Aufbau Datenbanksysteme

Lehrveranstaltung Datenbanktechnologien

Prof. Dr. Ingo Claßen Prof. Dr. Martin Kempa

Hochschule für Technik und Wirtschaft Berlin

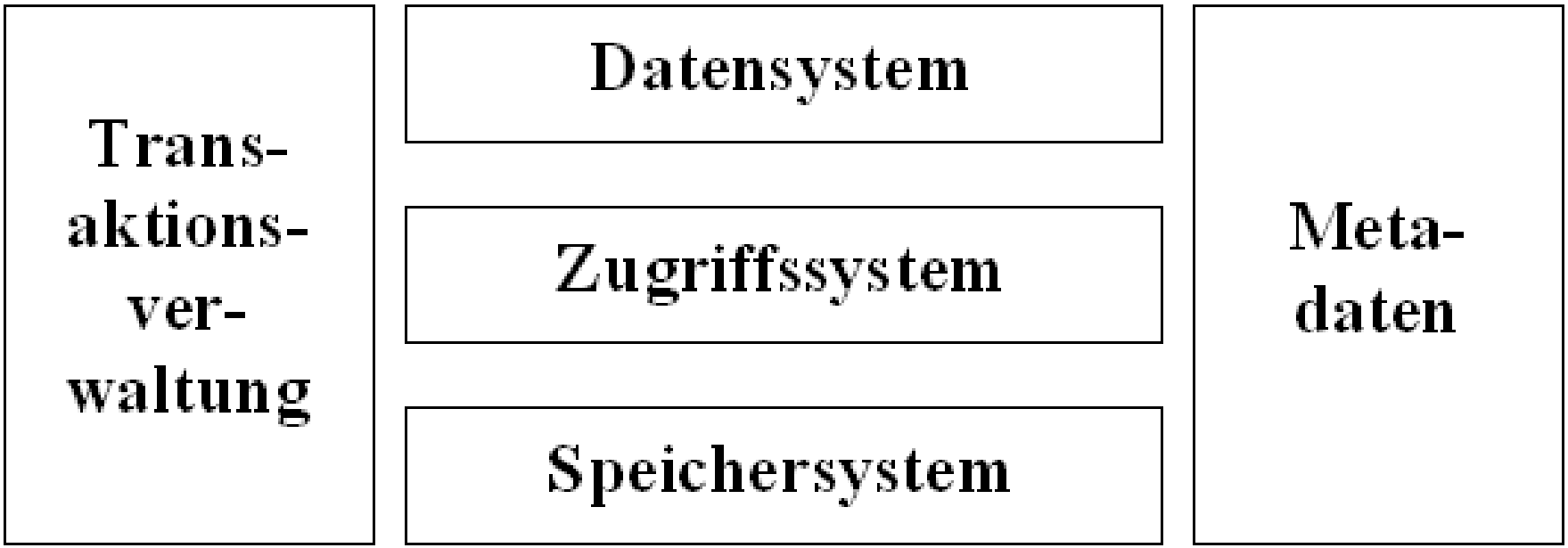
Speichersystem

Zugriffssystem

Datensystem

Metadaten

Softwarearchitektur eines Datenbanksystems

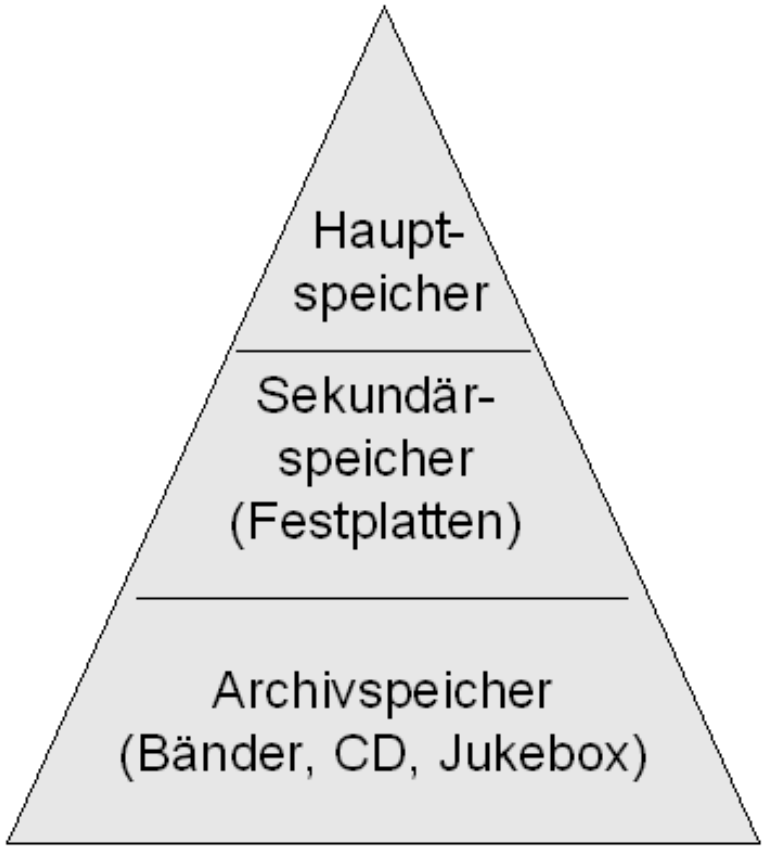


Speichersystem

- Aufgaben
  - Segmente und Seiten werden auf der Basis von Dateien und Blöcken realisiert
  - Seitenpuffer mit Seitenwechselstrategie wird realisiert
- Ziel

