Dresden Database
Systems Group

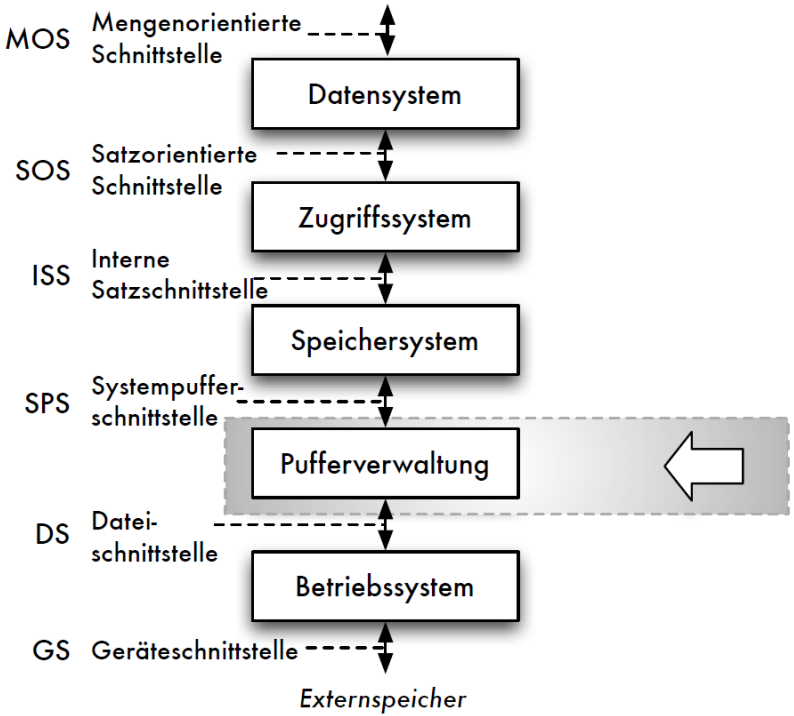
3. Systembuffer

Architektur von Datenbanksystemen I

Einordung und Aufgaben

Dresden Database Systems Group

Einordnung



Beschreibung

- Puffer: ausgezeichneter Bereich im Hauptspeicher
- In Pufferrahmen gegliedert, jeder Pufferrahmen kann Seite der Platte aufnehmen

Aufgaben

- Pufferverwaltung muss angeforderte Seite im Puffer suchen → **effiziente Suchverfahren**
- Parallele Datenbanktransaktionen → **geschickte Speicherzuteilung im Puffer**
- Puffer gefüllt: **adäquate Seitenersetzungsstrategien**

Beachte

- Unterschiede zwischen einem Betriebssystem-Puffer und einem Datenbank-Puffer

Arbeitsweise und Aufgaben/2

Dresden Database Systems Group

Rolle der Pufferverwaltung in einem DBS

- Lese- und Schreiboperationen werden über einen Systempuffer abgewickelt
- Puffer kann nur einen Bruchteil der gesamten Datenbank aufnehmen

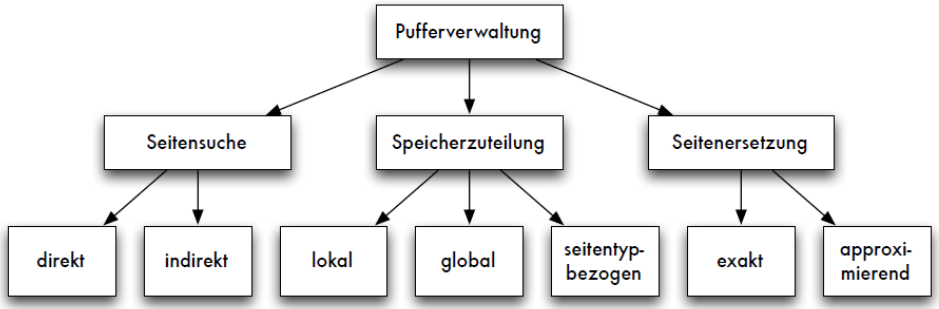
Eigenschaften eines DBMS-Puffers

- 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

Aufgaben im Überblick



Dienste eines Systempuffers

Dresden Database Systems Group

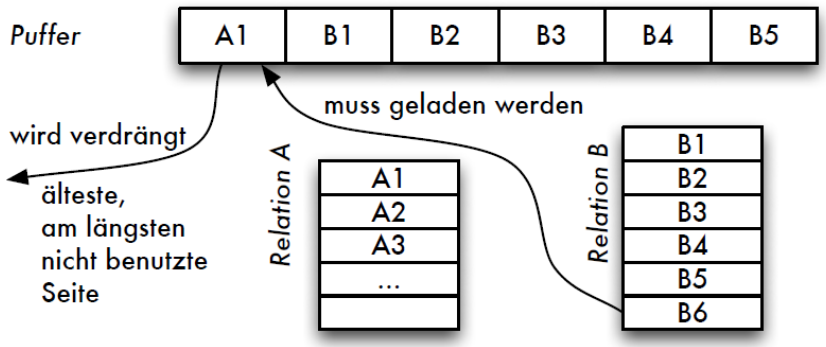
Systematik der Aufrufe

- 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

Mangelende Eignung des BS-Puffers

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 Seite nur im ersten Schritt beim Auslesen des ersten Vergleichstupels benötigt wurde

Problem

- Im nächsten Schritt wird das zweite Tupel von A1 benötigt
- Weiteres „Aufschaukeln“: um A1 laden zu können, muss B1 entfernt werden (im nächsten Schritt benötigt) usw.

