Teil VII

Physische Datenorganisation

Physische Datenorganisation: Haupt- und Sekundärspeicherstrukturen

Speichermedien

- verschiedene Zwecke:
  - Daten zur Verarbeitung bereitstellen
  - Daten langfristig speichern (und trotzdem schnell verfügbar halten)
  - Daten sehr langfristig und preiswert archivieren unter Inkaufnahme etwas längerer Zugriffszeiten
- in diesem Abschnitt:
  - Speicherhierarchie
  - Magnetplatte
  - Kapazität, Zugriffskosten, Geschwindigkeit

Speicherhierarchie

1. Extrem schneller Prozessor mit Registern
2. Sehr schneller **Cache-Speicher**
3. Schneller **Hauptspeicher**
4. Langsamer **Sekundärspeicher** mit wahlfreiem Zugriff
5. Sehr langsamer **Nearline-Tertiärspeicher** bei dem die Speichermedien automatisch bereitgestellt werden
6. Extrem langsamer **Offline-Tertiärspeicher**, bei dem die Speichermedien per Hand bereitgestellt werden

- Tertiärspeicher: CD-R (Compact Disk Recordable), CD-RW (Compact Disk ReWritable), DVD (Digital Versatile Disks), Magnetbänder etwa DLT (Digital Linear Tape)

Cache-Hierarchie

- Eigenschaften der Speicherhierarchie
  - Ebene x (etwa Ebene 3, der Hauptspeicher) hat wesentlich schnellere Zugriffszeit als Ebene $x + 1$ (etwa Ebene 4, der Sekundärspeicher)
  - aber gleichzeitig einen weitaus höheren Preis pro Speicherplatz
  - und deshalb eine weitaus geringere Kapazität
  - Lebensdauer der Daten erhöht sich mit der Höhe der Ebenen

Cache-Hierarchie /2

- **Zugriffslücke** (Unterschiede zwischen den Zugriffsgeschwindigkeiten auf die Daten) vermindern $\Rightarrow$ Cache-Speicher speichern auf Ebene x Daten von Ebene $x + 1$ zwischen:
  - **Cache** (Hauptspeicher-Cache) schnellere Halbleiterspeicher-Technologie für die Bereitstellung von Daten an Prozessor (Ebene 2 in der Speicherhierarchie)
  - **Plattenspeicher-Cache** im Hauptspeicher: **Puffer**
  - Cache beim Zugriff auf Daten im WWW über HTTP: Teil des Plattenspeichers, der Teile der im Internet bereitgestellten Daten zwischenspeichert

Zugriffslücke

- Magnetplatten pro Jahr 70% mehr Speicherdichte
- Magnetplatten pro Jahr 7% schneller
- Prozessorleistung pro Jahr um 70% angestiegen
- Zugriffslücke zwischen Hauptspeicher und Magnetplattenspeicher beträgt 10^5
- Größen:
  - ns für Nanosekunden (also 10^{-9} Sekunden, ms für Millisekunden (10^{-3} Sekunden)
  - KB (KiloByte = 10^3 Bytes), MB (MegaByte = 10^6 Bytes), GB (GigaByte = 10^9 Bytes) und TB (TeraByte = 10^{12} Bytes)

Zugriffslücke in Zahlen

Speicherart	**typische Zugriffszeit**		**typische Kapazität**
	time	CPU cycles	
Cache-Speicher	6 ns	12	512 KB bis 32 MB
Hauptspeicher	60 ns	120	128 MB bis 64 GB
— Zugriffslücke 10^5 —			
Magnetplatten-speicher	8-12 ms	$16*10^6$	160 GB bis 4 TB
Platten-Farm oder -Array	12 ms	$24*10^6$	im TB/PB-Bereich

Lokalität des Zugriffs

- Caching-Prinzip funktioniert nicht, wenn immer neue Daten benötigt werden
- in den meisten Anwendungsfällen: **Lokalität** des Zugriffs
- D.h., Großteil der Zugriffe (in den meisten Fällen über 90%) auf Daten aus dem jeweiligen Cache
- Deshalb: Pufferverwaltung des Datenbanksystems wichtiges Konzept

Typische Merkmale von Sekundärspeicher

Merkmal	Kapazität	Latenz	Bandbreite
1983	30 MB	48.3 ms	0.6 MB/s
1994	4.3 GB	12.7 ms	9 MB/s
2003	73.4 GB	5.7 ms	86 MB/s
2009	2 TB	5.1 ms	95 MB/s
2010 SSD	500 GB	read 65 μs write 85 μs	read 250 MB/s write 170 MB/s
2015 SSD	4 TB	read 0,031 ms write 0,023 ms	bis 510 MB/s bis 490 MB/s
2018 SSD	100 TB		bis 3500 MB/s

Speicherkapazität und Kosten

Größe	Information oder Medium
1 KB	$= 1.000$
0.5 KB	Buchseite als Text
30 KB	eingescannte, komprimierte Buchseite
1 MB	$= 1.000.000$
5 MB	Die Bibel als Text
20 MB	eingescanntes Buch
500 MB	CD-ROM; Oxford English Dictionary
1 GB	$= 1.000.000.000$
4.7 GB	Digital Versatile Disk (DVD)
10 GB	komprimierter Spielfilm
100 GB	ein Stockwerk einer Bibliothek
200 GB	Kapazität eines Videobandes

Speicherkapazität und Kosten /2

Größe	Information oder Medium
1 TB	$= 1.000.000.000.000$
1 TB	Bibliothek mit 1M Bänden
30 TB	Library of Congress Bände als Text gespeichert
1 PB	$= 1.000.000.000.000.000$
1 PB	Eingescannte Bände einer Nationalen Bibliothek
1 PB	223,101 DVD's
15 PB	weltweite Plattenproduktion in 1996
200 PB	weltweite Magnetbandproduktion in 1996
$>$55 EB	weltweite Plattenproduktion in 2009
$>$ 2596 EB	Gesamtspeicherkapazität in 2012
$>$ 163 ZB	Menge an Daten in 2025 ?

Speicherarrays: RAID

- Kopplung billiger Standard-platten unter einem speziellen Controller zu einem einzigen logischen Laufwerk
- Verteilung der Daten auf die verschiedenen physischen Festplatten übernimmt Controller
- zwei gegensätzliche Ziele:
  - Erhöhung der Fehlertoleranz (Ausfallsicherheit, Zuverlässigkeit) durch Redundanz
  - Effizienzsteigerung durch Parallelität des Zugriffs

Erhöhung der Fehlertoleranz

- Nutzung zusätzlicher Platten zur Speicherung von Duplikaten (Spiegeln) der eigentlichen Daten $\Rightarrow$ bei Fehler: Umschalten auf Spiegelplatte
- bestimmte RAID-Levels (1, 0+1) erlauben eine solche Spiegelung
- Alternative: Kontrollinformationen wie Paritätsbits nicht im selben Sektor wie die Originaldaten, sondern auf einer anderen Platte speichern
- RAID-Levels 2 bis 6 stellen durch Paritätsbits oder Error Correcting Codes (ECC) fehlerhafte Daten wieder her
- ein Paritätsbit kann einen Plattenfehler entdecken und bei Kenntnis der fehlerhaften Platte korrigieren

Erhöhung der Effizienz

- Datenbank auf mehrere Platten verteilen, die parallel angesteuert werden können $\Rightarrow$ Zugriffszeit auf große Datenmengen verringert sich fast linear mit der Anzahl der verfügbaren Platten
- Verteilung: bit-, byte- oder blockweise
- höhere RAID-Levels (ab Level 3) verbinden Fehlerkorrektur und block- oder bitweises Verteilen von Daten
- Unterschiede:
  - **schnellerer Zugriff** auf bestimmte Daten
  - **höherer Durchsatz** für viele parallel anstehende Transaktionen durch eine Lastbalancierung des Gesamtsystems

