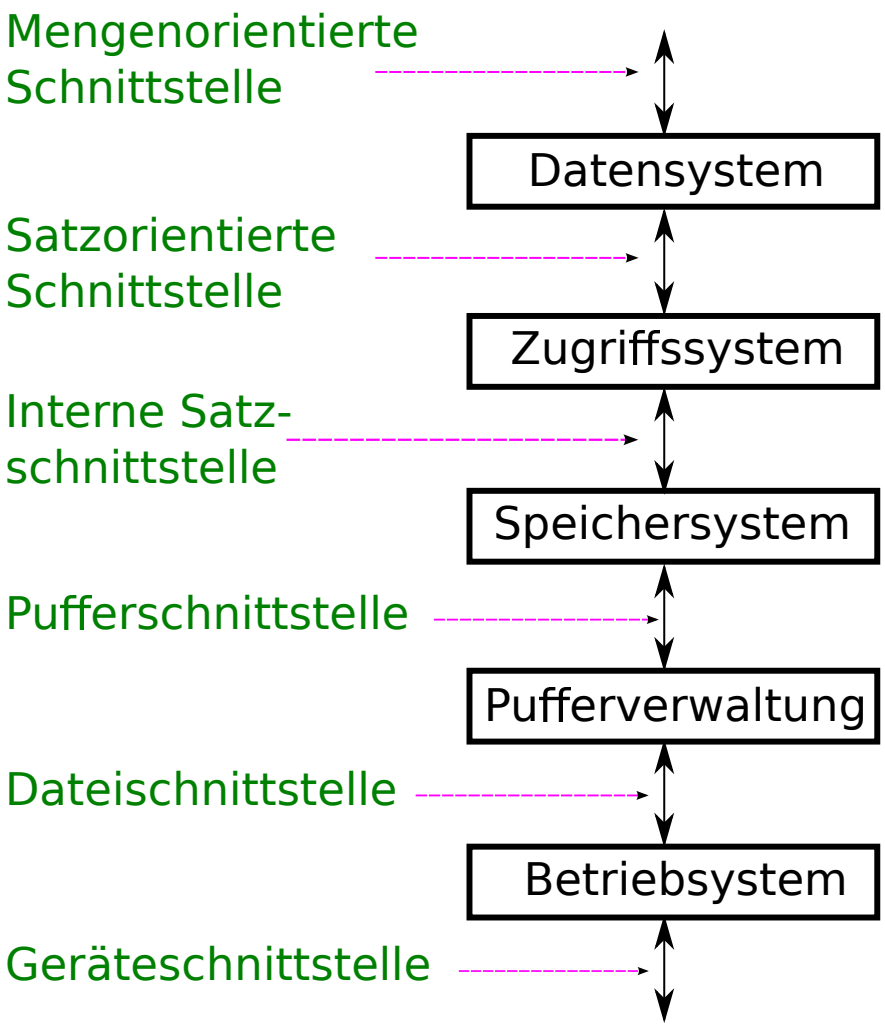
Datenbanksysteme

Wintersemester 2016/17

Prof. Dr.-Ing. Sebastian Michel
TU Kaiserslautern

smichel@cs.uni-kl.de

5 Schichtenmodell

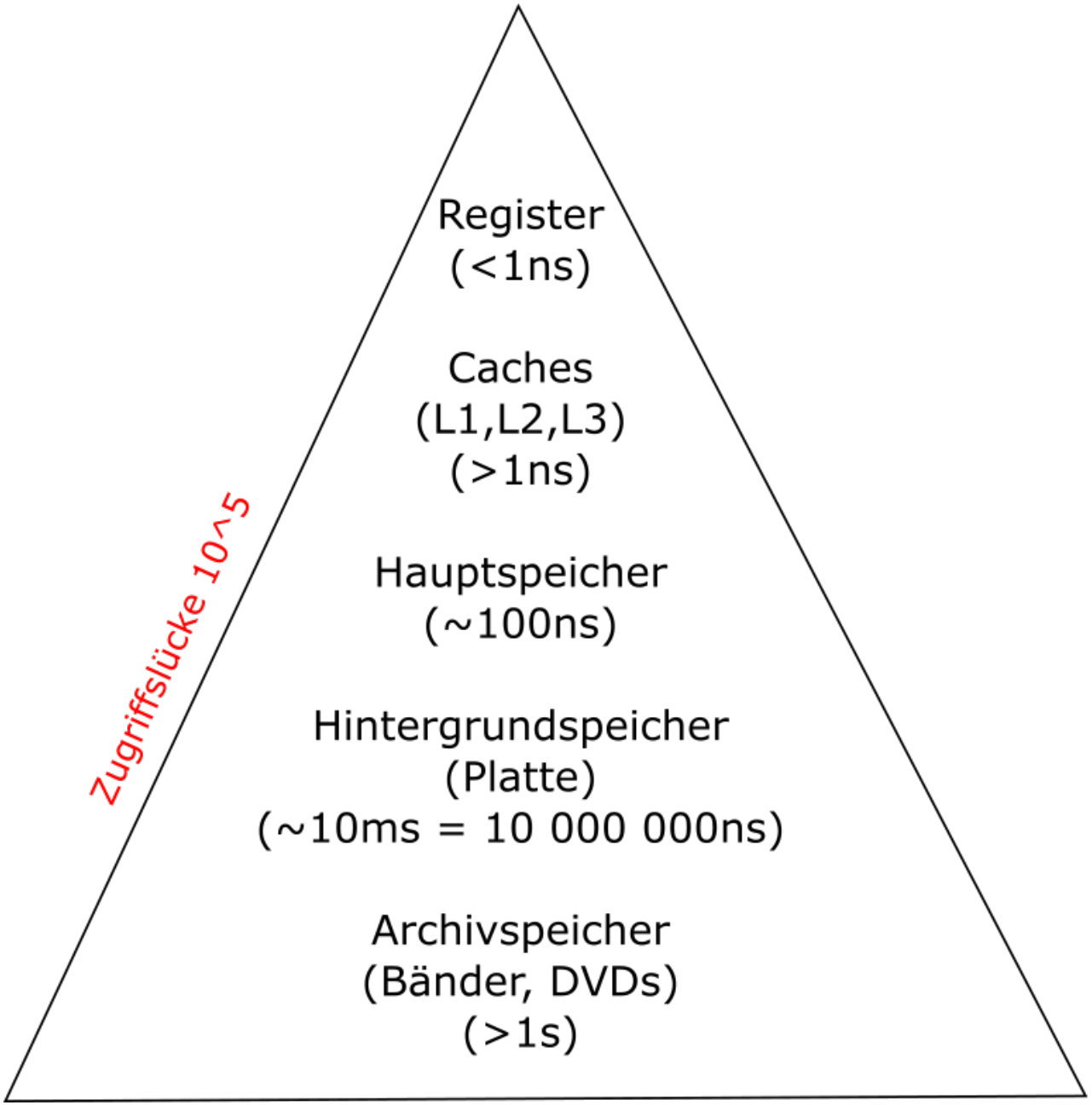


Datensystem	Relationen, Sichten	SQL: select .... from ...
Zugriffssystem	ext. Sätze, Scans, Indexstrukturen	FIND NEXT satz STORE satz
Speichersystem	int. Sätze, Bäume, Hashtabellen	INSERT in B Baum
Pufferverwaltung	Segmente, Seiten	Bereitstellen Seite j Freigeben Seite j
Betriebsystem	Dateien, Blöcke	Lies Block k Schreibe Block k
	Zylinder Spuren	Treiber

Was kostet wieviel?