Mangelende Eignung des BS-Puffers

Dresden Database Systems Group

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

Allgemeine Arbeitsweise

Dresden Database Systems Group

Anforderung (Schnittstelle für darüberliegende Schichten)

- Logische Seitenreferenz (Seite i aus Segment j)

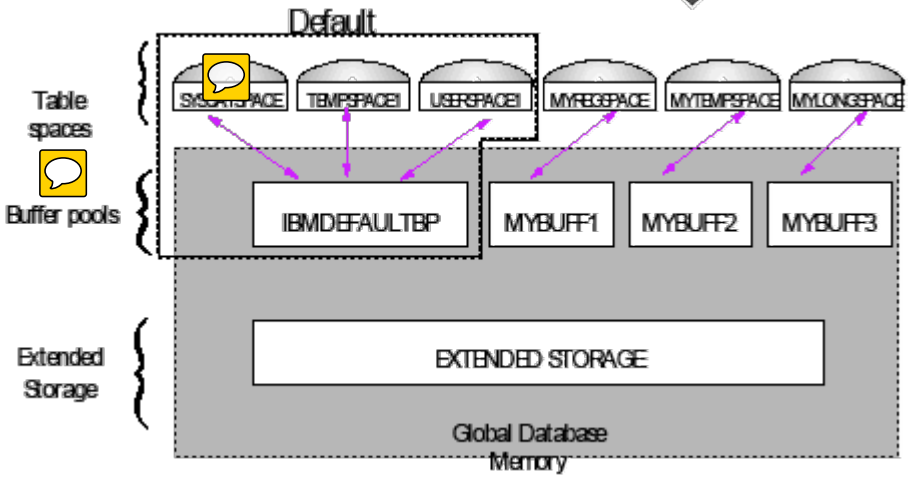
Fall 1: Seite im Puffer

- 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

Überblick (DB2)



Erzeugung

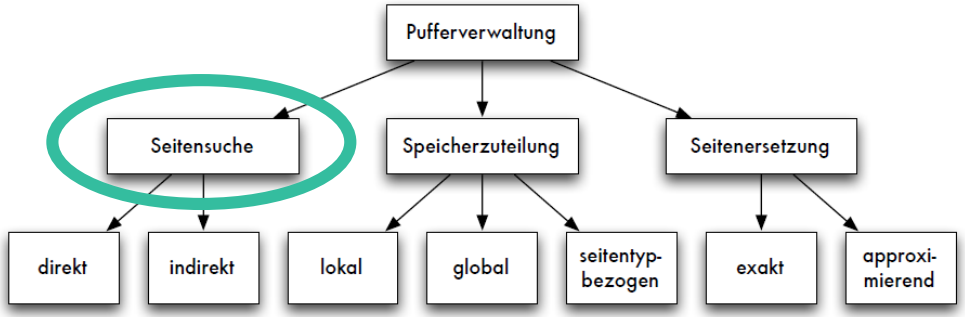
CREATE BUFFERPOOL bpname IMMEDIATE SIZE sz PAGESIZE pgsz
CREATE BUFFERPOOL bpname DEFERRED SIZE sz PAGESIZE pgsz

- immediate: Puffer wird sofort erzeugt, falls ausreichend Speicher zur Verfügung
- deferred: Aktivierung beim nächsten Neustart