Sicherungsmedien: Tertiärspeicher

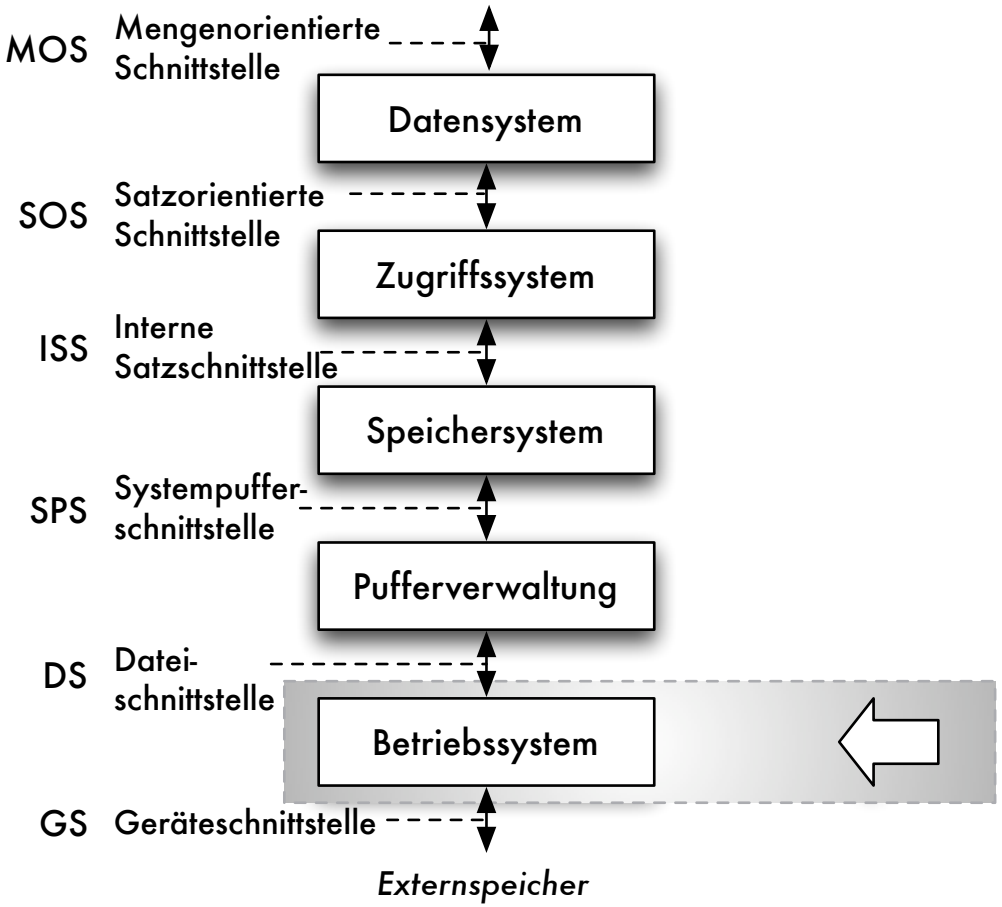
- weniger oft benutzte Teile der Datenbank, die eventuell sehr großen Umfang haben (Text, Multimedia) „billiger“ speichern als auf Magnetplatten
- aktuell benutzte Datenbestände zusätzlich sichern (archivieren)
- Tertiärspeicher: Medium austauschbar
  - **offline**: Medien manuell wechseln (optische Platten, Bänder)
  - **nearline**: Medien automatisch wechseln (*Jukeboxes*, *Bandroboter*)

Langzeitarchivierung

- Lebensdauer, Teilaspekte:
  - physische Haltbarkeit des Mediums garantiert die **Unversehrtheit** der Daten: 10 Jahre für Magnetbänder, 30 Jahre für optische Platten, Papier???
  - Vorhandensein von Geräten und Treibern garantiert die **Lesbarkeit** von Daten: Geräte für Lochkarten oder 8-Zoll-Disketten?
  - zur Verfügung stehende Metadaten garantieren die **Interpretierbarkeit** von Daten
  - Vorhandensein von Programmen, die auf den Daten arbeiten können, garantieren die **Wiederverwendbarkeit** von Daten

Verwaltung des Hintergrundspeichers

- Abstraktion von Speicherungs- oder Sicherungsmedium
- Modell: Folge von Blöcken



Betriebssystemdateien

- jede Relation oder jeder Zugriffspfad in genau einer Betriebssystem-Datei
- ein oder mehrere BS-Dateien, DBS verwaltet Relationen und Zugriffspfade selbst innerhalb dieser Dateien
- DBS steuert selbst Magnetplatte an und arbeitet mit den Blöcken in ihrer Ursprungsform (*raw device*)

Betriebssystemdateien /2

- Warum nicht immer BS-Dateiverwaltung?
  - Betriebssystemunabhängigkeit
  - In 32-Bit-Betriebssystemen: Dateigröße 4 GB maximal
  - BS-Dateien auf maximal einem Medium
  - betriebssystemseitige Pufferverwaltung von Blöcken des Sekundärspeichers im Hauptspeicher genügt nicht den Anforderungen des Datenbanksystems

Blöcke und Seiten

- Zuordnung der physischen Blöcke zu **Seiten**
- meist mit festen Faktoren: 1, 2, 4 oder 8 Blöcke einer Spur auf eine Seite
- hier: „ein Block — eine Seite“
- höhere Schichten des DBS adressieren über Seitennummer

Dienste des Dateisystems

- Allokation oder Deallokation von Speicherplatz
- Holen oder Speichern von Seiteninhalten
- Allokation möglichst so, dass logisch aufeinanderfolgende Datenbereiche (etwa einer Relation) auch möglichst in aufeinanderfolgenden Blöcken der Platte gespeichert werden
- Nach vielen Update-Operationen: **Reorganisationsmethoden**
- **Freispeicherverwaltung**: doppelt verkettete Liste von Seiten

Abbildung der Datenstrukturen

- Abbildung der konzeptuellen Ebene auf interne Datenstrukturen
- Unterstützung durch **Metadaten** (im Data Dictionary, etwa das interne Schema)

Konz. Ebene	**Interne Ebene**	**Dateisystem/Platte**
Relationen $\rightarrow$	Log. Dateien $\rightarrow$	Phys. Dateien
Tupel $\rightarrow$	Datensätze $\rightarrow$	Seiten/Blöcke
Attributwerte $\rightarrow$	Felder $\rightarrow$	Bytes

