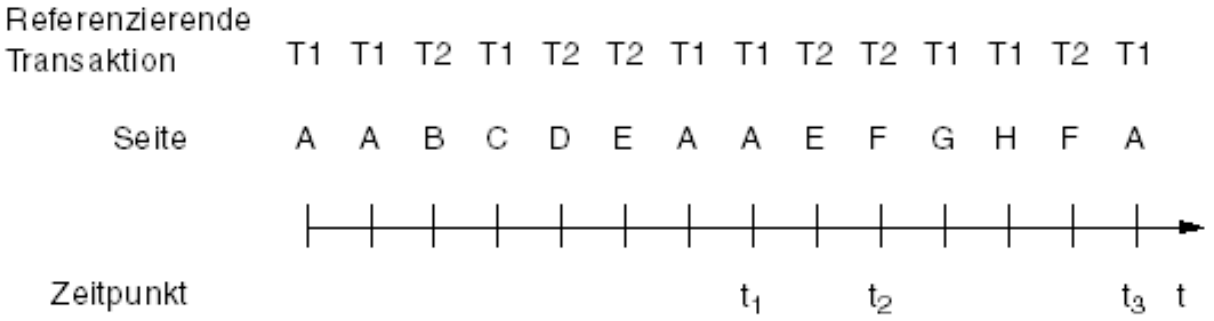
-> notwendig für partitionsorientierte Speicherzuteilung!

Begriffsbildung

- Working-Set W(t, τ) ist Menge der verschiedenen Seiten, die von dem betrachteten Programm innerhalb seiner τ letzten Referenzen, vom Zeitpunkt t aus rückwärts gerechnet, angesprochen wurden.
- Fenstergröße (window size): τ
- Working-Set-Größe: $w(t,\tau) = |W(t,\tau)|$
- Aktuelle Lokalität: $AL(t,\tau) = \frac{w(t,\tau)}{\tau}$
- Durchschnittliche Lokalität: $L(\tau) = \frac{1}{n} \cdot \left(\sum_{t=1}^{n} AL(t,\tau) \right)$

  (n = Länge des Referenzstrings)

Beispiele für Working Sets



W(t, τ) mit $\tau = 5$:

T1: W $(t_1,5)$ = {A, C}, w $(t_1,5)$ = |W $(t_1,5)$| = 2
T2: W $(t_1,5)$ = {B, D, E}, w $(t_1,5)$ = |W $(t_1,5)$| = 3

T1: W $(t_2,5)$ = {A, C}, w $(t_2,5)$ = |W $(t_2,5)$| = 2
T2: W $(t_2,5)$ = {B, D, E, F}, w $(t_2,5)$ = |W $(t_2,5)$| = 4

T1: W $(t_3,5)$ = {A, G, H}, w $(t_3,5)$ = |W $(t_3,5)$| = 3
T2: W $(t_3,5)$ = {D, E, F}, w $(t_3,5)$ = |W $(t_3,5)$| = 3

AL(t, τ) mit $\tau = 5$:

T1: AL $(t_1,5)$ = 0.4
T2: AL $(t_1,5)$ = 0.6

T1: AL $(t_2,5)$ = 0.4
T2: AL $(t_2,5)$ = 0.8

T1: AL $(t_3,5)$ = 0.6
T2: AL $(t_3,5)$ = 0.6

L(τ) mit $\tau = 5$:

am Beispiel des Working-Set-Ansatzes

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

Verdrängung

- 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)** gilt

Fenstergröße kritischer Parameter -> Thrashing-Gefahr

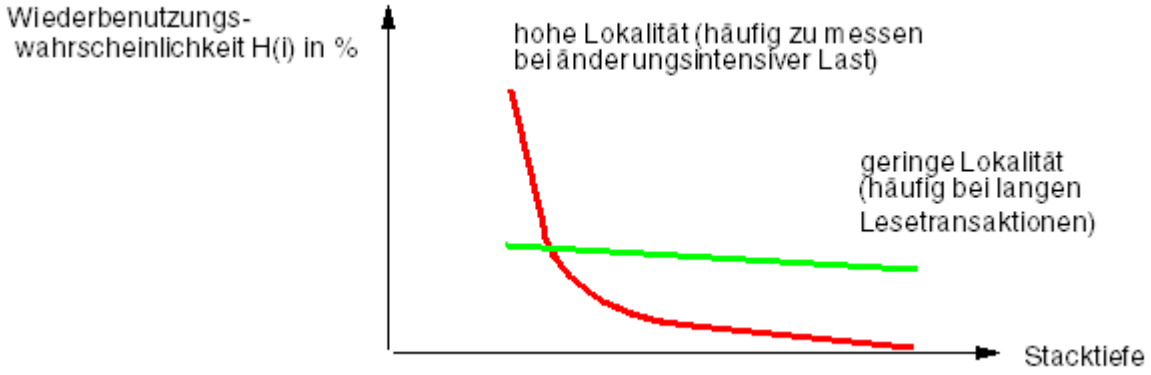
Prinzip

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

Bestimmung der Stacktiefenverteilung

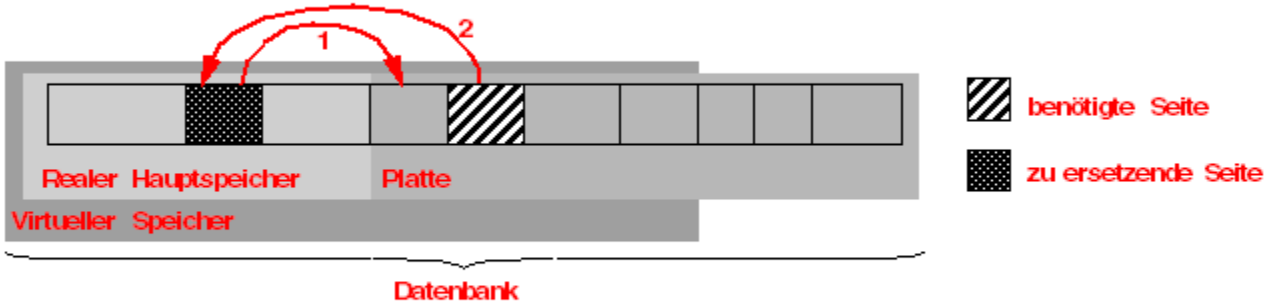
- pro Stackposition wird Zähler geführt
- Rereferenz einer Seite führt zur Zählererhöhung für die jeweilige Stackposition

-> Zählerwerte entsprechen der Wiederbenutzungshäufigkeit



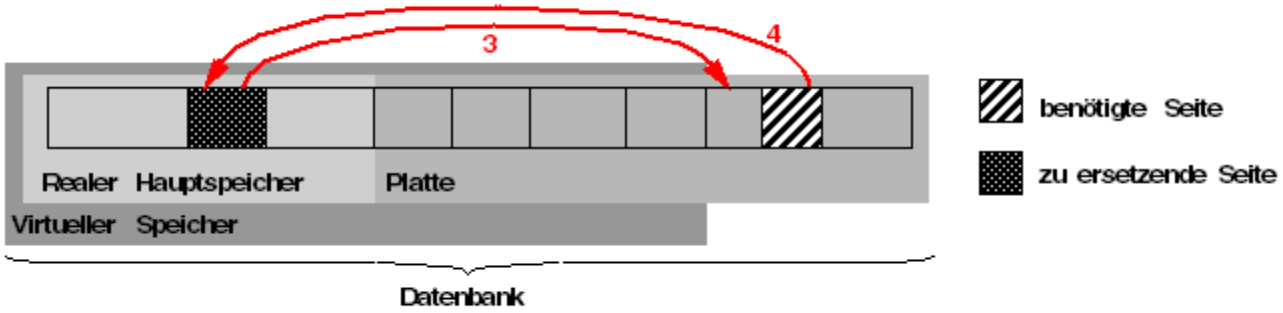
Page Fault

- Die benötigte Seite befindet sich zwar im DB-Puffer, die Pufferseite ist aber gerade ausgelagert.
- Das Betriebssystem muss die referenzierte Seite vom Hintergrundspeicher einlesen.