Puffergröße verändern

ALTER BUFFERPOOL bpname IMMEDIATE/DEFERRED SIZE sz

Suche im DB-Puffer



Suche im DB-Puffer

Dresden Database Systems Group

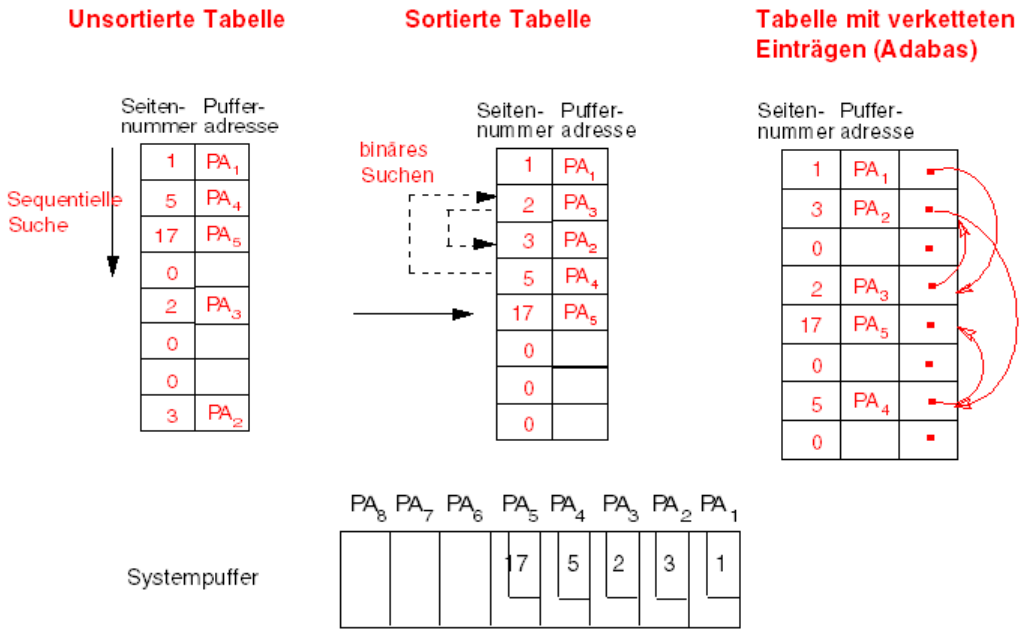
Forderung

- 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 Überlaufketten

Tabellenbasierte Suchstrategien

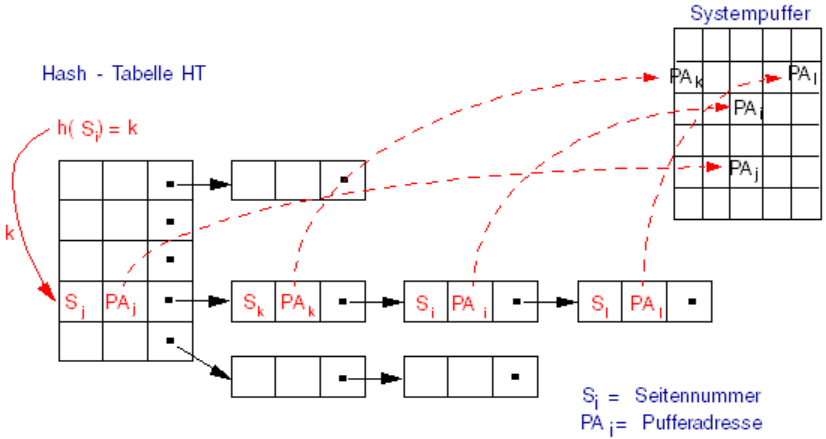


Hashverfahren zur Suche im DB-Puffer

Dresden Database Systems Group

Prinzip

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



Bewertung Suchstrategien

Dresden Database Systems Group

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

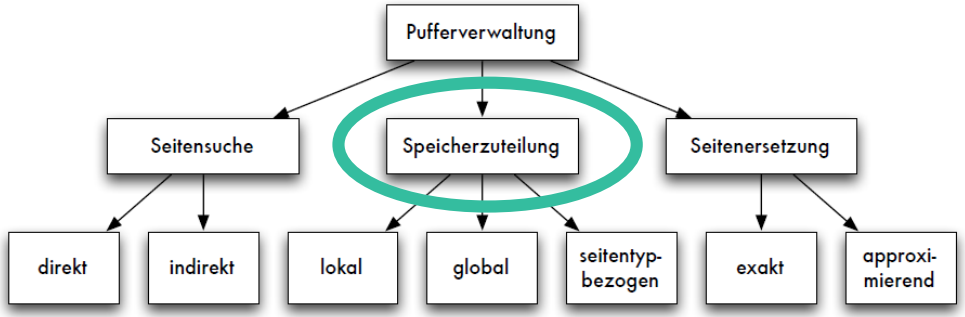
Tabelle mit verketteten Einträgen

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

Hashverfahren

- beste Lösung

Speicherzuteilung

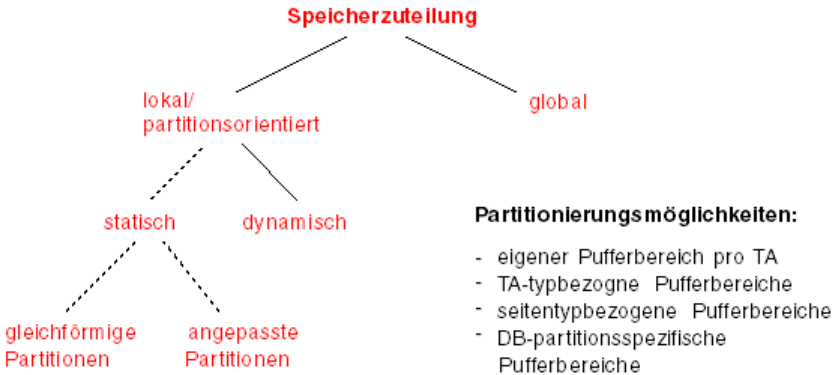


Speicherzuteilung

Dresden Database Systems Group

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



Speicherzuteilung im Puffer

Dresden Database Systems Group

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

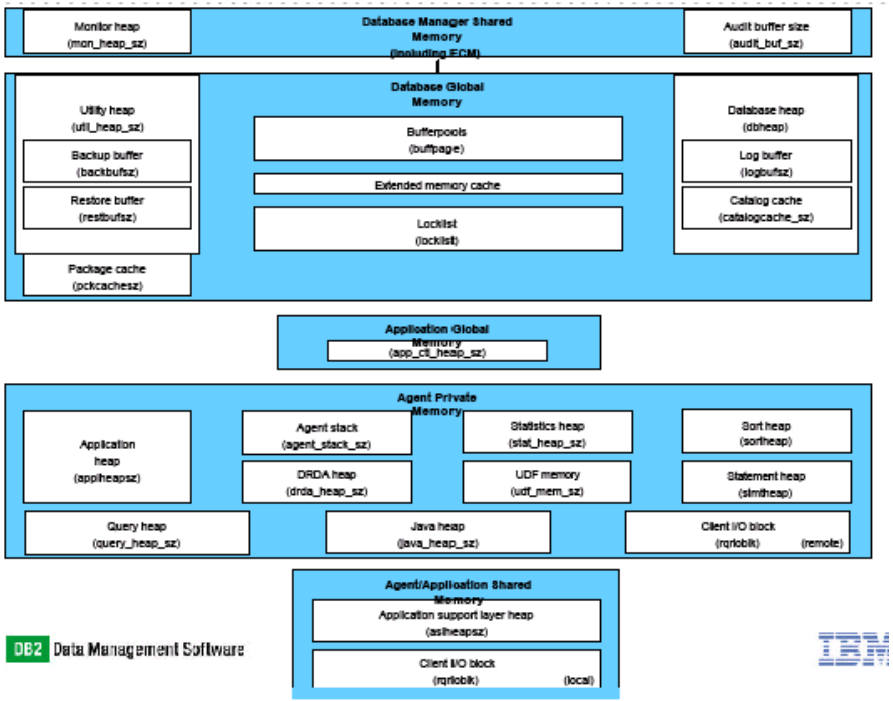
Statische Speicherzuteilung

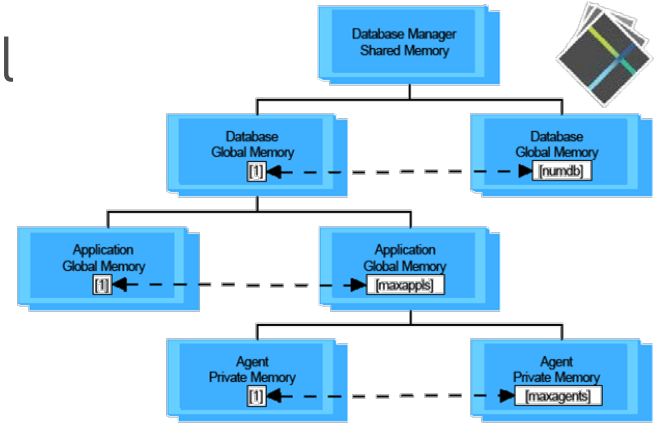
- eine erforderliche Menge von Rahmen (Partition) muss verfügbar sein, bevor die Transaktion gestartet wird (preclaiming). **Uninteressant in DBS**.

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!)