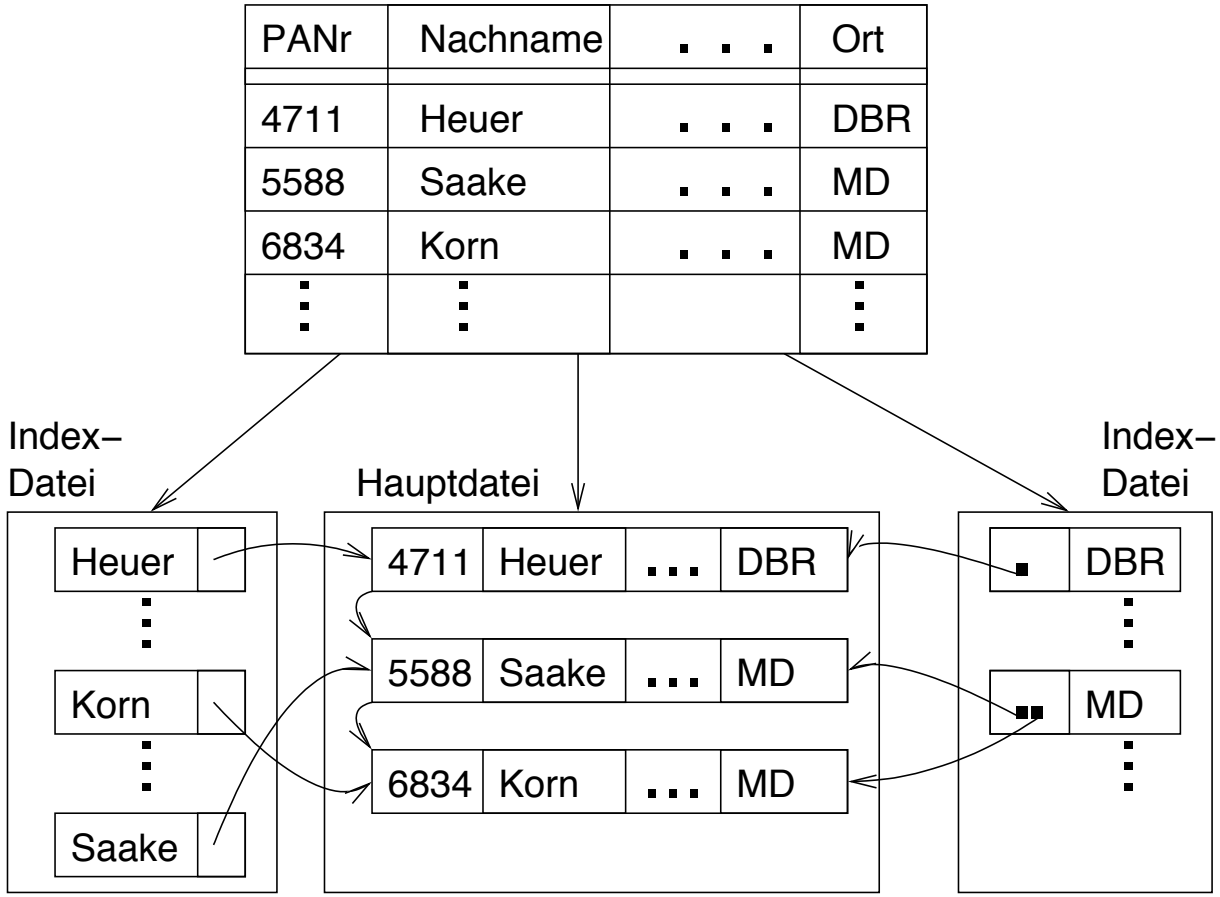
Varianten der Abbildungen

- Beispiel 1: jede Relation in je einer logischen Datei, diese insgesamt in einer einzigen physischen Datei
- Beispiel 2: Cluster-Speicherung – mehrere Relationen in einer logischen Datei