  Einsparen von Read-/Write-Operationen in der Datei
- Schnittstelle
  - Datenstrukturen: Segmente, die aus Seiten (page) bestehen
  - Operationen:
    - create/delete Segment
    - open/close Segment
    - write Seite in Segment/read Seite from Segment
    - release Seite in Segment

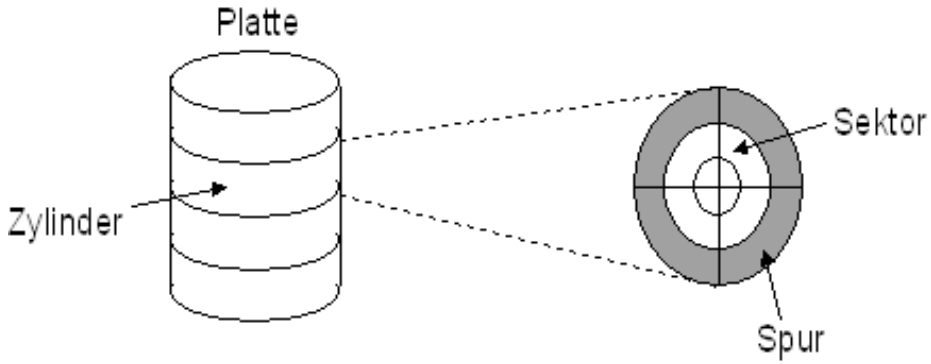
Speicherhierarchie



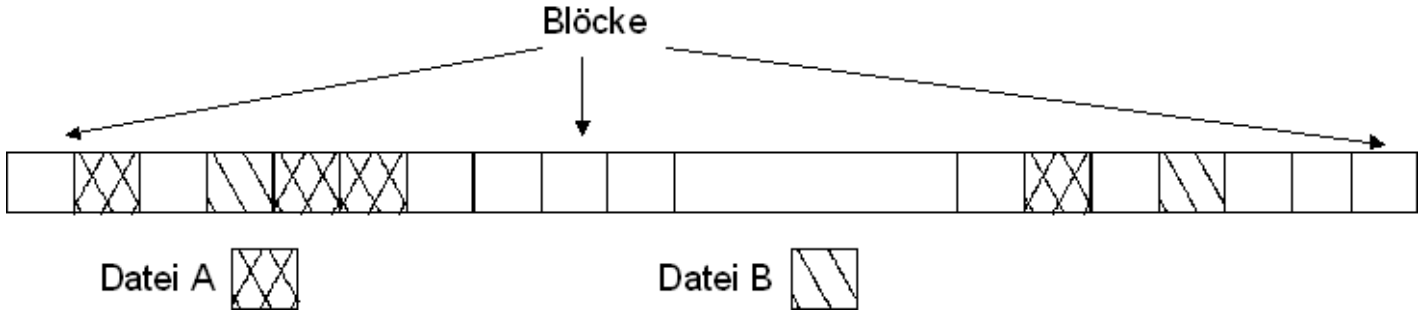
- schnell,
  relativ klein,
  flüchtig
- langsam,
  relativ groß,
  nicht flüchtig
- sehr langsam,
  sehr groß,
  nicht flüchtig