- L1 cache reference 0,5 ns
- L2 cache reference 7 ns
- Main memory reference 100 ns
- Compress 1K bytes with Zippy 10 000 ns
- Send 2K bytes over 1 Gbps network 20 000 ns
- Read 1 MB sequentially from memory 250 000 ns
- Round trip within same datacenter 500 000 ns
- Disk seek 10 000 000 ns
- Read 1 MB sequentially from network 10,000,000 ns
- Read 1 MB sequentially from disk 30 000 000 ns
- Send packet CA $\rightarrow$ Netherlands $\rightarrow$ CA 150 000 000 ns

Numbers by Jeff Dean (Google)



Analogie

Kopf
(1 Min)

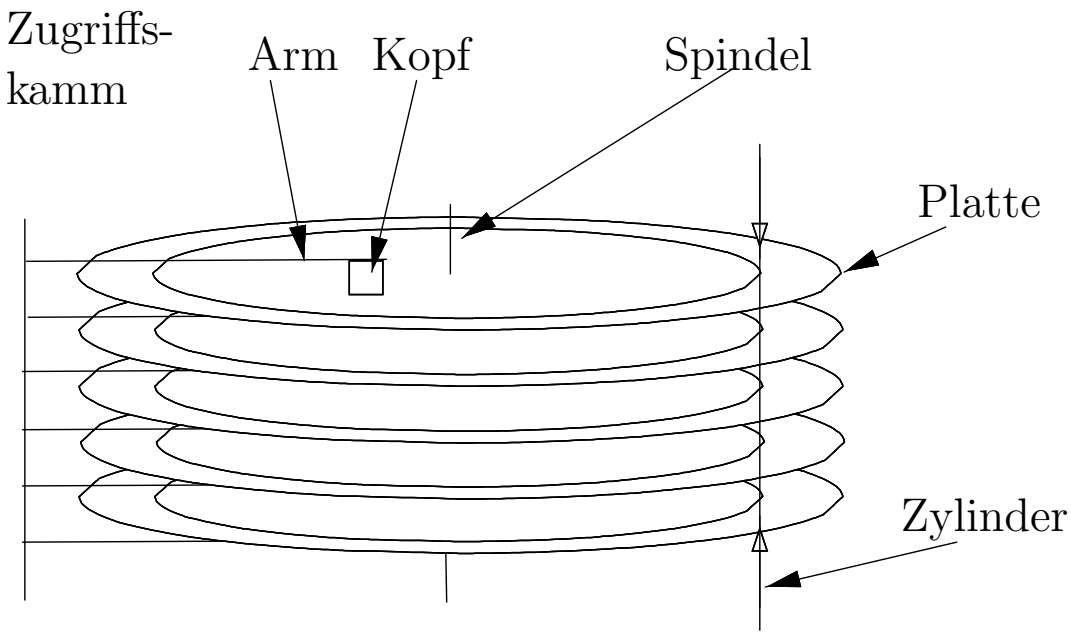
Raum
(10 Min)

Stadt
(1 Std)

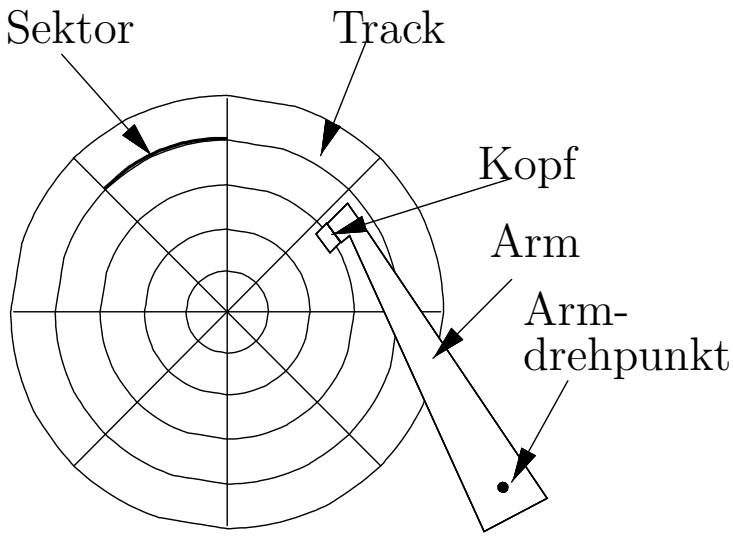
Pluto
(2 Jahre)

Andromeda
(2000 Jahre)

Aufbau einer (klassischen) Festplatte



a. Seitenansicht



b. Draufsicht

Aufbau Festplatte (Tracks, Sektoren, Zonen)

- **Track:** Teil eines Zylinders auf einer Platte
- **Sektor:** Teil eines Tracks. Anzahl Sektoren pro Track ursprünglich gleich für alle Tracks
- Aber, auf Zylinder/Tracks weiter "außen" passen mehr Sektoren. Daher, aktuelle Hardware, variable Anzahl von Tracks: äußere Tracks haben mehr Sektoren als innere Tracks; Zylinder sind in Zonen unterteilt.

Blöcke

- Aka. Sektoren, Aka. physical Record. Kleinste Transfereinheit bei Block-Storage-Devices. Seit wenigen Jahren typischerweise 4KB oder sonst (historisch) 512 Byte groß.
- **Achtung**, es gibt auch Unterschiede zu Blöcken bzw. Seiten des Dateisystems, dort kann ein Block auch mehrere Festplattenblöcke umfassen.

Beispiel

Die Festplatte SAMSUNG HD103SJ, die in meinem Desktop PC im Büro eingebaut ist hat laut (Linux tool) `hdparm -i` (oder `hdparm -I` für mehr Details):

- 1953525168 Sektoren
- a 512 Bytes.

Das macht: 1 TB

Ausgabe (Auszug) von hdparm -I /dev/sda

```
CHS current addressable sectors:   16514064
LBA    user addressable sectors:  268435455
LBA48  user addressable sectors: 1953525168
Logical  Sector size:                   512 bytes
Physical Sector size:                   512 bytes
device size with M = 1024*1024:      953869 MBytes
device size with M = 1000*1000:     1000204 MBytes (1000 GB)
```

Physische Organisation einer Datenbank

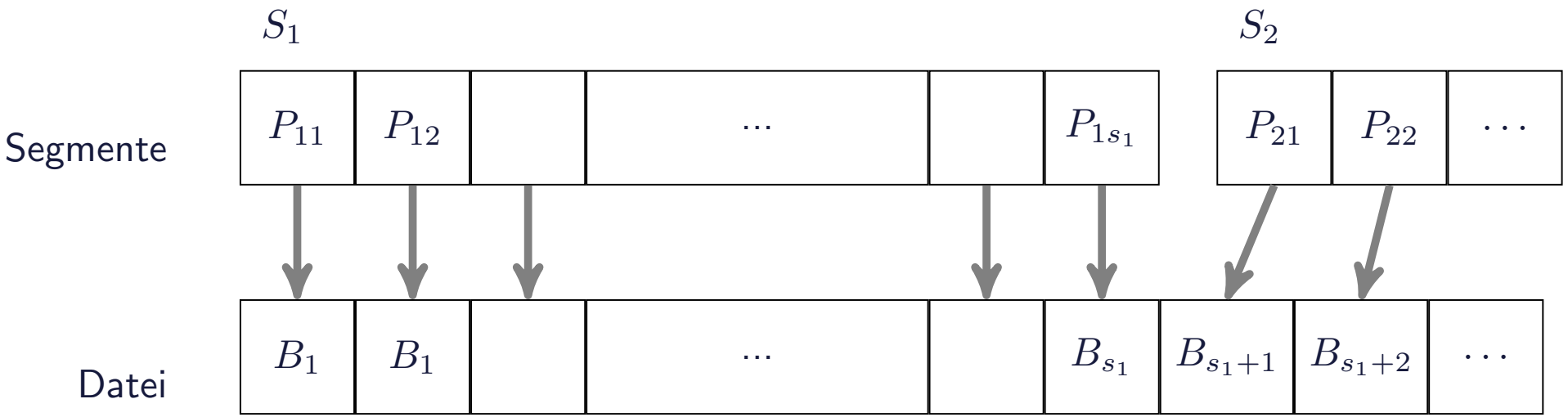
Das Datenbanksystem organisiert den physikalischen Speicher in verschiedene Schichten.

- **Datenbank**: Menge von Dateien
- **Datei**: Sequenz von Blöcken.
- **Segmente**: Organisationseinheit im DBMS (bzgl. Sperren, Rechten, etc.)