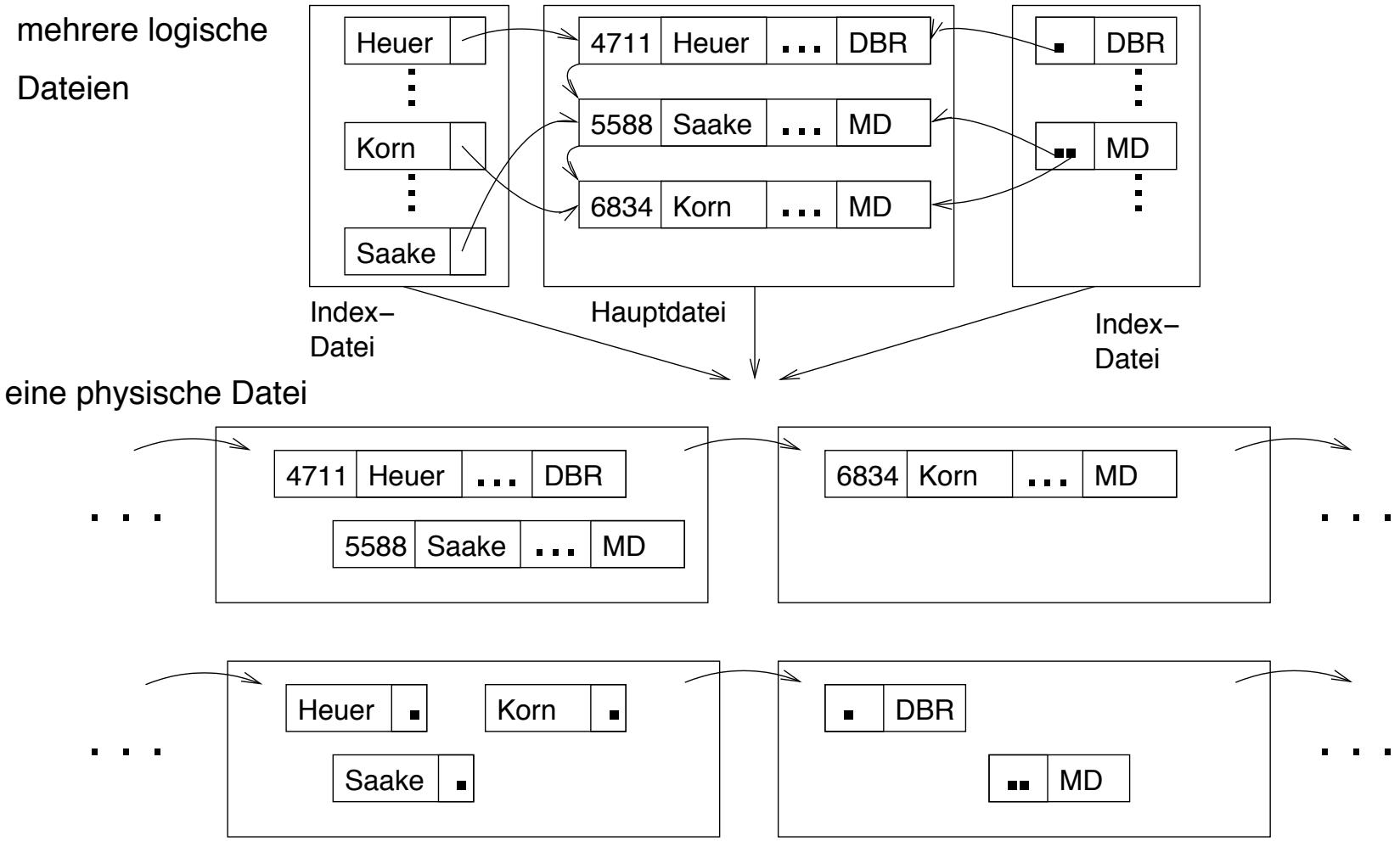
Übliche Form der Speicherung

Eine Relation

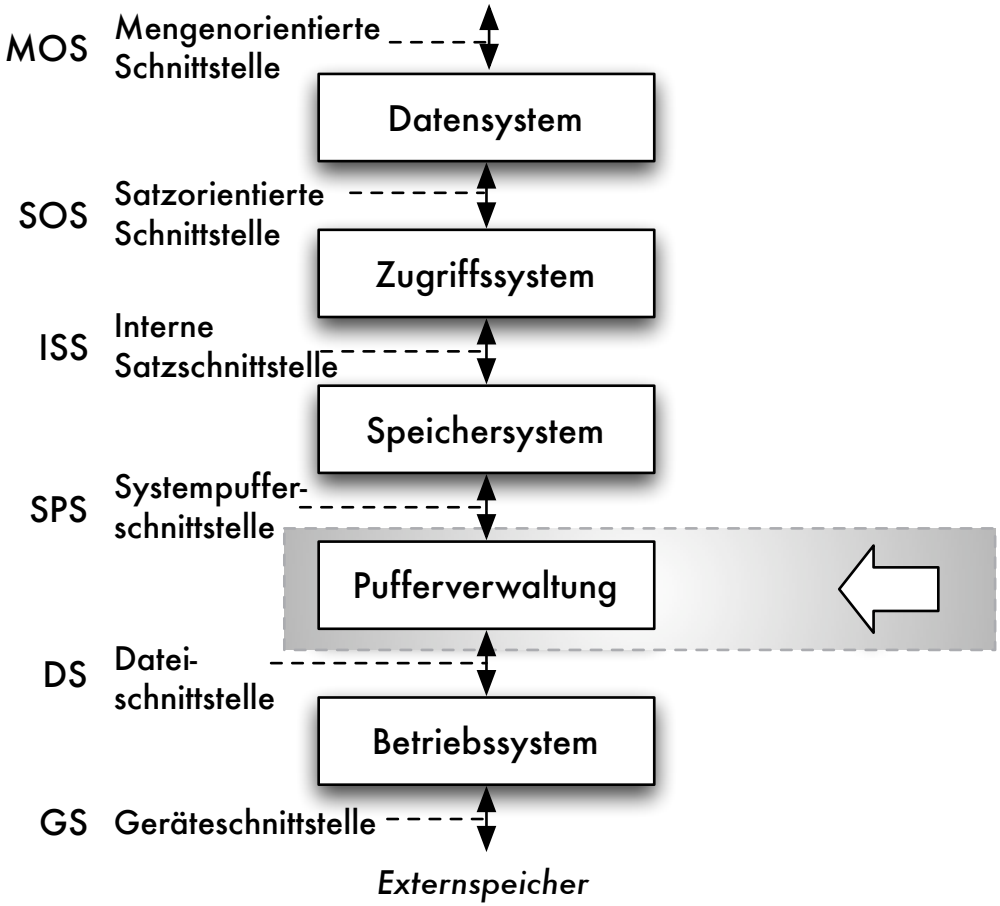
mehrere logische Dateien



Übliche Form der Speicherung /2



Pufferverwaltung im Detail



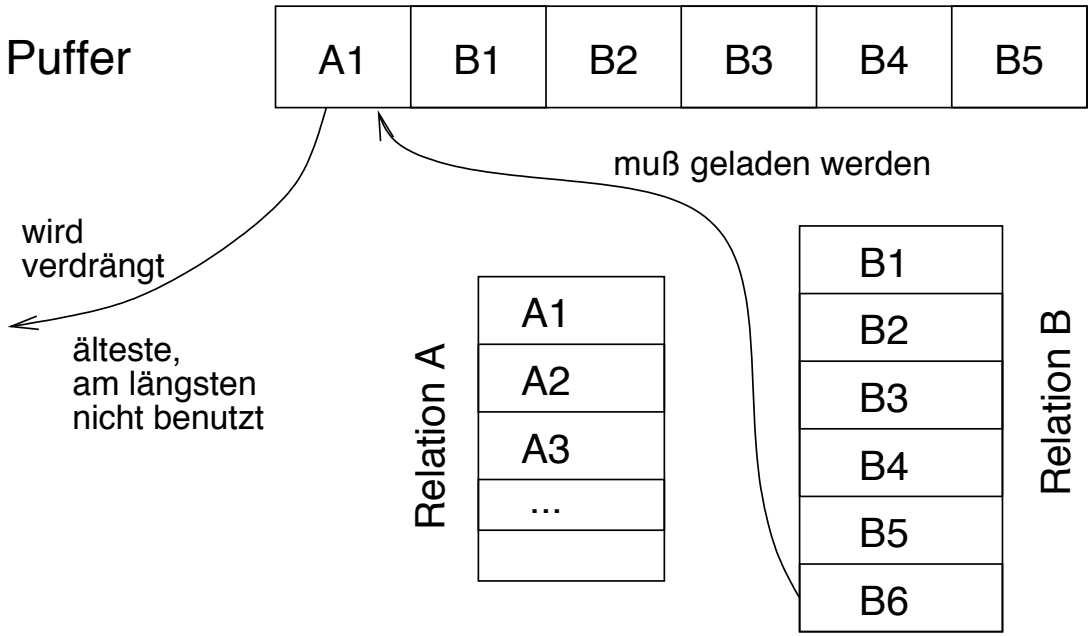
Aufgaben der Pufferverwaltung

- **Puffer**: ausgezeichneter Bereich des Hauptspeichers
- in **Pufferrahmen** gegliedert, jeder Pufferrahmen kann eine Seite der Platte aufnehmen
- Aufgaben:
  - Pufferverwaltung muss angeforderte Seiten im Puffer suchen $\Rightarrow$ effiziente **Suchverfahren**
  - parallele Datenbanktransaktionen: geschickte **Speicherzuteilung** im Puffer
  - Puffer gefüllt: adäquate **Seitenersetzungsstrategien**
  - *Unterschiede zwischen einem Betriebssystem-Puffer und einem Datenbank-Puffer*
  - spezielle Anwendung der Pufferverwaltung: **Schattenspeicherkonzept**

Aufgaben der Pufferverwaltung /2

Mangelnde Eignung des BS-Puffers

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



Mangelnde Eignung des BS-Puffers /2

- Ablauf
  - FIFO: A_1 verdrängt, da älteste Seite im Puffer
  - LRU: A_1 verdrängt, da diese Seite nur im ersten Schritt beim Auslesen des ersten Vergleichstupels benötigt wurde
- Problem
  - im nächsten Schritt wird das zweite Tupel von A_1 benötigt
  - weiteres „Aufschaukeln“: um A_1 laden zu können, muss B_1 entfernt werden (im nächsten Schritt benötigt) usw.

Suchen einer Seite

- **Direkte Suche:**
  - ohne Hilfsmittel linear im Puffer suchen
- **Indirekte Suche:**
  - Suche nur noch auf einer kleineren Hilfsstruktur
  - *unsortierte und sortierte Tabelle*: alle Seiten im Puffer vermerkt
  - *verkettete Liste*: schnelleres sortiertes Einfügen möglich
  - *Hashtabelle*: bei geschickt gewählter Hashfunktion günstigster Such- und Änderungsaufwand

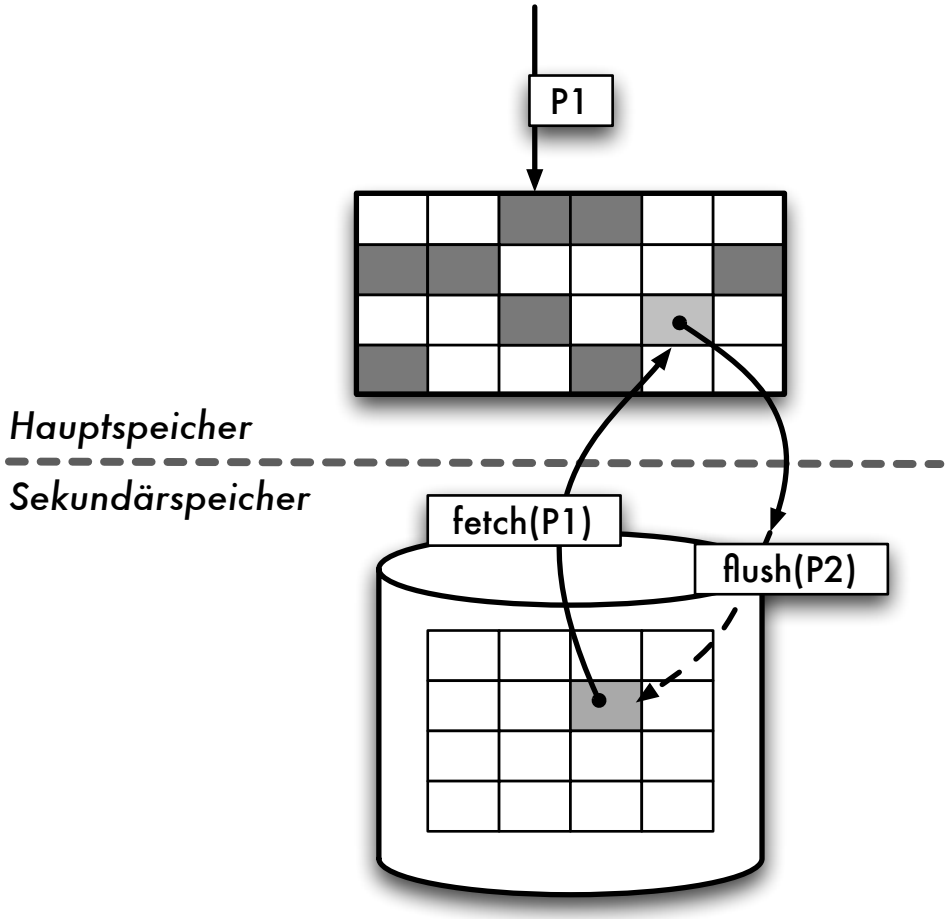
Speicherzuteilung im Puffer

- bei mehreren parallel anstehenden Transaktionen
  - **Lokale Strategien**: Jeder Transaktion bestimmte disjunkte Pufferteile verfügbar machen (Größe statisch vor Ablauf der Transaktionen oder dynamisch zur Programmlaufzeit entscheiden)
  - **Globale Strategien**: Zugriffsverhalten aller Transaktionen insgesamt bestimmt Speicherzuteilung (gemeinsam von mehreren Transaktionen referenzierte Seiten können so besser berücksichtigt werden)
  - **Seitentypbezogene Strategien**: Partition des Puffers: Pufferrahmen für Datenseiten, Zugriffspfadseiten, Data-Dictionary-Seiten, usw. - eigene Ersetzungstrategien für die jeweiligen Teile möglich