Exkurs: Dateien

- Platte

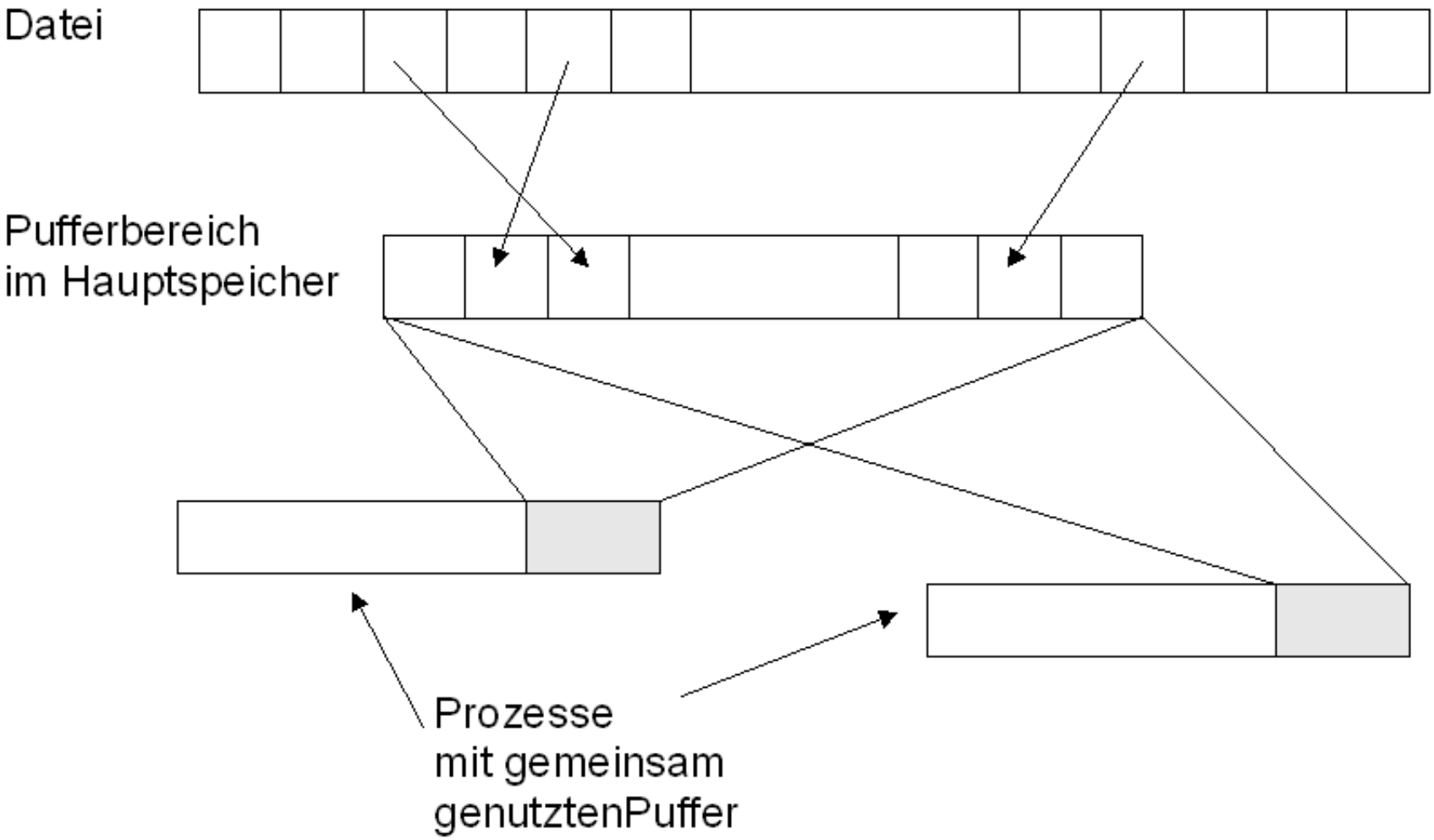


- Linearisierung (abrollen)



- Datei = Menge von Blöcken
  Abbildung: Blocknummer $\rightarrow$ (Zylinder, Spur/Kopf, Sektor/Block)