Zugriffseinheiten

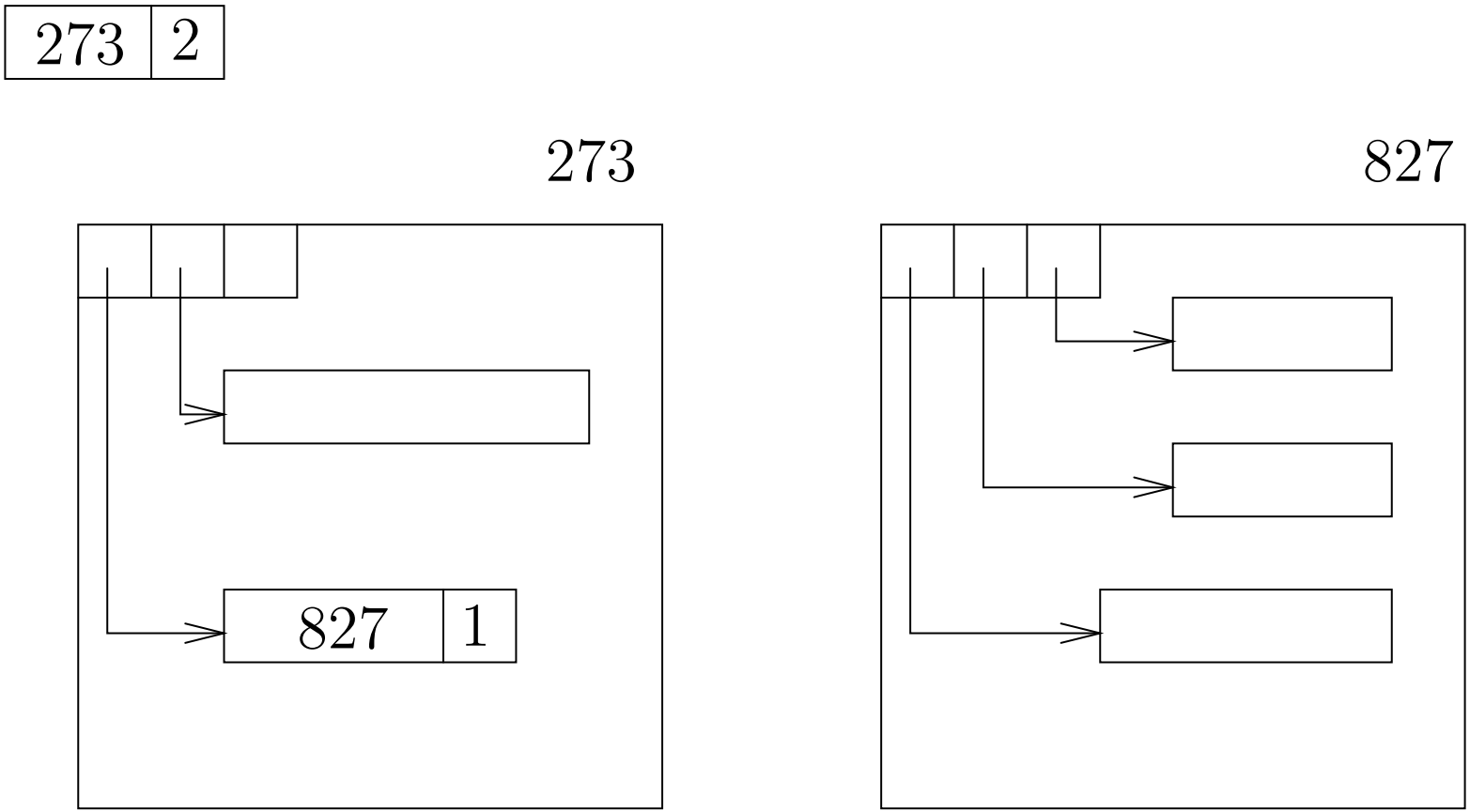
- **Segmente**
- **Seiten** werden in Segmenten gespeichert
- **Seite** enthält Sätze
- **Satz**: In einer Seite gespeicherte Sequenz von Bytes. Menge von echten Daten, verschiedene Felder.
- Bzw. man redet auch von **Tupeln** im DB (Relationen) Kontext

Seitenabbildung: Direkte Seitenadressierung



- Direkte Zuordnung zwischen Seiten eines Segments S und Blöcken einer Datei D.
- Seite P_i mit $1 \leq i \leq s$ wird in Block B_j $(1 \leq j \leq d)$ gespeichert, so dass $j = K - 1 + i$ und $d \geq K - 1 + s$.
- K bezeichnet die Nummer des ersten für S reservierten Blocks.
- **In der Regel:** 1:1 Zuordnung, d.h. $K = 1$ und $s = d$.

Seite



- Seite (Page) ist organisiert in Anzahl von Bereiche (Slots)
- Slots zeigen auf Daten
- ... oder auch auf andere Seiten

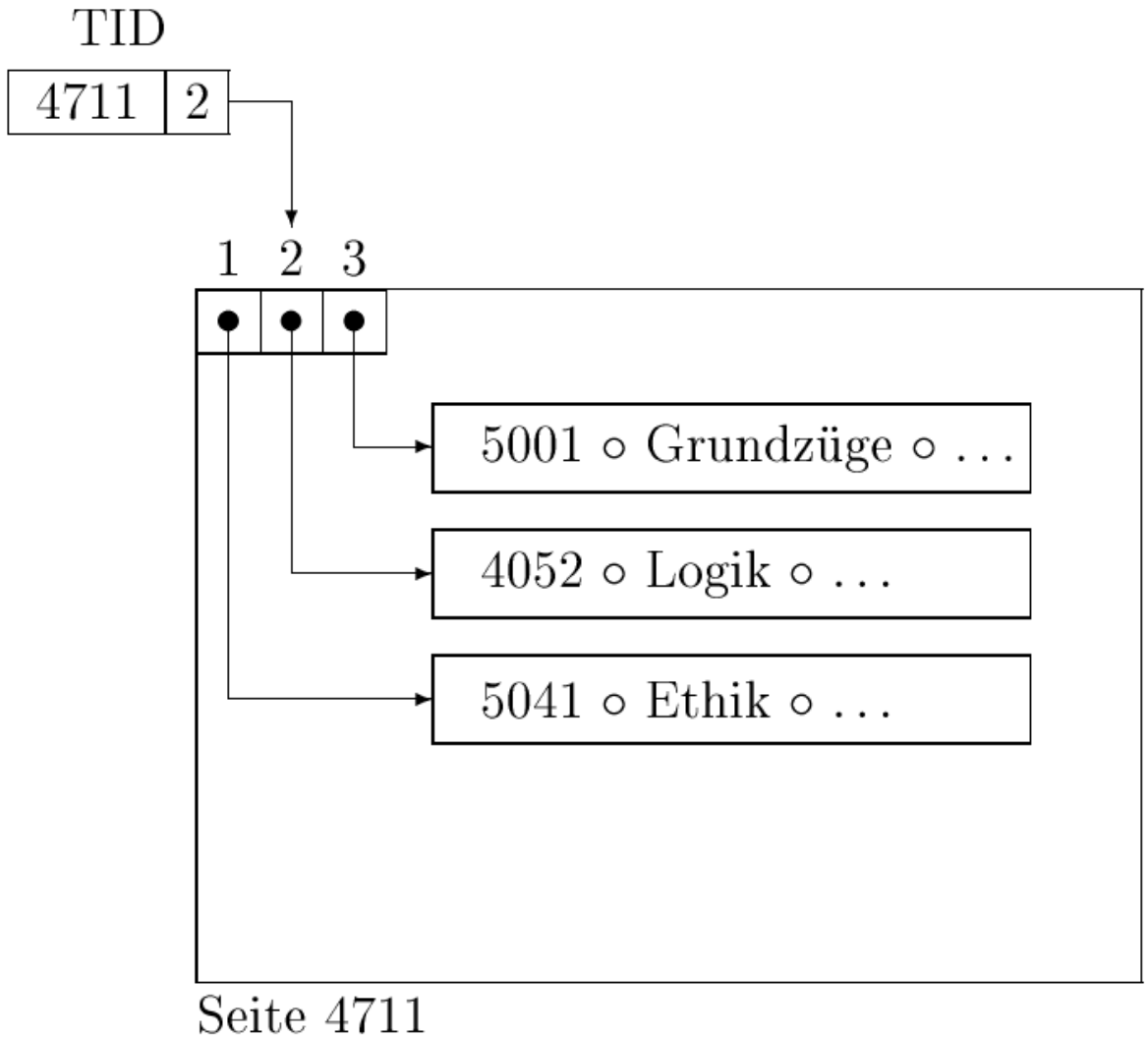
Tuple Identifier (TID)