Seitenersetzungsstrategien

- Speichersystem fordert Seite E_2 an, die nicht im Puffer vorhanden ist
- Sämtliche Pufferrahmen sind belegt
- vor dem Laden von E_2 Pufferrahmen freimachen
- nach den unten beschriebenen Strategien Seite aussuchen
- Ist Seite in der Zwischenzeit im Puffer verändert worden, so wird sie nun auf Platte **zurückgeschrieben**
- Ist Seite seit Einlagerung in den Puffer nur gelesen worden, so kann sie überschrieben werden (**verdrängt**)

Seitenersetzung schematisch



Seitenersetzung in DBMS

- **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: Verfahren

- **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 (z.B. bei BLOBs sinnvoll)
- **optimale Strategie**: Welche Seite hat maximale Distanz zu ihrem nächsten Gebrauch? (nicht realisierbar, zukünftiges Referenzverhalten nicht vorhersehbar)

$\rightsquigarrow$ Realisierbare Verfahren besitzen keine Kenntnisse über das zukünftige Referenzverhalten

- *Zufallsstrategie*: jeder Seite gleiche Wiederbenutzungswahrscheinlichkeit zuordnen

Fazit

- Pufferverwaltungsstrategie mit großem Einfluss auf Performance
- in kommerziellen Systemen meist LRU mit Variationen
- besondere Behandlung von Full-Table-Scans
- weiterer Einflussfaktor: **Puffergröße**
- Indikator: Trefferrate (engl. *hit ratio*)

$$\text{hit ratio} = \frac{\text{Anz. log. Zugriffe} - \text{Anz. phys. Zugriffe}}{\text{Anz. log. Zugriffe}}$$

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

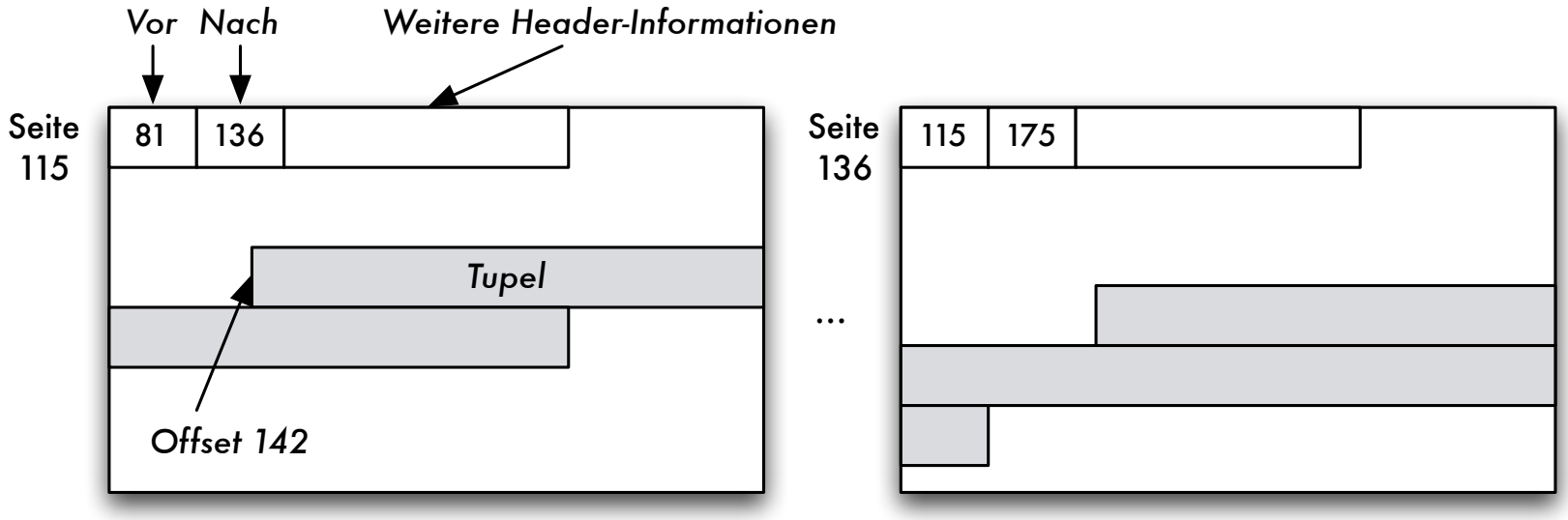
Daten, die in den nächsten 5 Min. wieder referenziert werden, sollten im Hauptspeicher gehalten werden

Seite

- Block:
  - kleinste adressierbare Einheit auf Externspeicher
  - Zuordnung zu Seiten im Hauptspeicher
- Aufbau von Seiten
  - *Header*
    - Informationen über Vorgänger- und Nachfolger-Seite
    - eventuell auch Nummer der Seite selbst
    - Informationen über Typ der Sätze
    - freier Platz
  - *Datensätze*
  - unbelegte Bytes

Seitenorganisation

- Organisation der Seiten: doppelt verkettete Liste
- freie Seiten in Freispeicherverwaltung



Seite: Adressierung der Datensätze

- adressierbare Einheiten
  - Zylinder
  - Spuren
  - Sektoren
  - Blöcke oder Seiten
  - Datensätze in Blöcken oder Seiten
  - Datenfelder in Datensätzen
- Beispiel: Adresse eines Satzes durch Seitennummer und Offset (relative Adresse in Bytes vom Seitenanfang)

$$(115, 142)$$

Seitenzugriff als Flaschenhals

- Maß für die Geschwindigkeit von Datenbankoperationen: Anzahl der Seitenzugriffe auf dem Sekundärspeicher (wegen Zugriffslücke)
- Faustregel: Geschwindigkeit des Zugriffs $\Leftarrow$ Qualität des Zugriffspfades $\Leftarrow$ Anzahl der benötigten Seitenzugriffe
- Hauptspeicheroperationen nicht beliebig vernachlässigbar

Einpassen von Datensätzen auf Blöcke

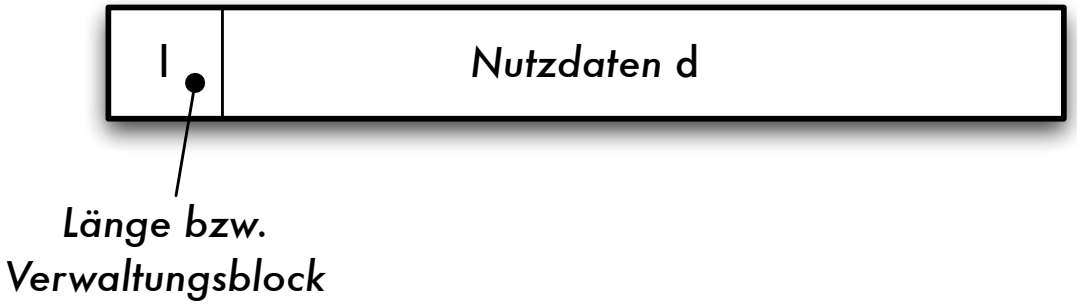
- Datensätze (eventuell variabler Länge) in die aus einer fest vorgegebenen Anzahl von Bytes bestehenden Blöcke einpassen: **Blocken**
- Blocken abhängig von variabler oder fester Feldlänge der Datenfelder
  - Datensätze mit variabler Satzlänge: höherer Verwaltungsaufwand beim Lesen und Schreiben, Satzlänge immer wieder neu ermitteln
  - Datensätze mit fester Satzlänge: höherer Speicheraufwand

Sätze fester Länge

- SQL: Datentypen fester und variabler Länge
  - *char(n)* Zeichenkette der festen Länge *n*
  - *varchar(n)* Zeichenkette variabler Länge mit der Maximallänge *n*
- Aufbau der Datensätze, falls alle Datenfelder feste Länge:
  1. *Verwaltungsblock* mit Typ eines Satzes (wenn unterschiedliche Satztypen auf einer Seite möglich) und Löschbit
  2. *Freiraum* zur Justierung des Offset
  3. *Nutzdaten* des Datensatzes