Seitenpuffer



Pufferverwaltung: Bereitstellen einer Seite

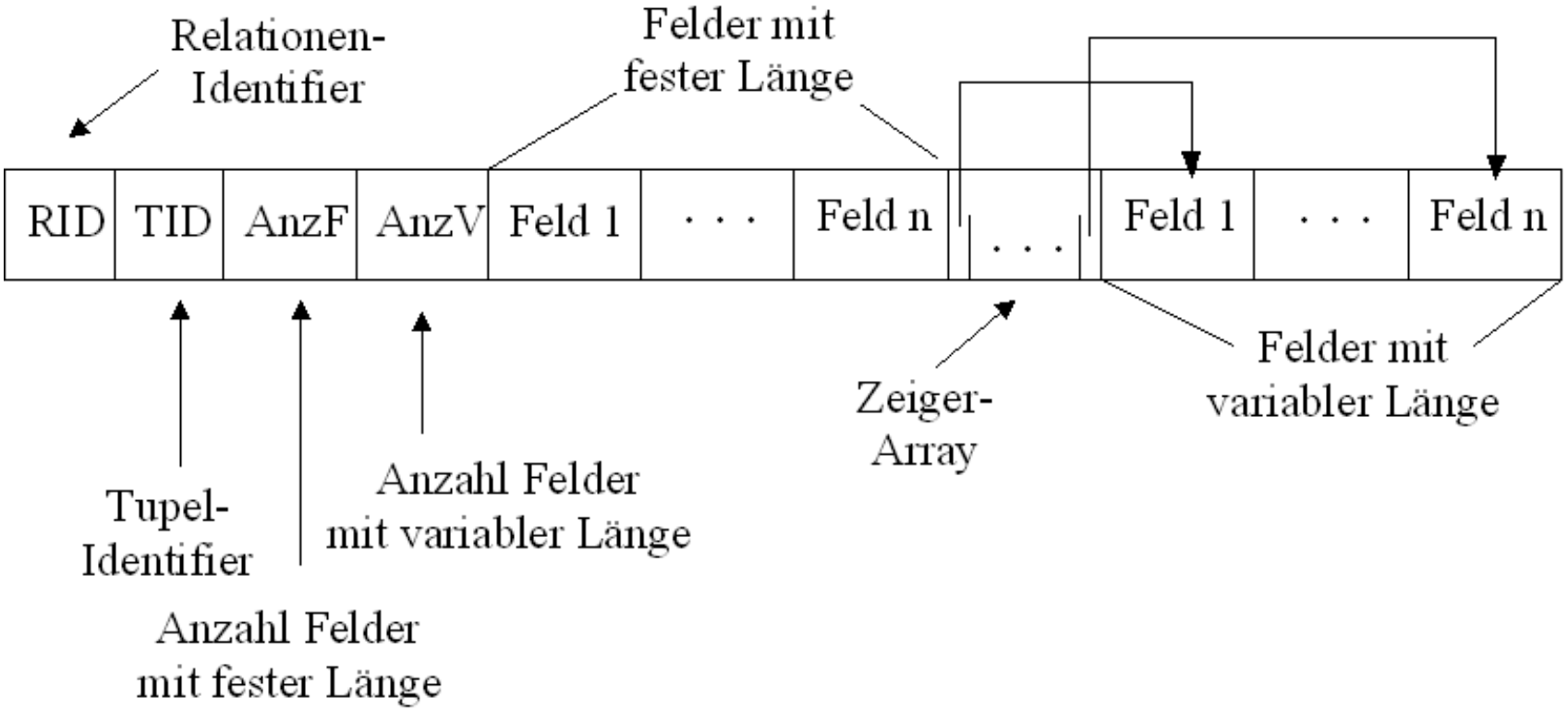
```
/* Seite s = (d,b), d. h. Block b aus Datei d */

if s ist im Puffer then
  return Pufferspeicheradresse für s
else
  suche freien Pufferplatz
  if kein freier Pufferplatz vorhanden then
    suche geeignete Seite zu entfernen aus dem Puffer
    schreibe Seite zurück auf Platte, falls Änderungen
    entferne Seite aus dem Puffer
  endif
  /* an dieser Stelle ist p der freie Pufferplatz */
  lies Block b aus Datei d in Pufferplatz p
  return Pufferspeicheradresse von p
endif
```