Example: DB2 Memory Model





Dresden Database Systems Group

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

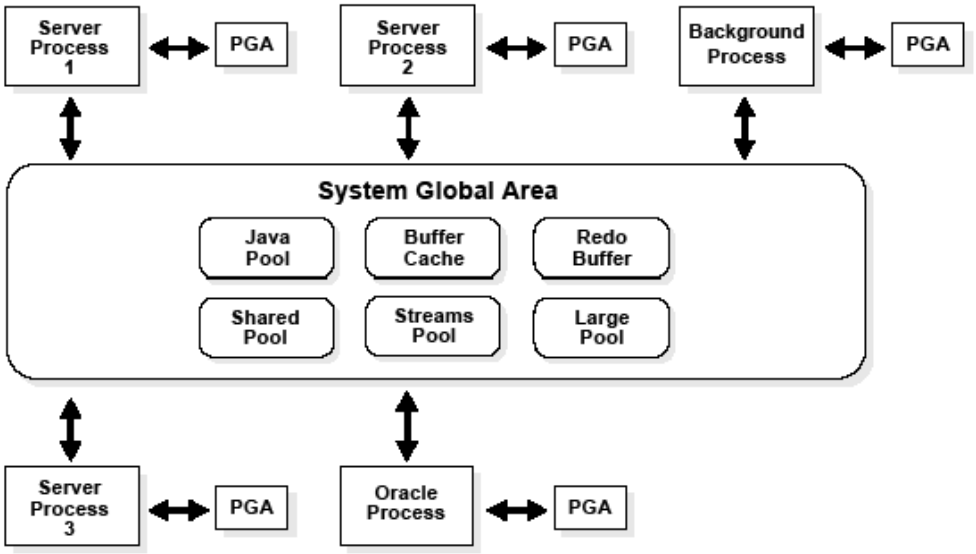
Application Shared Memory Set

- 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

Example: Memory-Management (Oracle)



PGA (Program Global Area): individuell für jeden Datenbankprozess

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

Lokalität

Dresden Database Systems Group

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

-> notwendig für partitionsorientierte Speicherzuteilung!

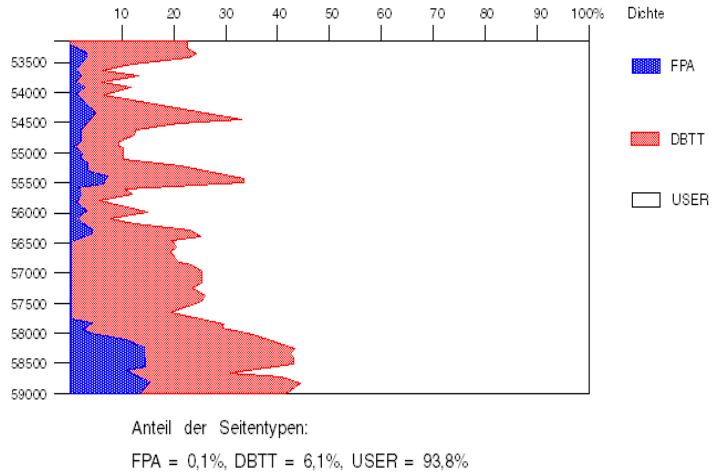
Beispiel zu Lokalität

Dresden Database Systems Group

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

Working-Set-Modell

Dresden Database Systems Group

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)

Feststellung

- Je kleiner W(t, τ) desto größer ist die Lokalität der Transaktion zum Zeitpunkt t

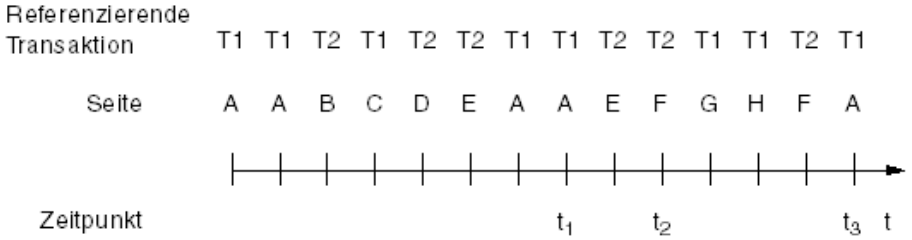
Ziel

- Optimale Zuteilung von Seitenreferenzen zu allen konkurrierenden Transaktionen
- Jede Transaktion kann effizient bearbeitet werden, wenn sie W(t, τ) Pufferrahmen erhält

Working-Set-Modell (2)

Dresden Database Systems Group

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 $\tau = 5$:

Dynamische Zuordnung der Pufferrahmen

Dresden Database Systems Group

Grundsätze

- Die Working-Set-Strategie stellt jeder TA zum Zeitpunkt t w(t, τ) Pufferrahmen zur Verfügung
- Verändert sich die Working-Set-Größe einer TA (Lokalität wächst oder schrumpft), dann wird die Anzahl der Pufferrahmen abgepasst (verringert: immer; vergrößert: wenn Seitenrahmen frei)
- Die Wahl der Fenstergröße τ ist wichtig für den Erfolg des Verfahrens
- Working-Set-Model beinhaltet nicht per se ein Verfahren, dass die Verdrängung der Seiten realisiert. Es muss um ein solches immer ergänzt werden

Umsetzung

- Bei jeder Fehlseitenbedingung muss zu jeder TA der aktuelle Working Set ermittelt werden. Dazu muss festgestellt werden, welche Seiten eine TA innerhalb der letzten τ Referenzen angesprochen hat

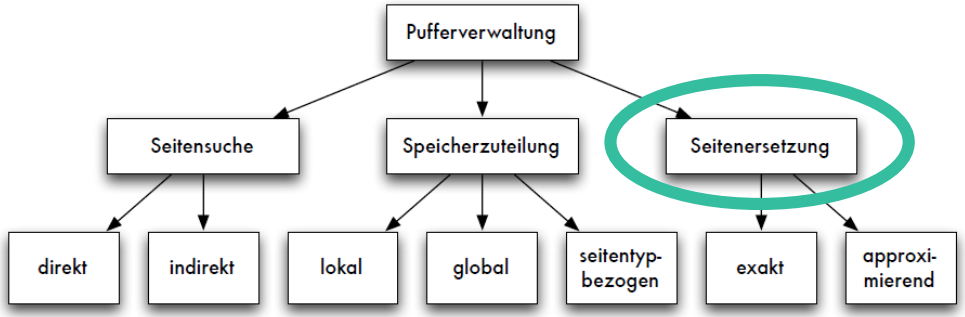
Implementierung

- Für jede Transaktion T wird ein transaktionsbezogener Referenzzähler TRZ(T) geführt
- Für jede Seite i und jede Transaktion T wird der letzte Referenzzeitpunkt LRZ (T, i) geführt
- Bei einer logischen Referenz werden folgende Anweisungen ausgeführt:

  1) TRZ (T) = TRZ (T)+1;

  2) LRZ (T, i) = TRZ(T);
- alle Seiten sind ersetzbar, für die gilt: TRZ (T) – LRZ(T. i) >= τ

Seitenersetzung im DB-Puffer



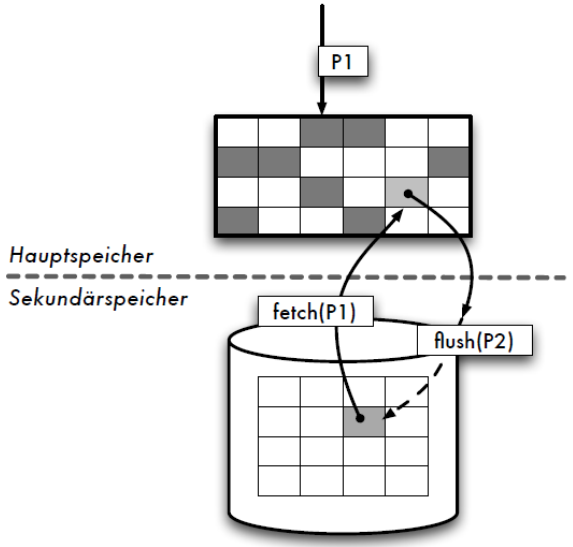
Seitenersetzungsstrategien

Dresden Database Systems Group

Überblick

- Speichersystem fordert Seite P1 an, die nicht im Puffer vorhanden ist
- Sämtliche Pufferrahmen sind belegt
- Vor dem Laden von P1 Pufferrahmen freimachen
- Nach den unter beschriebenen Strategien Seite aussuchen
- Ist Seite in der Zwischenzeit im Puffer verändert worden, so wird sie nun auf die Platte zurückgeschrieben
- Ist Seite seit Einlagerung in den Puffer nur gelesen worden, so kann sie überschrieben werden (verdrängt)

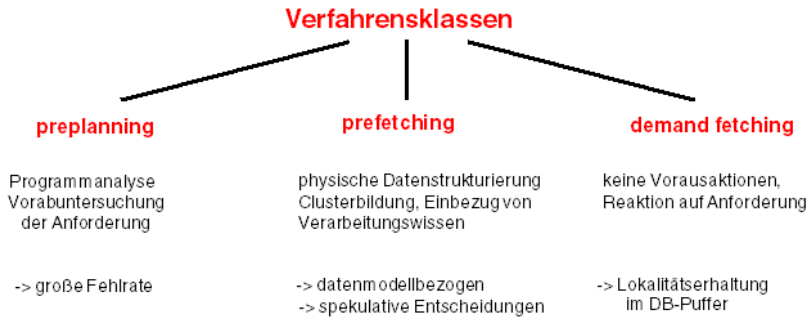
Seitenersetzung schematisch



Seitenersetzung in DBMS

Dresden Database Systems Group

Möglichkeiten



Fixieren von Seiten (Pin oder FIX)

- Fixieren von Seiten im Puffer verhindert verdrängen
- Speziell für Seiten, die in Kürze wieder benötigt werden

Freigeben von Seiten (Unpin oder Unfix)

- Freigeben zum Verdrängen
- Speziell für Seiten, die nicht mehr benötigt werden

Zurückschreiben einer Seite

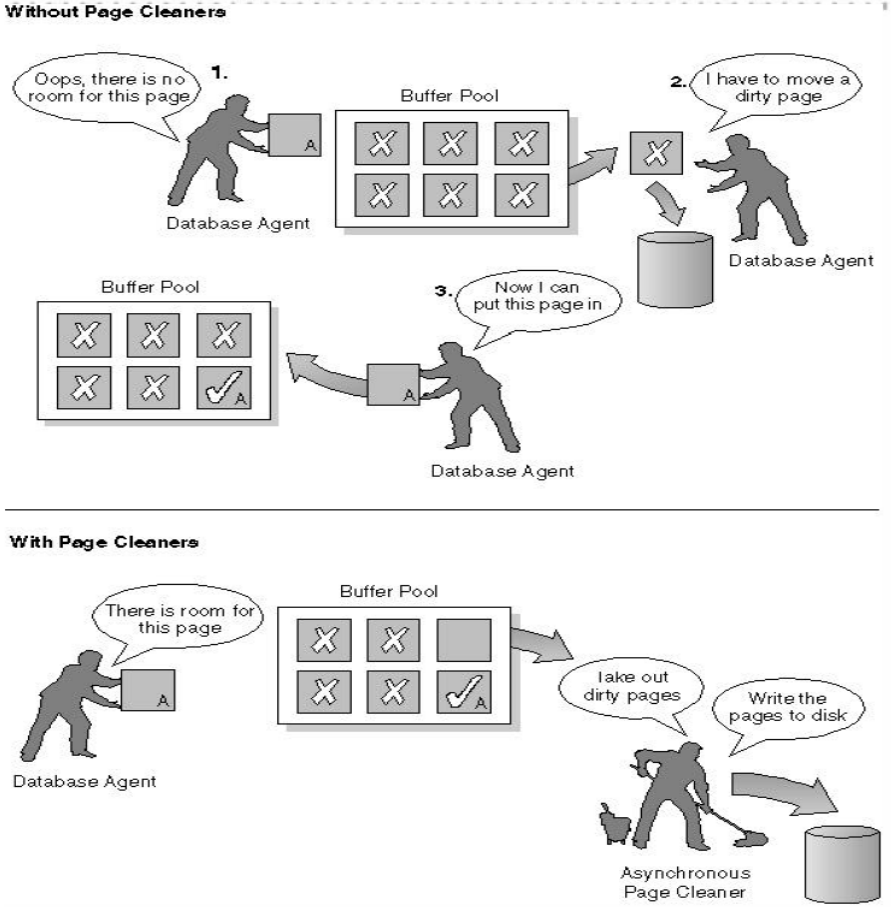
- Auslösen des Zurückschreibens für geänderte Seiten bei Transaktionsende