Ein TID ist eine Paar bestehend aus

- **Seiten ID** (z.B. Datei/Segment Nummer plus Nummer der Seite)
- **Slot Nummer**

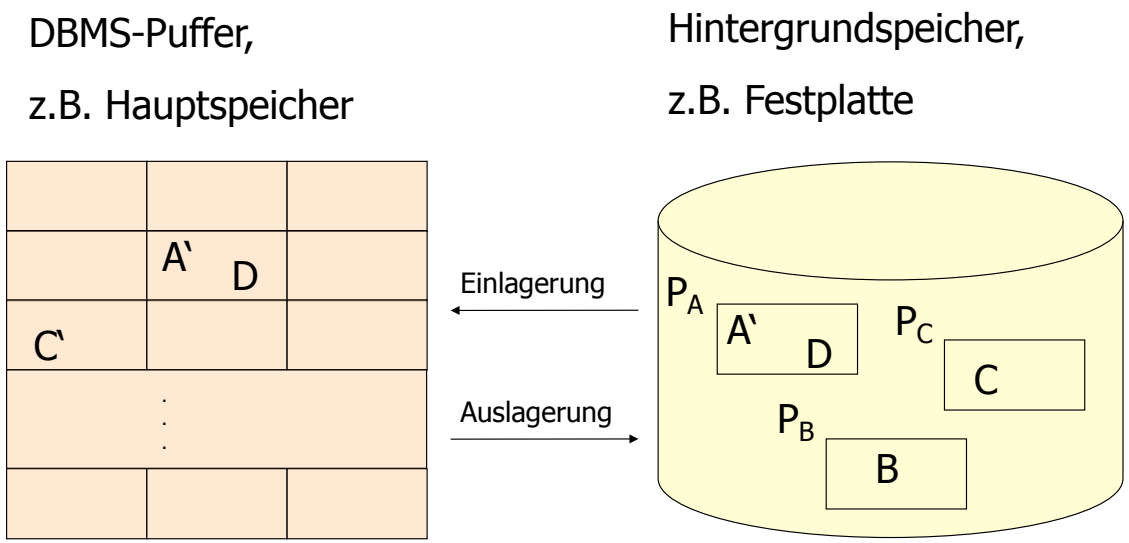
TID wird manchmal auch Row Identifier (RID) genannt

Speicherung von Tupeln auf Seiten



Datenbank-Pufferverwaltung

- **Motivation**: Bereits erwähnte Zugriffslücke zwischen Hauptspeicher und Festplatte (Externspeicher)
- **Idee**: Halte Seiten, auf die zugegriffen wurde, in einem Puffer im Hauptspeicher
- Dieser Puffer kann durchaus sehr groß sein (hunderte Megabyte oder viele Gigabytes). Trotzdem viel kleiner als DB selbst.



Datenbank-Pufferverwaltung (2)

5 Minuten Regel

"Pages referenced every five minutes should be memory resident."

Siehe Papier von Jim Gray & Franco Putzolu:
`http://www.hpl.hp.com/techreports/tandem/TR-86.1.pdf`

Besonderheiten DB Puffer (vs. OS Puffer)

- **DB-spezifische Referenzmuster**
- z.B. sequentielle oder baumartige Zugriffsfolgen
- Gleiches gilt für **Prefetching**. Hier kann in DB aufgrund von Seiteninhalten oft eine Voraussage gemacht werden.

Empfehlung zum Thema Datenbank-Pufferverwaltung: Das Buch von Härder und Rahm: "Datenbanksysteme - Konzept und Techniken der Implementierung" ist hier sehr ausführlich.

Seitenreferenzstrings / Page Reference Strings (PRSs)

Jeder Datenzugriff ist eine logische Seitenreferenz.
Die Aufgabe des DB-Puffer-Managements ist die Minimierung von physischen Seitenreferenzen.
Im Folgenden werden **Seitenreferenzstrings** betrachtet:

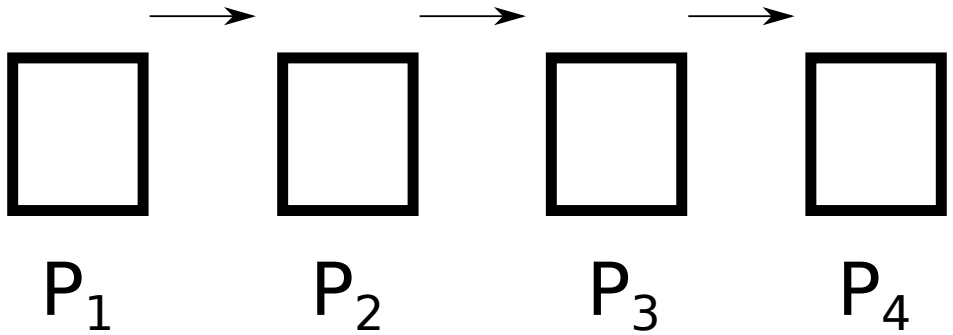
$$PRS = \langle r_1, r_2, \ldots, r_i, \ldots r_n \rangle \quad \text{mit} \quad r_i = (T_i, P_i)$$

- Transaktion T_i
- Referenzierte Seite P_i

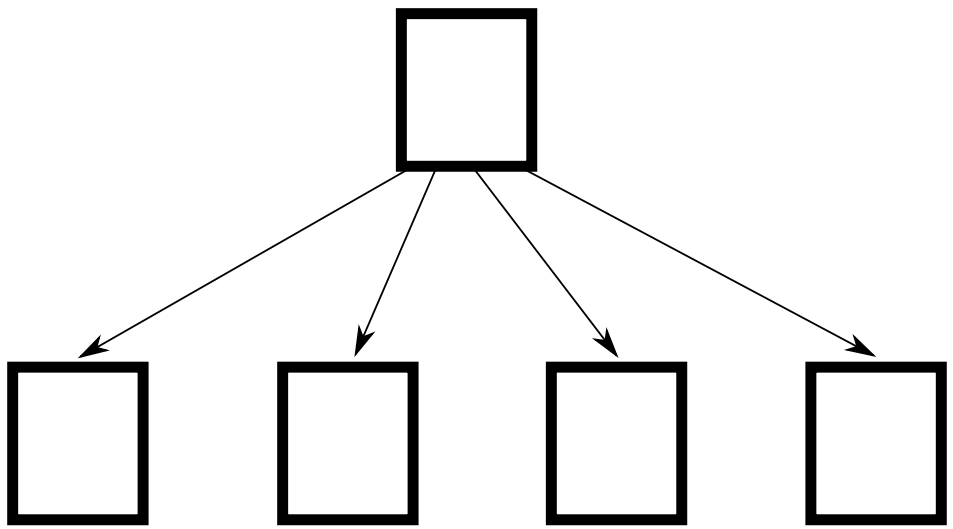
Die Analyse des Seitenreferenzstrings kann Aufschluss geben über **Zugriffs-Charakteristiken**, die zu **Optimierungen der Performance** führen können.

Typische Referenzmuster in Datenbanksystemen

1. Sequentielle Suche. Z.B. Scan einer kompletten Tabelle.



2. Hierarchische Zugriffspfade. Beispiel: Suche anhand eines B+ Baums.



Sequentialität

- PRs zeigen typischerweise **Phasen von Sequentialität und Lokalität**
- **Sequentielle Referenzsequenz (SRS)**: Zwei aufeinander folgende Referenzen r_i und r_{i+1} gehören zur gleiche sequentiellen Referenzsequenz falls

$$P_{i+1} - P_i = 0 \text{ oder } 1$$

  d.h. die Referenzen greifen auch benachbarte Seiten zu
- **Länge einer sequentiellen Referenzsequenz (LRS):**
  - LRS ist die Anzahl verschiedener Seiten die in SRS referenziert werden
  - Beispiel: A A B B D E E F F H enthält (AABB) mit LRS 2, (DEEFF) mit LRS 3 und (H) mit LRS 1.
- **Maß für Sequentialität**
  - Kumulative Verteilung der SRS Längen, d.h. LRSs

$$S(x) = P[\text{SRS Länge } \leq x]$$

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

Lokalität

Wie kann die **Lokalität** von Seitenreferenzen gemessen werden?