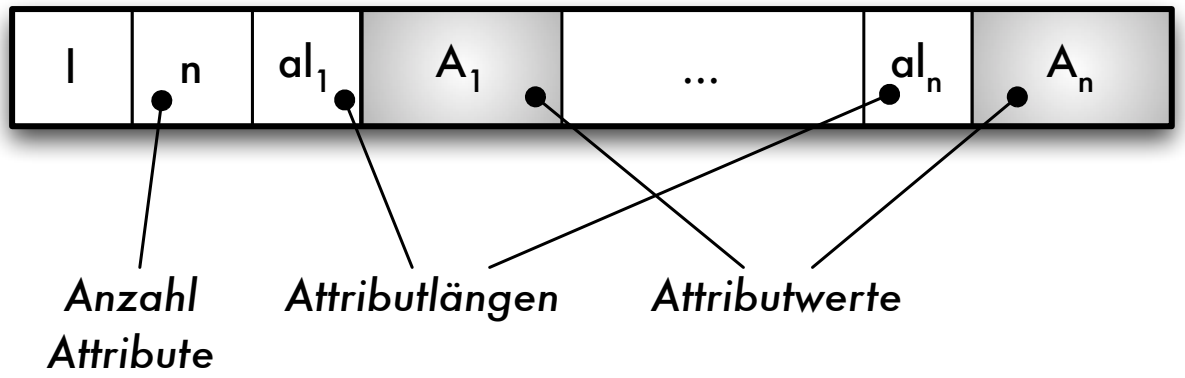
Sätze variabler Länge

- im Verwaltungsblock nötig: Satzlänge l, um die Länge des Nutzdaten-Bereichs d zu kennen

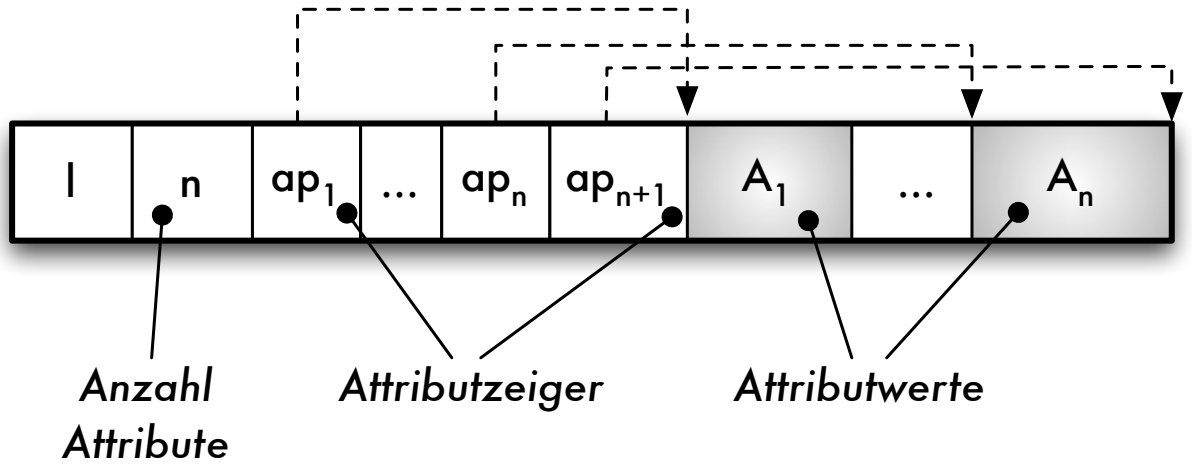


Sätze variabler Länge /2

- Strategie a)



- Strategie b)



Speicherung von Sätzen variabler Länge

- *Strategie a):* Jedes Datenfeld variabler Länge A_i beginnt mit einem *Längenzeiger* al_i, der angibt, wie lang das folgende Datenfeld ist
- *Strategie b):* Am Beginn des Satzes wird nach dem Satz-Längenzeiger l und der Anzahl der Attribute ein Zeigerfeld $ap_1, \ldots, ap_n$ für alle variabel langen Datenfelder eingerichtet

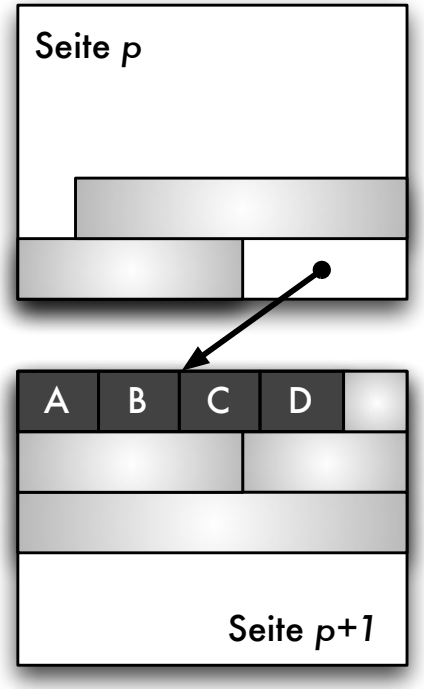
- Vorteil Strategie b): leichtere Navigation innerhalb des Satzes (auch für Sätze in Seiten $\Rightarrow$ TID)

Anwendung variabel langer Datenfelder

- „Wiederholgruppen“: Liste von Werten des gleichen Datentyps
  - Zeichenketten variabler Länge wie *varchar(n)* sind Wiederholgruppe mit *char* als Basisdatentyp, mathematisch also die Kleene’sche Hülle $(char)^*$
  - Mengen- oder listenwertige Attributwerte, die im Datensatz selbst denormalisiert gespeichert werden sollen (Speicherung als geschachtelte Relation oder Cluster-Speicherung), bei einer Liste von *integer*-Werten wäre dies $(integer)^*$
  - Adressfeld für eine Indexdatei, die zu einem Attributwert auf mehrere Datensätze zeigt (*Sekundärindex*), also $(pointer)^*$

Blockungstechniken: Nichtspannsätze

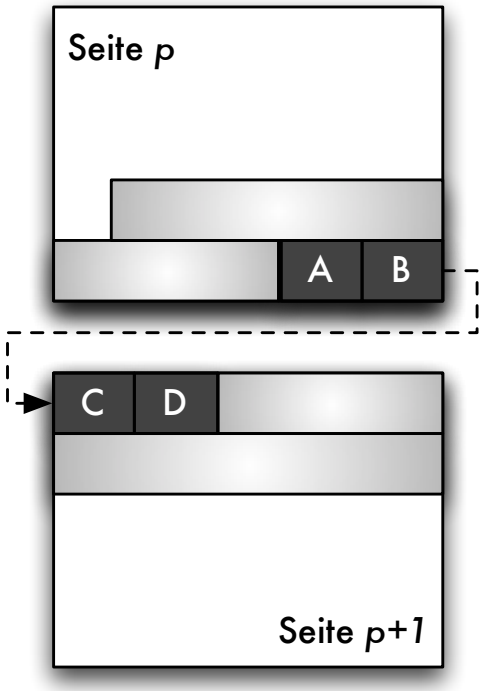
- jeder Datensatz in maximal einem Block



- Standardfall (außer bei BLOBs oder CLOBs)