Seitenersetzung in DBMS: I/O-Cleaners

Dresden Database Systems Group

Idee

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



Seitenersetzung: Verfahren

Dresden Database Systems Group

Grundsätzliches Vorgehen beim Laden einer Seite

- Demand-paging-Verfahren: genau eine Seite im Puffer durch angeforderte Seite ersetzen
- Prepaging-Verfahren: neben der angeforderten Seite auch weitere Seiten in den Puffer einlesen, die eventuell in der Zukunft benötigt werden

Ersetzen einer Seite im Puffer

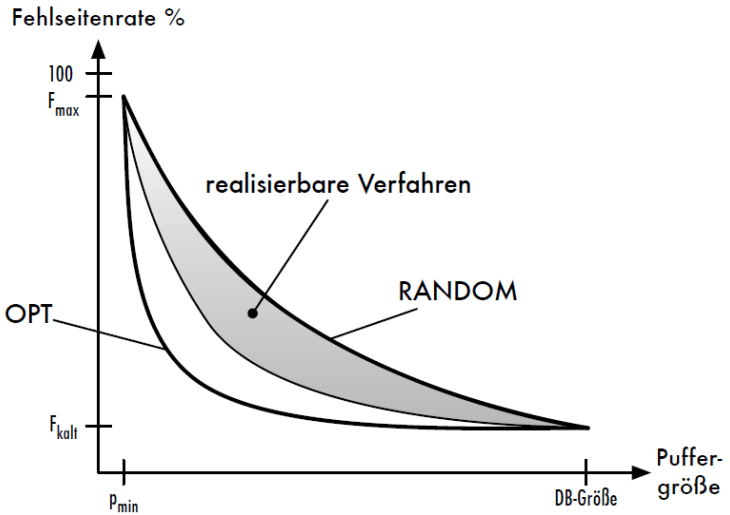
- Optimale Strategie: Welche Seite hat maximale Distanz zu ihrem Gebrauch? (nicht realisierbar, zukünftiges Referenzverhalten nicht vorhersehbar)
  → realisierbare Verfahren besitzen keine Kenntnisse über das zukünftige Referenzverhalten

- Zufallsstrategie: jede Seite gleiche Wiederbenutzungswahrscheinlichkeit zuordnen

Seitenersetzung: Verfahren/2

Dresden Database Systems Group

Eingrenzung



Fehlseitenrate

$$F = 1 - p(\frac{1 - F_{kalt}}{p_{DB}}) \cdot 100\%$$

- P: Puffergröße
- p_{DB}: Puffergröße, die gesamte Datenbank umfasst
- F_{kalt}: Fehlseitenrate beim Kaltstart (d.h. leerer Puffer)
  → Verhältnis von Anzahl der in den Puffer geladenen Seiten zur Anzahl der Referenzierungn