Working-Sets:

- $W(t,\tau)$ sei die Menge der Seiten, die von der betrachteten Transaktion innerhalb ihrer letzten τ Rerferenzen angesprochen wurden. Bezogen auf Zeitpunkt t.
- τ heißt Fenstergröße
- $w(t,\tau) = |W(t,\tau)|$ heißt **Working-Set-Größe**

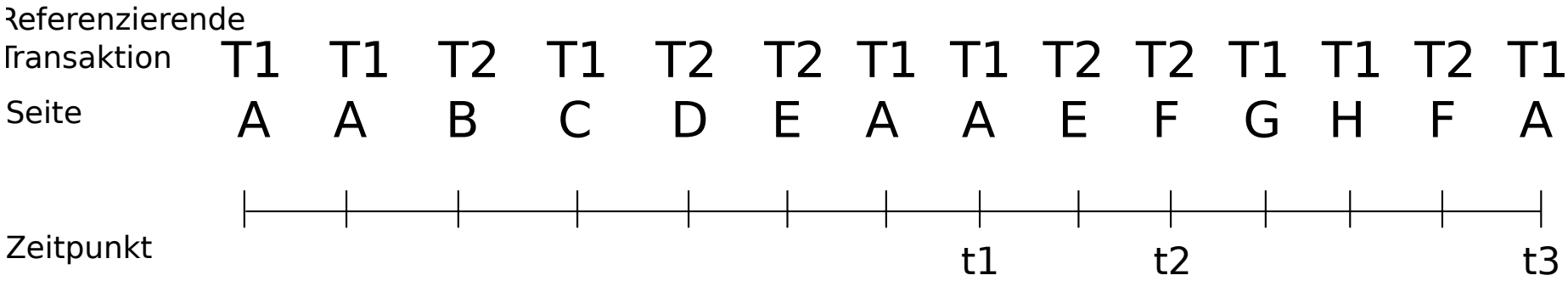
Dann haben wir zum Zeitpunkt t die Lokalität

$$AL(t,\tau) = \frac{w(t,\tau)}{\tau}$$

und die **durchschnittliche Lokalität** ist

$$L(\tau) = \frac{\sum_{t=1}^{n} AL(t,\tau)}{n}$$

Beispiel zu Working-Sets



T1: $W(t1,5) = \{A,C\}$ $w(t1,5) = |W(t1,5)| = 2$

T2: $W(t1,5) = \{B,D,E\}$, $w(t1,5) = |W(t1,5)| = 3$

T1: $W(t2,5) = \{A,C\}$ $w(t2,5) = |W(t2,5)| = 2$

T2: $W(t2,5) = \{B,D,E,F\}$ $w(t2,5) = |W(t2,5)| = 4$

T1: $W(t3,5) = \{A,G,H\}$ $w(t3,5) = |W(t3,5)| = 3$

T2: $W(t3,5) = \{D,E,F\}$ $w(t3,5) = |W(t3,5)| = 3$

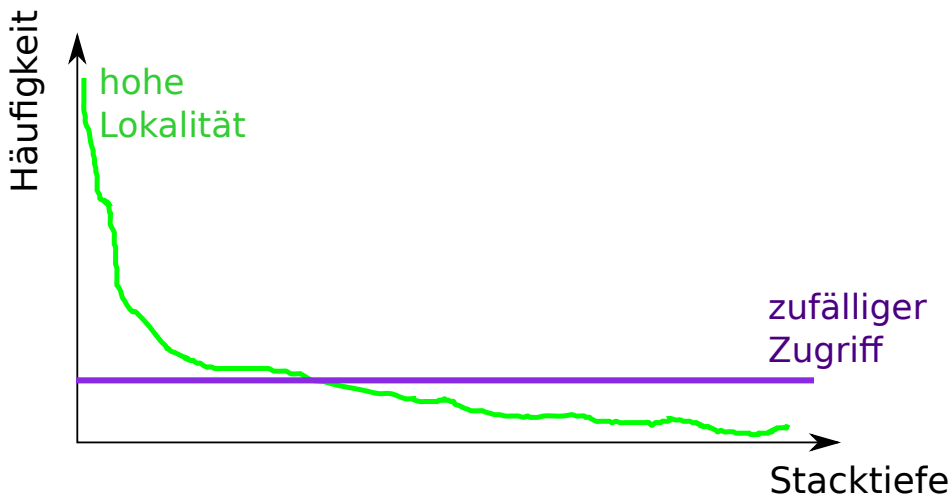
LRU-Stacktiefenverteilung

Weitere Charakterisierung der Lokalität.

- Präziser als Working-Set Ansatz.
- **Least-Recently-Used** (LRU) Stack enthält alle bereits referenzierten Seiten, sortiert bzgl. aktuellstem Zugriffszeitpunkt.

Bestimmung der Stacktiefenverteilung:

- Für jede Position des Stacks wird ein Zähler mitgeführt
- Bei Wieder-Referenz einer Seite wird der Zähler der entsprechenden Stack-Position hoch

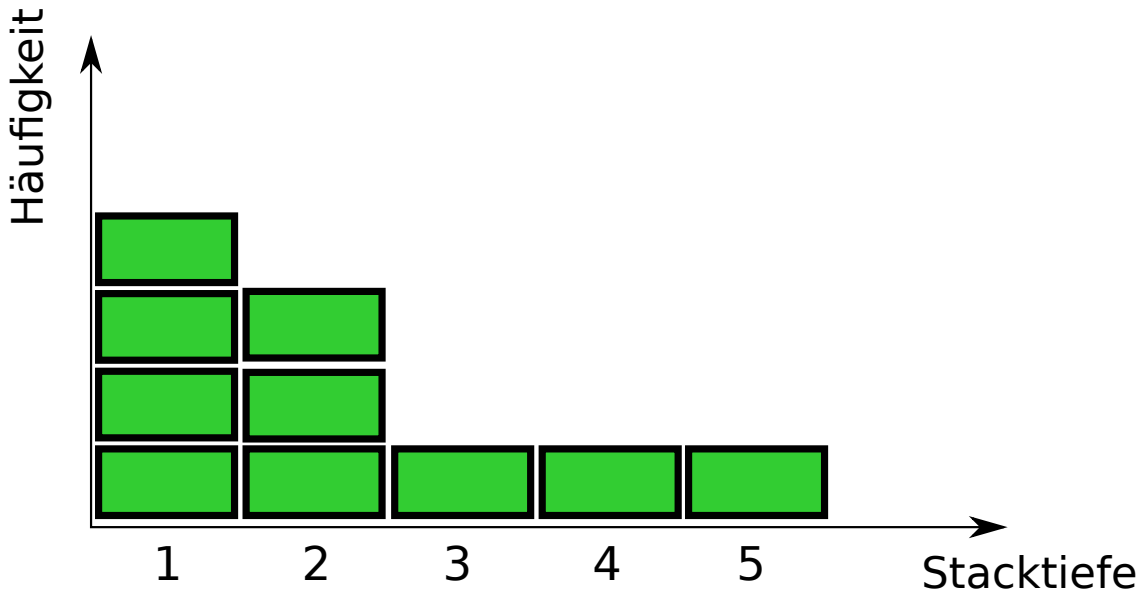


Wenn LRU als Seitenersetzungsstrategie benutzt wird, könnte man an Verteilung die Hit-Rate ablesen.

Beispiel: Berechnung der Stacktiefenverteilung

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

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



Ersetzungsstrategien: Konzept und Klassifizierung

Wenn eine Seite nicht im Puffer auffindbar ist wird sie in Puffer eingetragen. Falls dieser bereits voll ist, muss eine andere Seite weichen, aber welche?