Blockungstechniken: Spannsätze

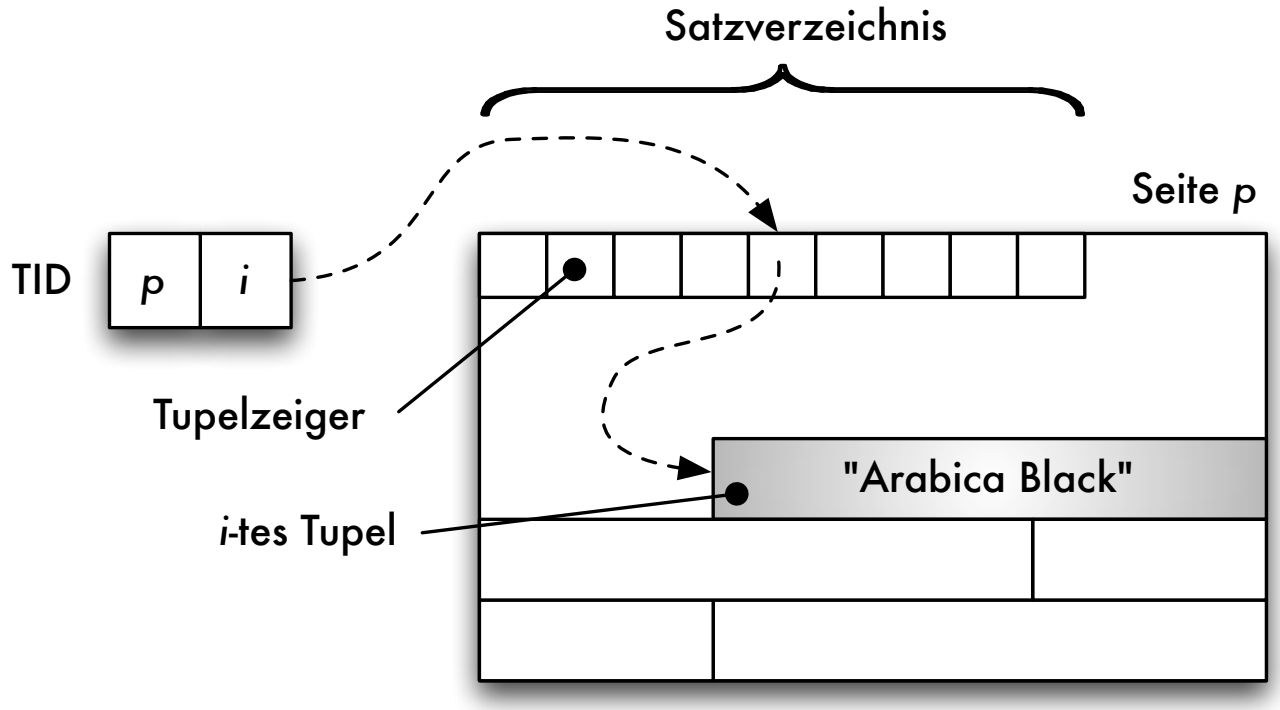
- **Spannsätze**: Datensatz eventuell in mehreren Blöcken



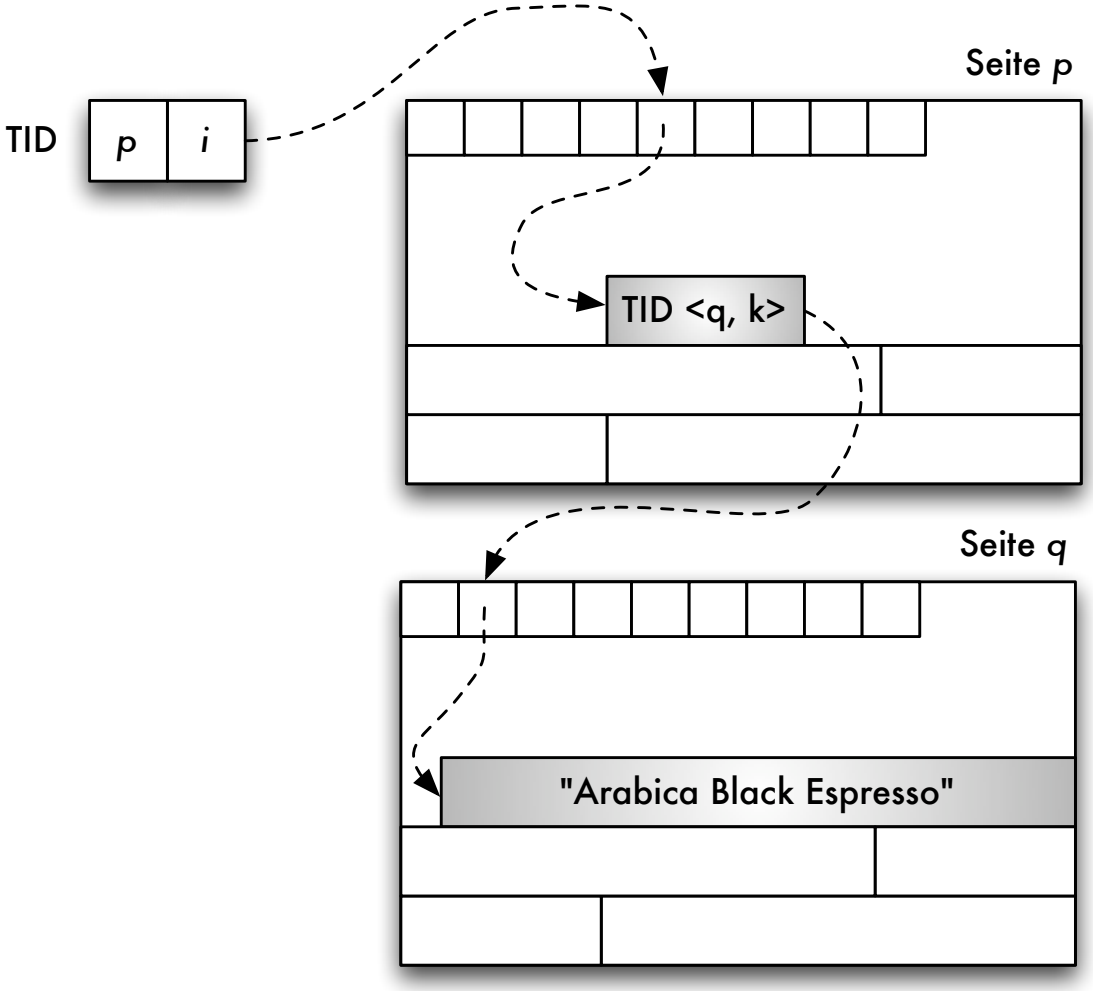
Adressierung: TID-Konzept

- **Tupel-Identifier** (TID) ist Datensatz-Adresse bestehend aus Seitennummer und Offset
- Offset verweist innerhalb der Seite bei einem Offset-Wert von i auf den i-ten Eintrag in einer Liste von **Tupelzeigern** (Satzverzeichnis), die am Anfang der Seite stehen
- Jeder Tupel-Zeiger enthält Offsetwert
- Verschiebung auf der Seite: sämtliche Verweise von außen bleiben unverändert
- Verschiebungen auf eine andere Seite: statt altem Datensatz neuer TID-Zeiger
- diese zweistufige Referenz aus Effiziengründen nicht wünschenswert: Reorganisation in regelmäßigen Abständen