Seitenersetzung: Verfahren/3

Dresden Database Systems Group

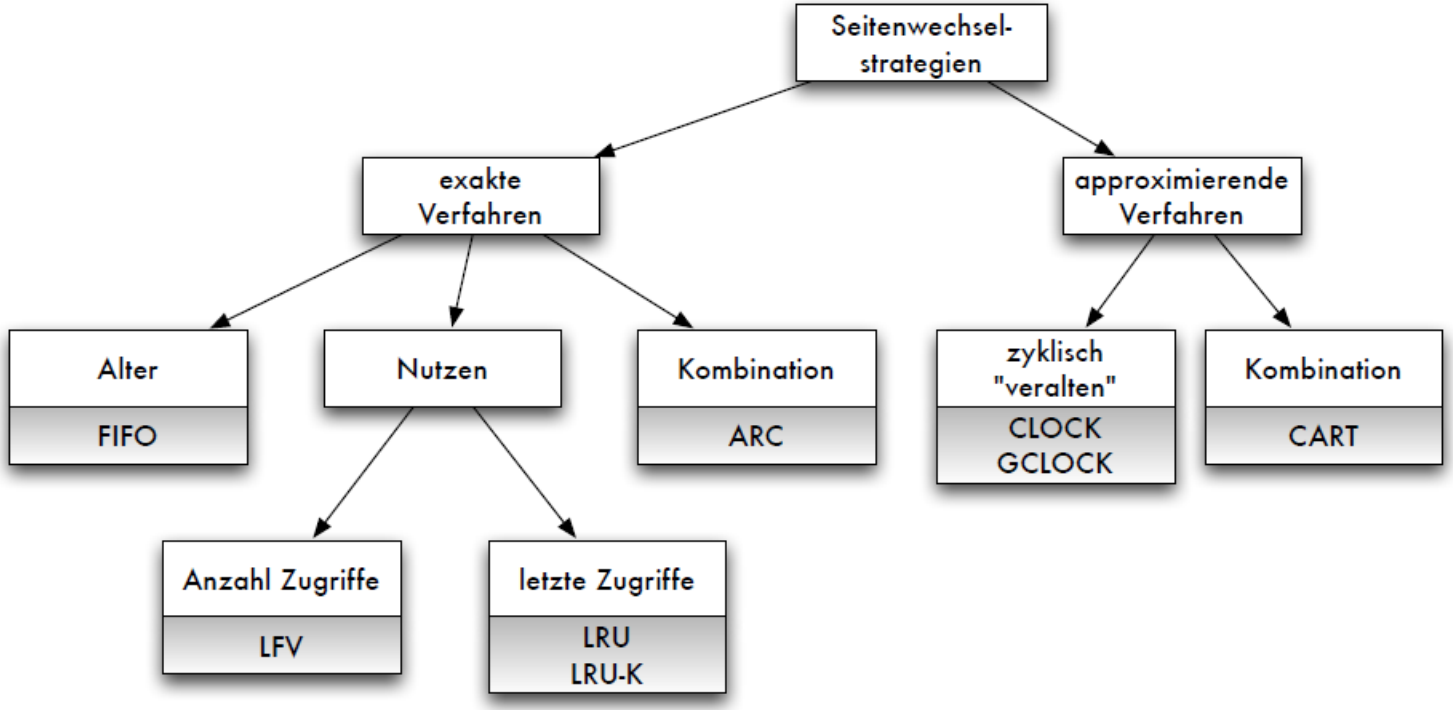
Schlussfolgerung

- Gute, realisierbare Verfahren sollen vergangenes Referenzverhalten auf Seiten nutzen, um Erwartungswerte für Wiederbenutzung schätzen zu können
- Besser als Zufallsstrategie
- Annäherung an optimale Strategie

Merkmale gängiger Strategien

- Alter einer Seite im Puffer
  - Alter einer Seite nach Einlagerung (globale Strategie G)
  - Alter einer Seite nach dem letzten Referenzzeitpunkt (die Strategie des jüngsten Verhaltens – J)
  - Alter einer Seite wird nicht berücksichtigt (-)
- Anzahl der Referenzen auf Seite im Puffer
  - Anzahl aller Referenzen auf einer Seite (globale Strategie – G)
  - Anzahl nur der letzten Referenzen auf eine Seite (Strategie des jüngsten Verhaltens – J)
  - Anzahl der Referenzen wird nicht berücksichtigt (-)

Gängige Strategien



Klassifikation gängiger Strategien

Dresden Database Systems Group

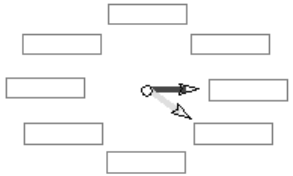
Verfahren	Prinzip	Alter	Anzahl
FIFO	älteste Seite ersetzt	G	–
LFU (least frequently used)	Seite mit geringster Häufigkeit ersetzen	–	G
LRU (least recently used)	Seite ersetzen, die am längsten nicht referenziert wurde (System R)	J	J
DGCLOCK (dyn. generalized clock)	Protokollierung der Ersetzungshäufigkeiten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

FIFO (First In First Out)

Dresden Database Systems Group

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)

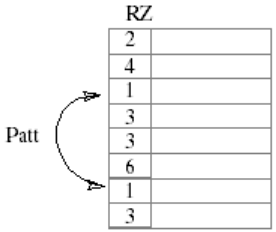
Erweiterung: CLOCK, GCLOCK und DGCLOCK

LFU (Least-Frequently-Used)

Dresden Database Systems Group

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

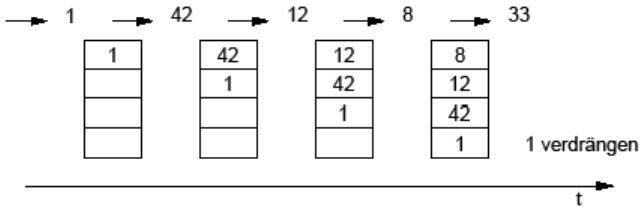
- Seiten, die punktuell sehr intensiv benutzt werden und dann nicht mehr, sind praktisch nicht zu verdrängen

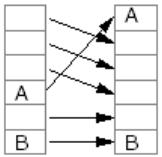
LRU (Least-Recently-Used)

Dresden Database Systems Group

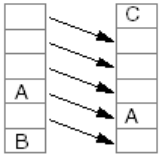
Ersetzung der Seite, die am längsten nicht mehr referenziert wurde

- 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.



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

Merkmale

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

LRU-Stacktiefenverteilung

Dresden Database Systems Group

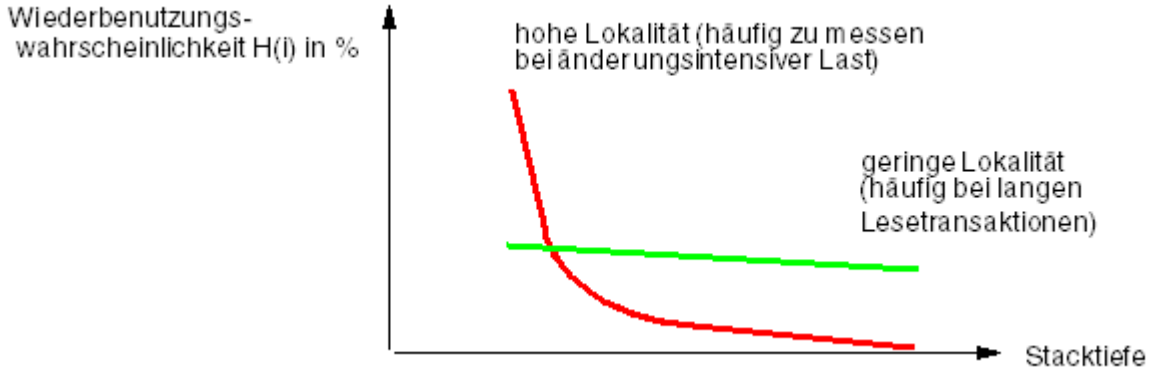
Prinzip

- 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



LRU: Probleme

Dresden Database Systems Group

Lock Contention in Multitasking-Umgebungen

- Zugriff auf LRU-Liste/Stack und Bewegung der Seite erfordert exklusiven Zugriff auf Datenstruktur
- Aufwendige Operation

Berücksichtigt nur Alter jedoch nicht Häufigkeit

- Oft gelesene Seiten mit langen Pausen zwischen den Zugriffen werden nicht adäquat berücksichtigt

Zerstörung des Puffers durch Scan-Operator

- Seiten werden nur einmalig gelesen, verdrängen jedoch andere (ältere) Seiten
- FIFO-Prinzip

LRU-K

Dresden Database Systems Group

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 $r_1, r_2, \ldots, r_t$
  - $b_t(p,K)$ 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