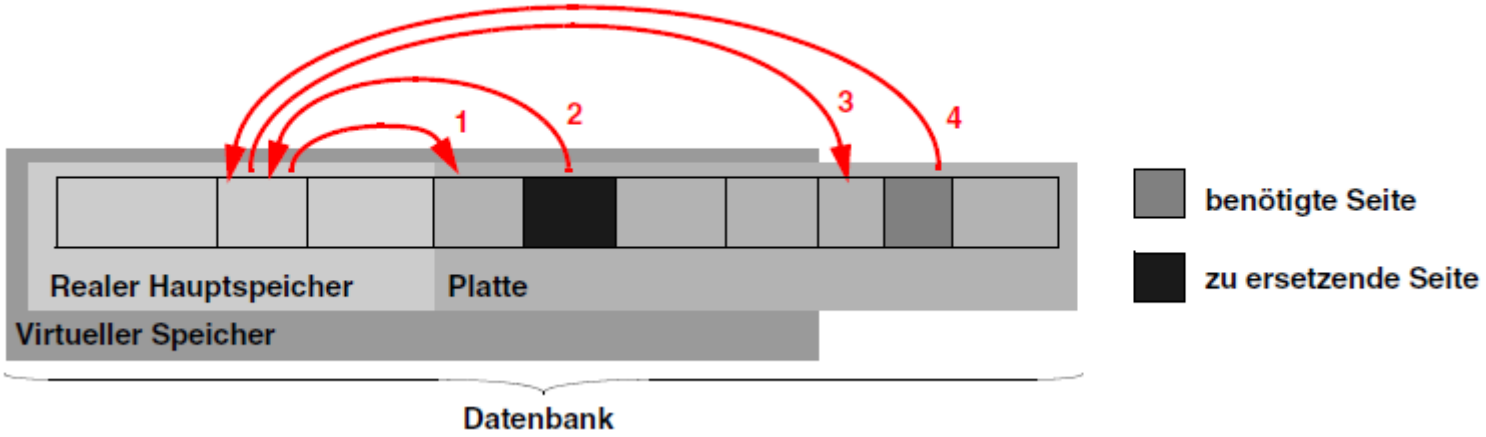
Database Fault

- Die benötigte Seite wird nicht im DB-Puffer aufgefunden.
- Die zu ersetzende Pufferseite befindet sich jedoch im Hauptspeicher, so dass sie freigegeben und bei Bedarf zurückgeschrieben werden kann.
- Die angeforderte Seite wird dann von der Datenbank eingelesen.



Double Page Fault

- Die benötigte Seite wird nicht im DB-Puffer aufgefunden
- Die zur Ersetzung ausgewählte Seite befindet sich nicht im Hauptspeicher.
- In diesem Fall muss zunächst die zu ersetzende Seite durch das Betriebssystem bereitgestellt werden, bevor ihre Freigabe und das Einlesen der angeforderten Seite erfolgen kann.



Eigenschaften eines DB-Systempuffers

- Nutzung von Kontextwissen zur Ermittlung der Ersetzungskandidaten
- Lokalität durch Working-Set Modell oder LRU-Stacktiefenverteilung

Algorithmen

- Suche einer Seite im Systempuffer
- Ermittlung der zu verdrängenden Seite

Ersetzungsstrategien

- klassisch ....
- wichtiger Unterschied: FIX und UNFIX-Semantik unterschiedlich zu einfacher Referenz !