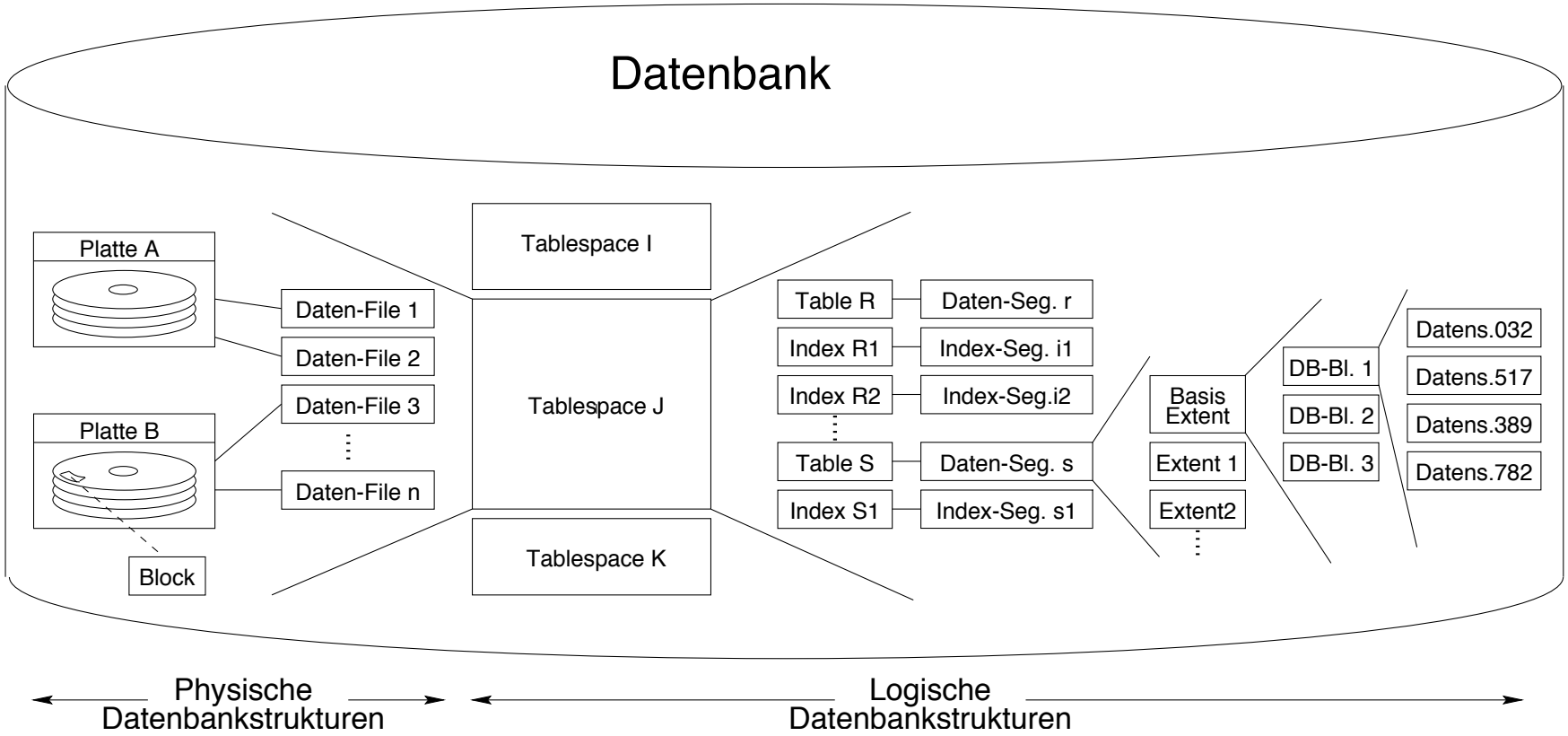
TID-Konzept: einstufige Referenz



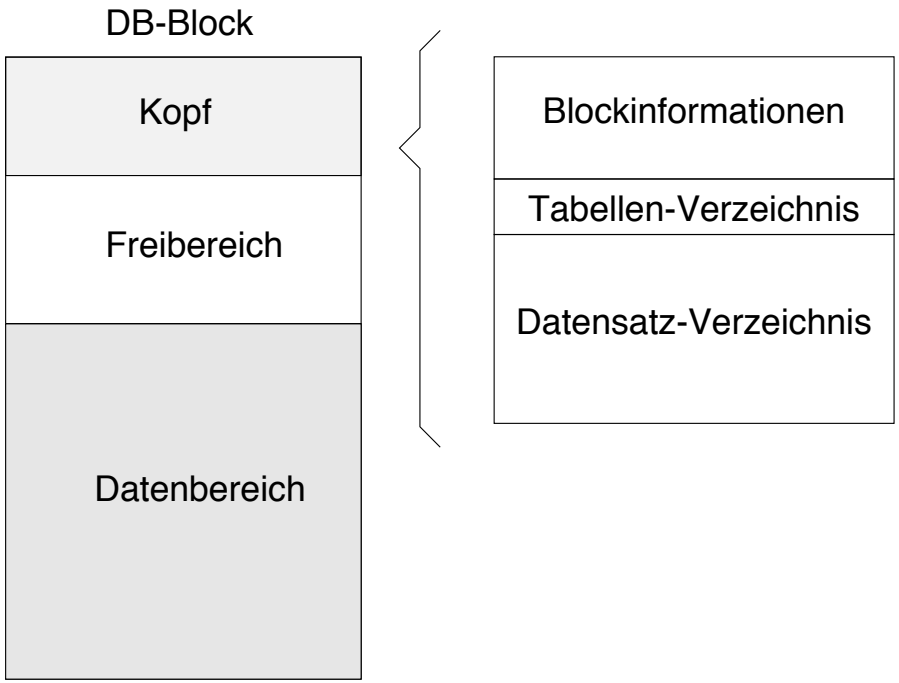
TID-Konzept: zweistufige Referenz



Oracle: Datenbankstruktur



Oracle: Blöcke

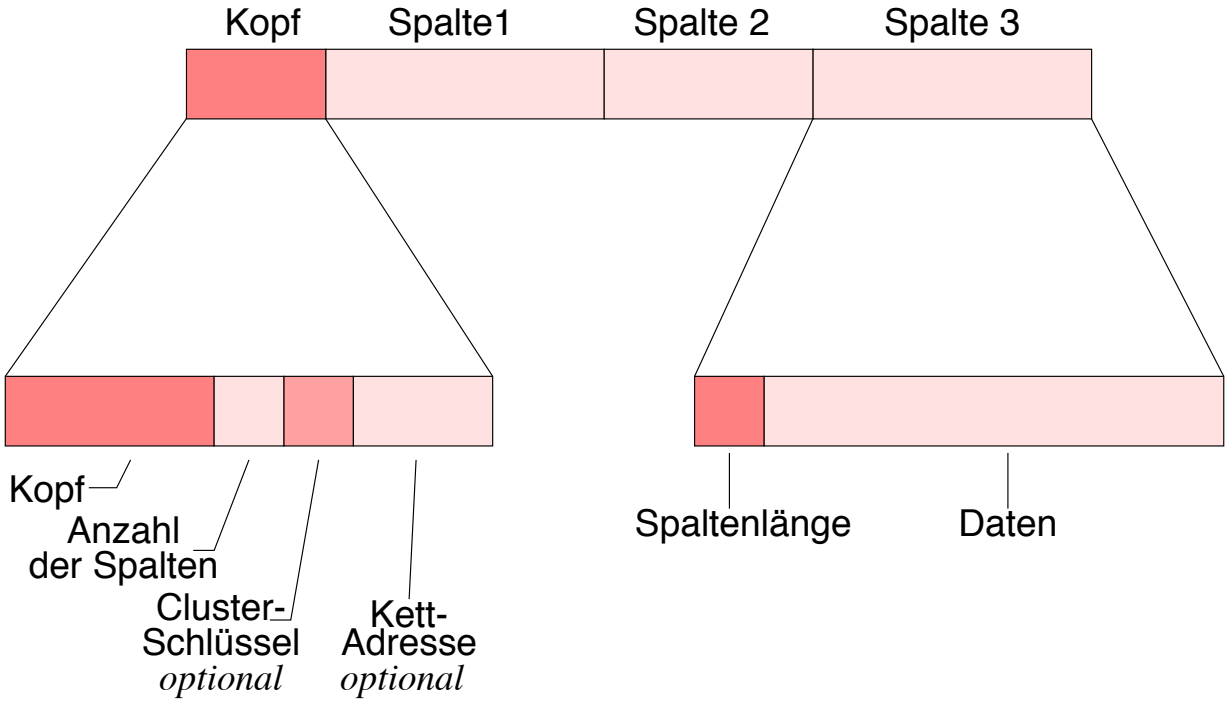


fest = 24 Byte

fest = 4 Byte für nicht geclusterte Tabellen

variabel - wächst mit Anzahl der Tupel

Oracle: Aufbau von Datensätzen



- Kettadresse für *Row Chaining*: Verteilung und Verkettung zu großer Datensätze ($>$ 255 Spalten) über mehrere Blöcke
- row id = (data object identifier, data file identifier, block identifier, row identifier)

Zusammenfassung (1)

- Speicherhierarchie und Zugriffslücke
- Speicher- und Sicherungsmedien
- Hintergrundspeicher: Blockmodell
- Pufferverwaltung: Seitensuche, Speicherzuteilung, Seitenersetzung
- Einpassen von Sätzen in Seiten
- Satzadressierung: TID-Konzept