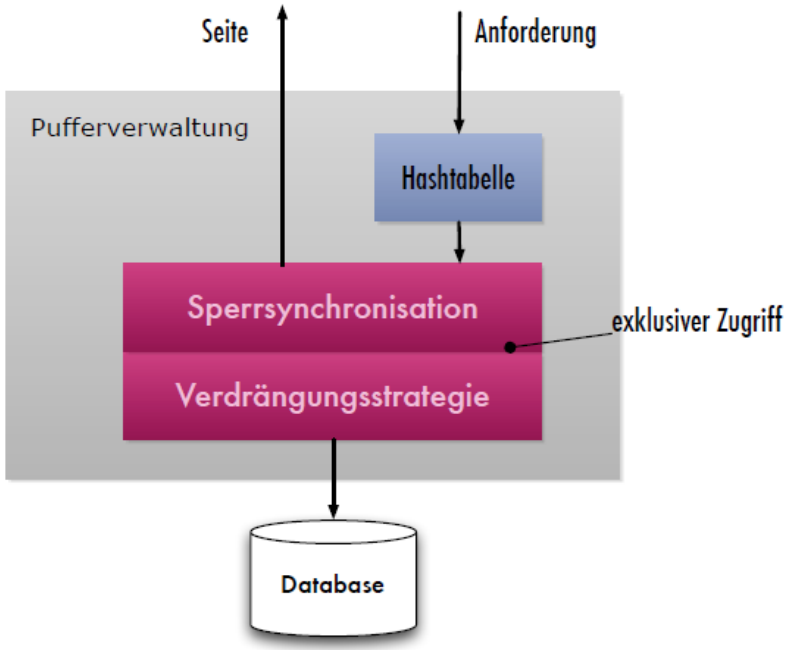
- berücksichtigt aktuelle Referenzierungen häufiger als ältere
- LRU-1 entspricht LRU
- typisch: LRU-2

Lock-Contention bei Pufferverwaltung

Dresden Database Systems Group

LATCHES

- Leichtgewichtige Sperren (wenige CPU-Instruktionen) für kurzzeitiges Sperren



LRUN versus LRR

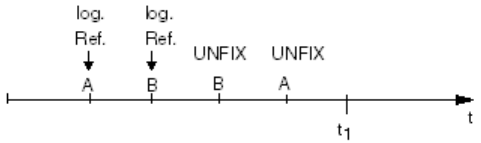
Dresden Database Systems Group

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

Fazit

Dresden Database Systems Group

Fazit

- Pufferverwaltungsstrategie mit großem Einfluss auf Performance
- in kommerziellen Systemen meist LRU mit Variationen
- besondere Behandlung von Full-Table-Scans
- weiterer Einflussfaktor: **Puffergröße**

5-Minuten-Regel (Gray, Putzolu, 1997)

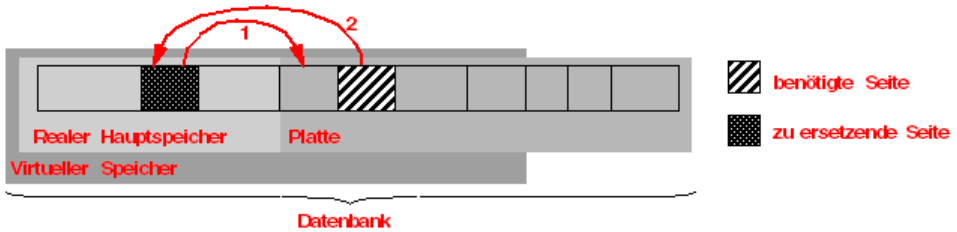
Daten, die in den nächsten 5 Min. wieder referenziert werden, sollten im Hauptspeicher gehalten werden

Problem: Virtueller Adressraum

Dresden Database Systems Group

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.

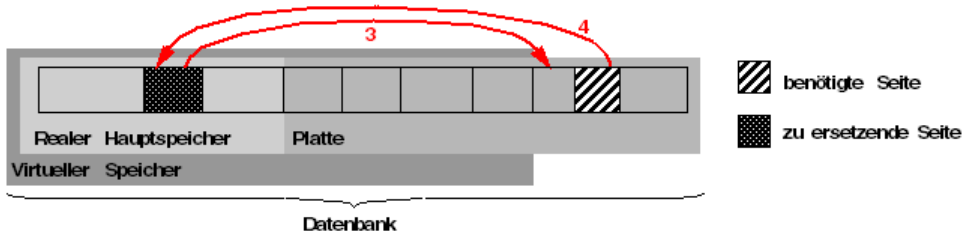


Problem: Virtueller Adressraum (2)

Dresden Database Systems Group

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.

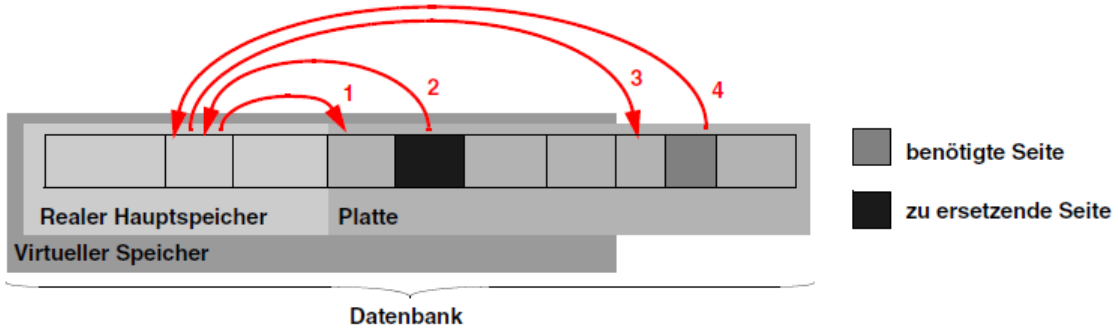


Problem: Virtueller Adressraum (3)

Dresden Database Systems Group

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.



Zusammenfassung

Dresden Database Systems Group

Eigenschaften eines DB-Systempuffers

- Nutzung von Kontextwissen zur Ermittlung der Ersetzungskandidaten

Algorithmen

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

Ersetzungsstrategien

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