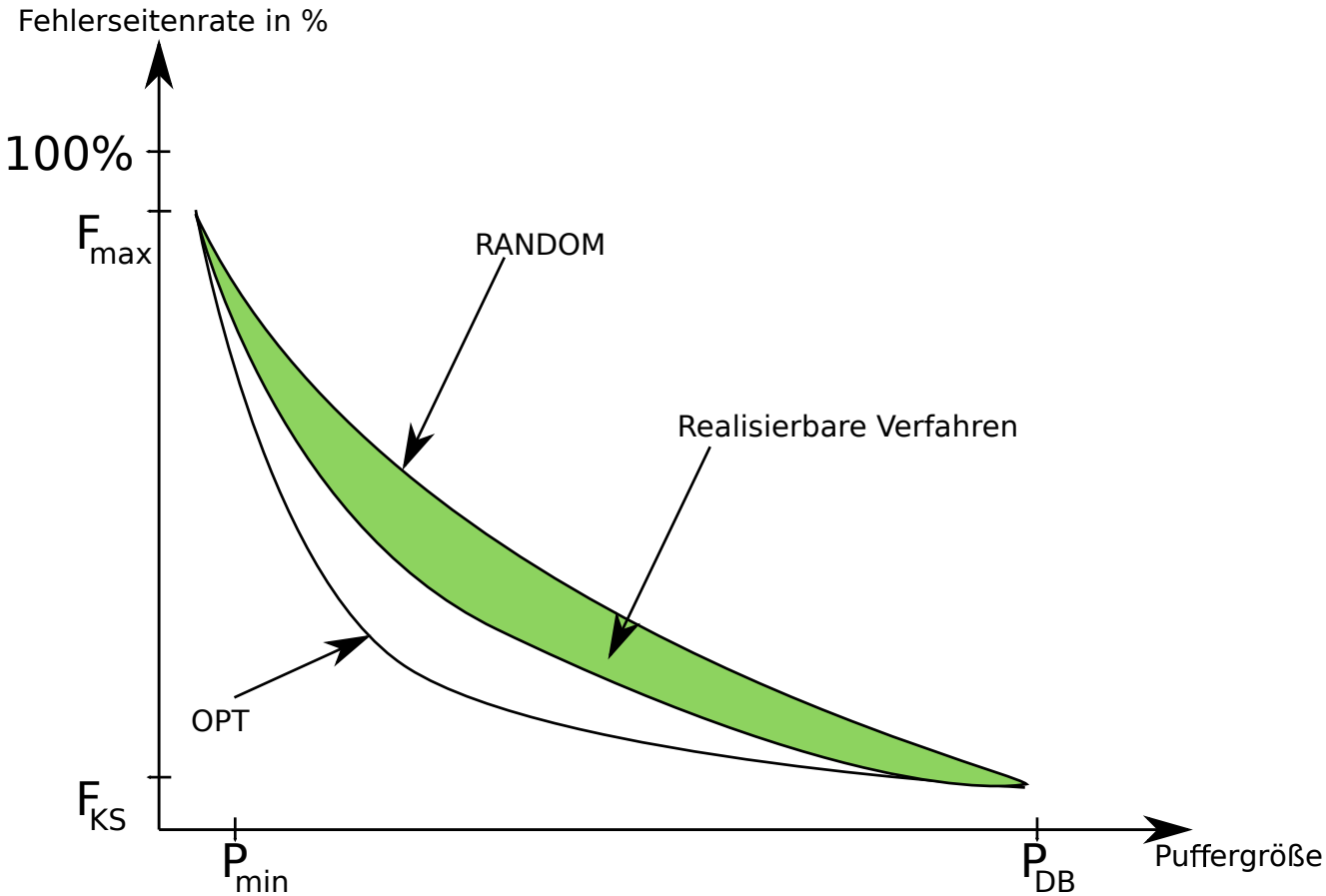
Klassifizierung von **Strategien** anhand ...

- ob das **Alter seit Einlagerung**, seit letztem Zugriff oder überhaupt nicht, und
- ob alle **Referenzen**, die letzte Referenz oder keine

bei der Auswahlentscheidung (welche Seite ersetzt werden soll) zum Tragen kommt.

Erste (triviale) Idee: Zufälliges Ersetzen von Seiten (RANDOM Strategie).

Optimales vs. Zufälliges vs. Realisierbare Verfahren



P_{min}= minimale Größe des DB-Puffers
P_{DB} = Datenbankgröße
F_{KS} = Fehlerseitenrate bei Kaltstart

FIFO

First-In, First-Out

- Ersetzt diejenige Seite, die am längsten im Puffer ist

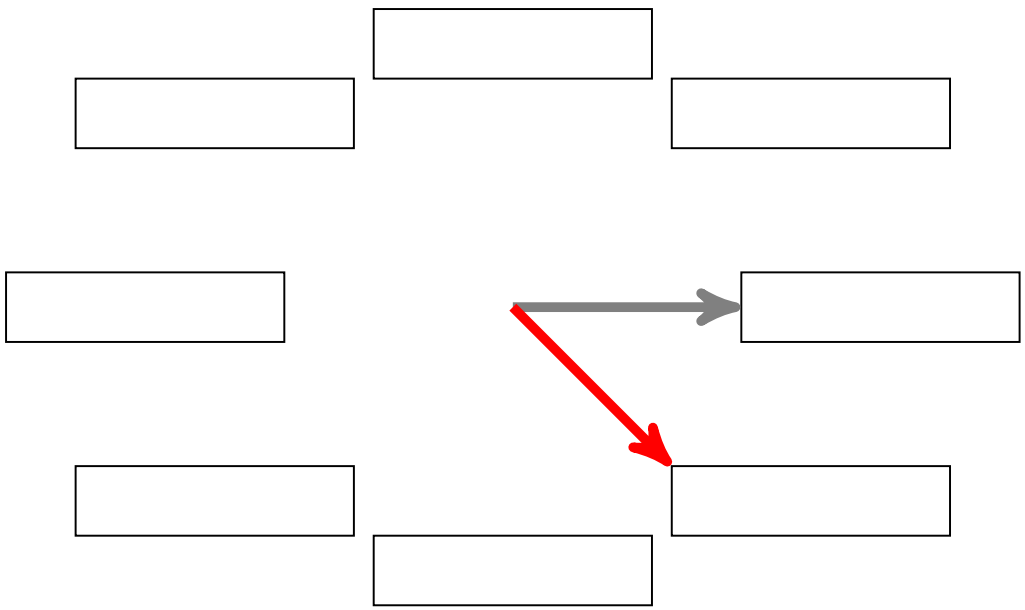


Illustration: Kreisförmige Anordnung. Zeiger verweist auf den älteste Eintrag. Grauer Zeiger= alt. Roter Zeiger=neu.

Least-Frequently-Used (LFU)

- **Ersetzt Seite mit der geringsten Referenz- (Zugriffs-) Häufigkeit**
- Für jede Seite wird ein **Zugriffszähler** (aka. Referenzzähler RZ) geführt.
- **Die Seite mit dem kleinsten Zählerstand wird ersetzt.**

Seiten, auf die kurzzeitig sehr sehr oft zugegriffen wurde bleiben sehr lange im Puffer, auch wenn nicht mehr wirklich benutzt.

Alter des Zugriffs (bzw. der letzten Zugriffe) wird nicht berücksichtigt.

- "Problem" kann durch periodisches Herabsetzen der Referenzzähler adressiert werden.

Least-Recently-Used (LRU)

- **Ersetzung basierend auf Zeit seit dem letzten Zugriff auf Seite.**
- Halte Seiten in Form eines Stacks:
  - Eine Seite kommt bei jeder Referenz auf oberste Position
  - Seite auf der untersten Position des Stacks wird bei Bedarf ersetzt

Beispiel:

Zugriff (in dieser Reihenfolge) auf Seiten:
A, B, C, D, A, C, A, B, B, B, C, D, A. Puffer hat Platz für 3 Seiten. Seite E soll eingelagert werden. Welche Seite muss weichen?

Beobachtung:

Was passiert, wenn in einer Leseoperation von der Festplatte viele neue Seiten gelesen werden?
Wieso ist das problematisch?

Theoretische Sicht

Wir betrachten einen **String von Referenzen auf Seiten**

$$r_1, r_2, \ldots, r_t, \ldots$$

wobei $r_t = p$ bedeutet, dass auf Seite p zugegriffen wurde.
Zu einem Zeitpunkt t nehmen wir an, dass jede **Seite eine gewisse Wahrscheinlichkeit b_p besitzt als nächstes Aufgerufen zu werden**, d.h. $Pr(r_{t+1} = p) = b_p$.

Interarrival Time

- **Wie viele Zugriffe liegen zwischen den Zugriffen auf eine Seite p?** Genau b_p^{-1}.

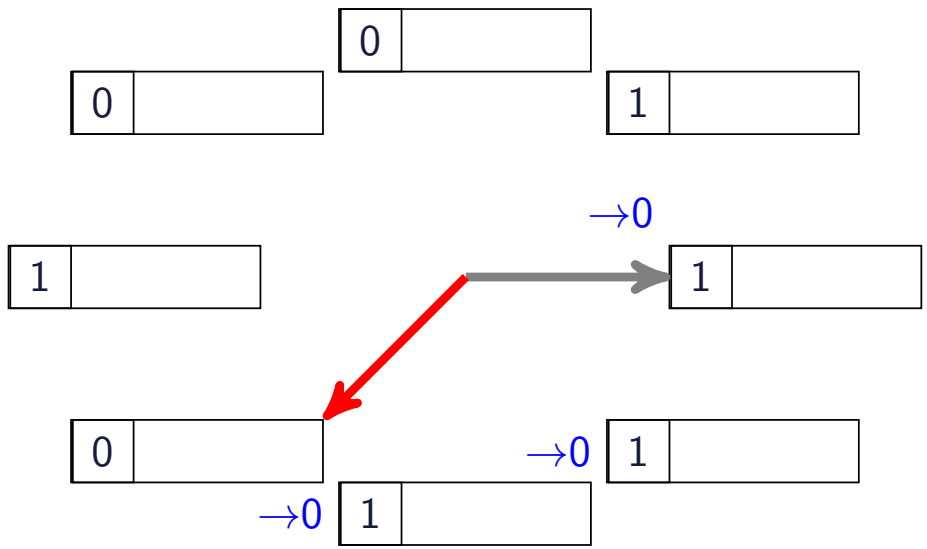
Interarrival Time Annäherung durch LRU

- Interarrival Time: Zwischen zwei Zugriffen auf Seite p liegen b_p^{-1} Zugriffe.

- **Diejenigen Seiten mit kleinster Interarrival Time bzw. größter Wahrscheinlichkeit b_p sollten im Puffer gehalten werden.**
- Dies wird **bei LRU approximiert** durch die Seiten mit dem Zeitpunkt des letzten Zugriffs auf Seite.

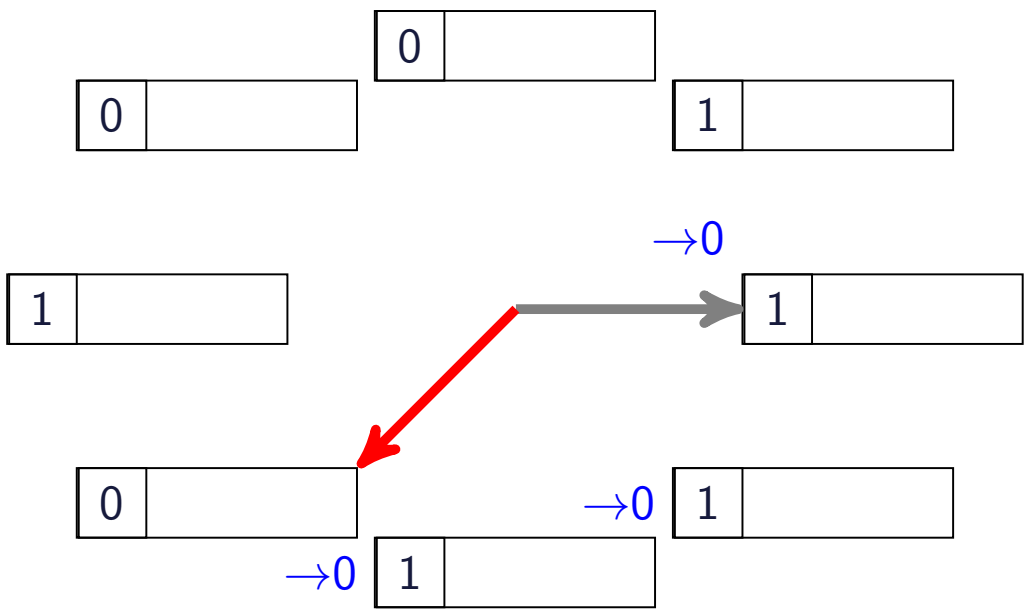
CLOCK

- Modifikation von FIFO: Jede Seite enthält ein **Benutzt-Bit, das bei Seitenreferenz auf 1 gesetzt wird.**
- Zyklische Suche: **Falls Bit auf 1, so wird es auf 0 gesetzt, nichts passiert. Zeiger wandert weiter. Falls Bit bereits auf 0, dann wird Seite entfernt.**
- Daher wird dieser Algorithmus **auch "Zweite Chance" (Second Chance) genannt.**
- **Befindet sich eine Seite bereits im Puffer, wird Bit auf 1 gesetzt, falls es 0 war, Zeiger bewegt sich nicht!**



CLOCK (2)

- In Abbildung unten: Grauer Zeiger beschreibt Stellung des Zeigers bevor die Suche nach einer zu ersetzenden Seite los geht
- Roter Zeiger beschreibt Seite die zum Ersetzen ausgewählt wurde.
- Wie wir sehen wurden die Zählerstände der vom Zeiger besuchten Seiten von 1 auf 0 gesetzt ($\rightarrow$0).



CLOCK: Beispiel

Betrachten wir die folgende Sequenz von Seiten Zugriffen: C, B, D, D, C, E für den Fall eines Puffers der Größe 3. Der initiale Puffer ist leer. Mit $\rightarrow$ ist der Zeiger auf die Stelle (Seite) im Puffer dargestellt, die bei der nächsten Ersetzung zuerst angeschaut wird.

init		
$\rightarrow$	—	0
	—	0
	—	0

1: C		
	C	1
$\rightarrow$	—	0
	—	0

2: B		
	C	1
	B	1
$\rightarrow$	—	0

3: D		
$\rightarrow$	C	1
	B	1
	D	1

4: D		
$\rightarrow$	C	1
	B	1
	D	1

5: C		
$\rightarrow$	C	1
	B	1
	D	1

6: E		
	E	1
$\rightarrow$	B	0
	D	0

Wir sehen hier auch einen **Unterschied zu LRU**: C wird durch E ersetzt, obwohl der letzte Zugriff auf C weniger lange her ist als der Zugriff auf D oder B. Bei LRU wäre B verdrängt worden.

CLOCK: Weitere Erläuterung

Schauen wir uns den Übergang von Schritt 5 nach 6 genauer an:

5: C		
→	C	1
	B	1
	D	1

6: E		
	E	1
→	B	0
	D	0

Was ist hier passiert? Da E nicht im Puffer gespeichert war wird nun laut Zeigerstand zuerst geschaut ob C ersetzt werden kann. Da C einen Zählerstand von 1 hat wird C nicht ersetzt, bekommt aber den neuen Zählerstand von 0. Der Zeiger geht weiter auf B. Auch B kann nicht ersetzt werden: also bekommt auch B einen neuen Zählerstand von 0 und der Zeiger geht weiter auf D. Auch hier kann nur der Zählerstand auf 0 gesetzt werden. Und der Zeiger geht auf C, das nun einen Zähler von 0 hat und durch E ersetzt wird. Im folgenden sind diese Schritt angegeben:

5: C		
→	C	1
	B	1
	D	1

6a: E		
	C	0
→	B	1
	D	1

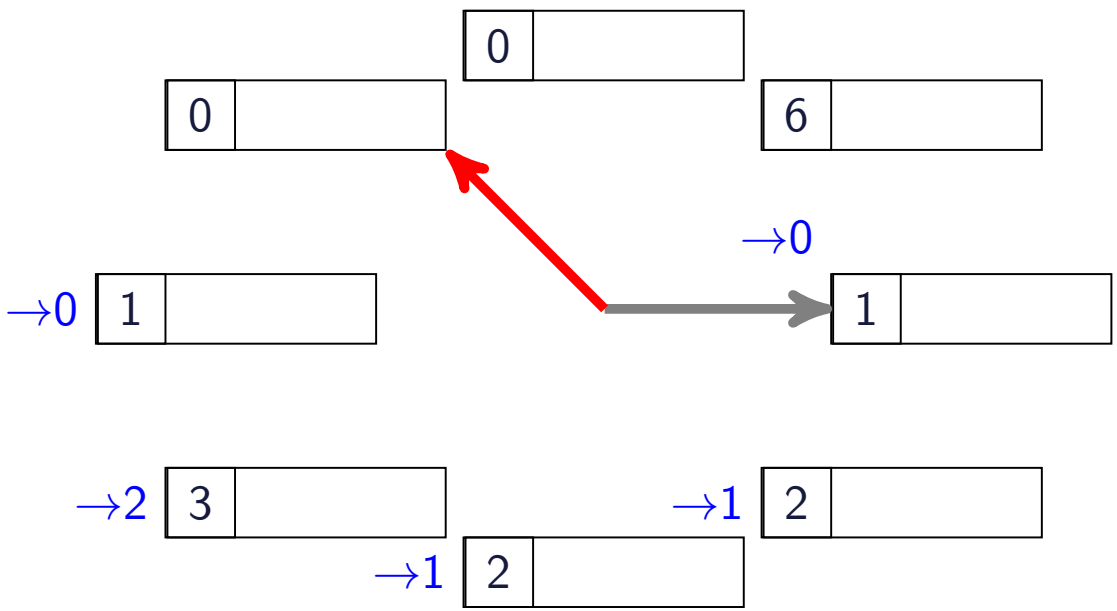
6b: E		
	C	0
	B	0
→	D	1

6c: E		
→	C	0
	B	0
	D	0

6d: E		
	E	1
→	B	0
	D	0

GCLOCK: Generalized CLOCK

- **Jede Seite bekommt Referenzzähler (RZ)**
- **Bei Referenz wird dieser erhöht**
- Wie bei CLOCK: **zyklische Suche**
  - Zähler wird herabgesetzt.
  - Null? Dann wird Seite ersetzt. Nicht Null? Dann weiter suchen.



GCLOCK: Varianten zur Berücksichtigung verschiedener Typen

Für verschiedene Typen T_i von Seite (z.B. Index-Seiten, Datenseiten) können verschiedene Werte zur Initialisierung und Inkrement von RZ angegeben werden.

Für Seite j vom Typ i kann Referenzzähler wie folgt verändert werden:

V1: -bei Erstreferenz: $RZ(j) = E_i$
-bei jeder weiteren Referenz: $RZ(j) = RZ(j) + W_i$

V2: -bei Erstreferenz: $RZ(j) = E_i$
-bei jeder weiteren Referenz: $RZ(j) = W_i$

Beobachtung: Wenn $E_i = 1$ und $W_i = 1$ (für alle i), dann geht V2 über in CLOCK, während V1 die Grundversion von GCLOCK darstellt.

LRU-k

- **Die letzten k Referenzzeitpunkte einer Seite werden berücksichtigt.**
- **Kann zwischen häufig und weniger häufig referenzierten Seiten unterscheiden**

Paper von J. O'Neill, P. O'Neill und G. Weikum: "The LRU-K Page Replacement Algorithm For Database Disk Buffering", aus dem Jahr 1993.

LRU-k: Definition

Backward k-Distance

Wir betrachten wieder einen String von Referenzen auf Seiten

$$r_1, r_2, \ldots, r_t, \ldots$$

Die **backward k-distance** $b_t(p,k)$ ist die **Distanz zur k-jüngsten Referenz** auf Seite p.

$$b_t(p,k) = \begin{cases} x & \text{falls } r_{t-x} \text{ die Seite } p \text{ referenziert und es gab genau } k-1 \text{ Auftreten einer Referenz auf } p \text{ in der Zeit zwischen } t-x \text{ und } t. \\ \infty & \text{falls } p \text{ weniger als } k \text{ mal referenziert wurde in } r_1, r_2, \ldots r_t \end{cases}$$

- **Ersetze die Seite mit der maximalen** $b_t(p,k)$.
- Es kann vorkommen, dass mehrere Seiten $b_t(p,k) = \infty$ haben, dann Ersetzung durch alternative Strategie (z.B. LRU)

Übersicht

Welche Ersetzungsstrategie passt in welche Zelle der Tabelle? (Mehrfachnennung möglich)

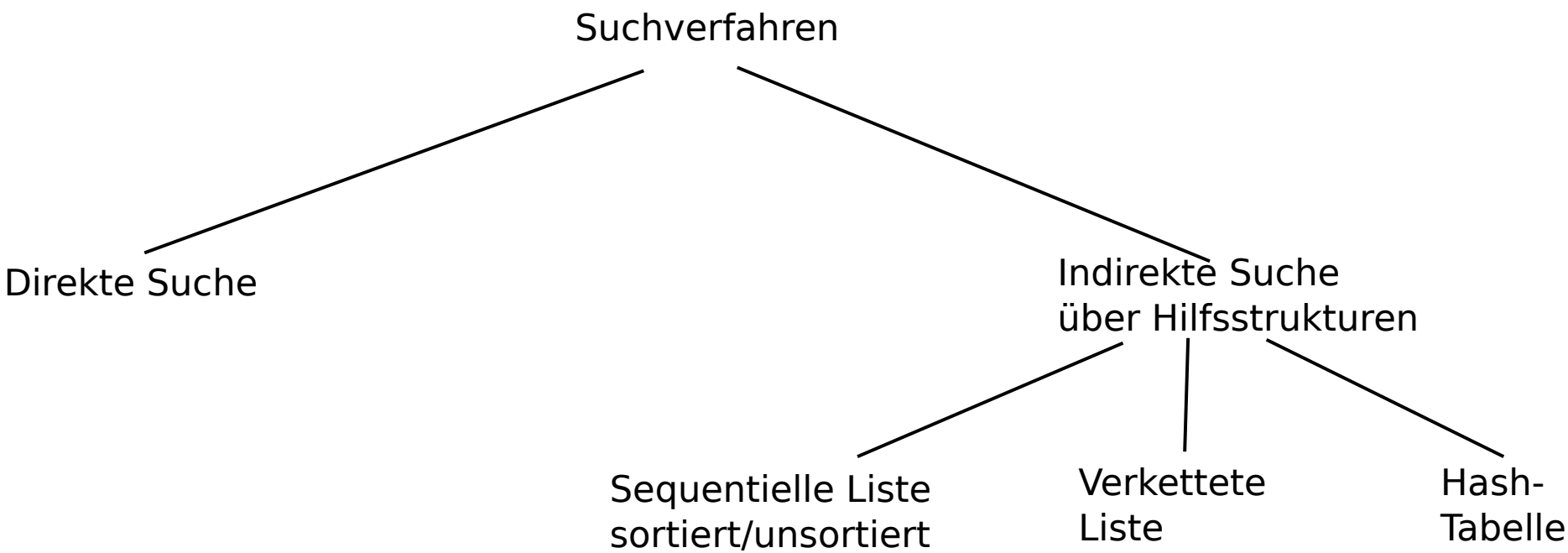
Berücksichtigung bei Auswahlentscheidung		Alter		
		nicht	seit letzter Referenz	seit Einlagerung
Referenzen	keine			
	letzte Referenz			
	alleReferenzen			

Übersicht

Welche Ersetzungsstrategie passt in welche Zelle der Tabelle? (Mehrfachnennung möglich)

<table>
<tr><td colspan="2" rowspan="2">Berücksichtigung
bei Auswahlentscheidung</td><td colspan="3">Alter</td></tr>
<tr><td>nicht</td><td>seit letzter Ref.</td><td>seit Einlagerung</td></tr>
<tr><td rowspan="3">Referenzen</td><td>keine</td><td>RAND.</td><td></td><td>FIFO</td></tr>
<tr><td>letzte Referenz</td><td></td><td>LRU, CLOCK
GCLOCK-V2</td><td></td></tr>
<tr><td>alleReferenzen</td><td>LFU</td><td>GCLOCK-V1
LRU-K</td><td></td></tr>
</table>

Auffinden einer Seite



Puffer: Ausblick

Ausblick

- Was passiert mit Seiten im Puffer, die geändert werden? Gleich auf Festplatte schreiben oder erstmal noch im Puffer lassen?
- Was passiert mit Seiten, die von noch laufender Transaktion benutzt werden aber ersetzt werden soll? Ersetzung zulassen oder nicht?
- Welche Auswirkungen hat dieses auf Fehlerbehandlung?

  $\rightarrow$ Kapitel über Recovery