Zugriffssystem

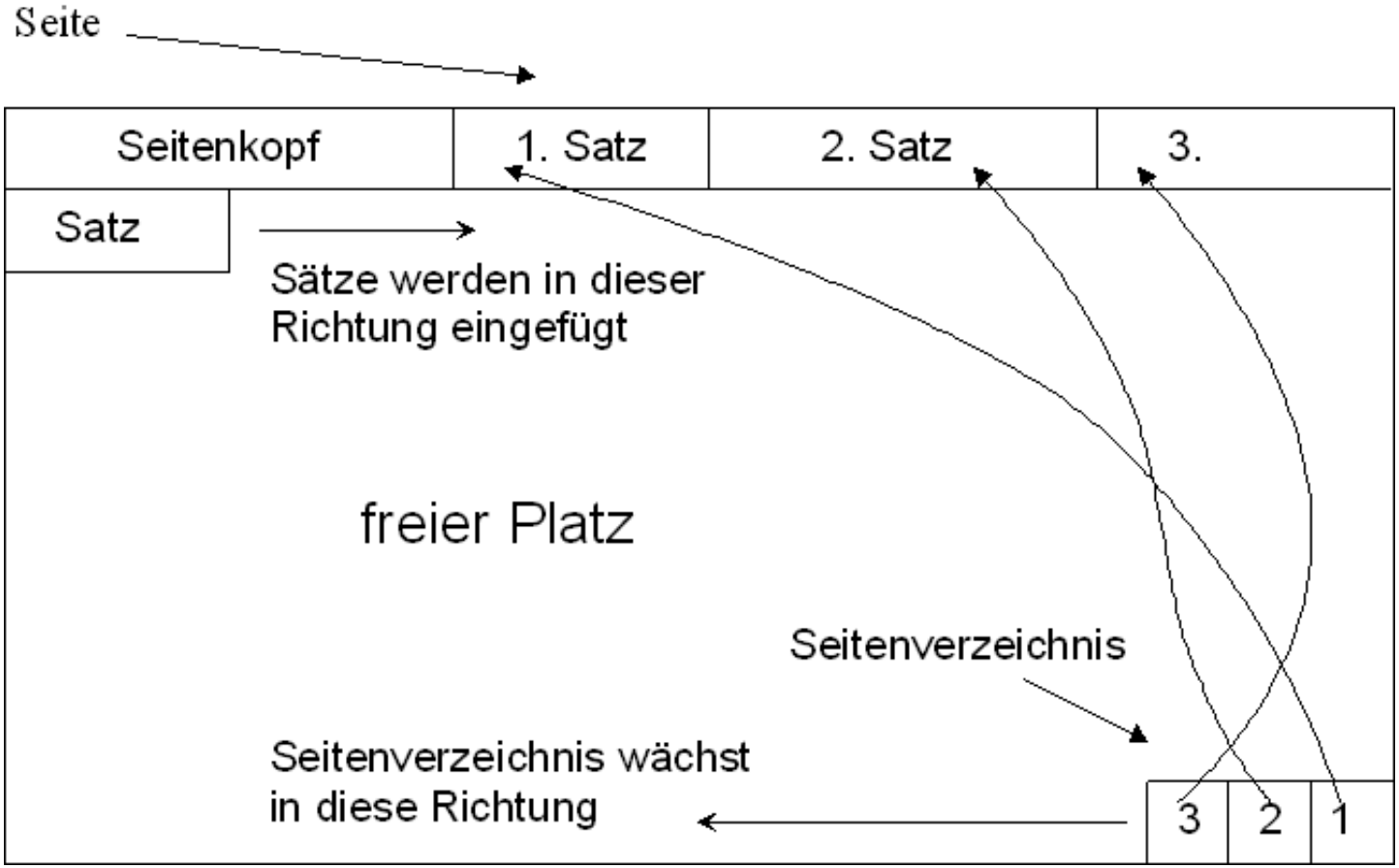
- Aufgaben
  - Freispeicherverwaltung
  - Record-Manager speichert interne Sätze
  - Zugriffspfadverwaltung realisiert geordnete Datenstrukturen.
  - Sperrverwaltung
  - Log/Recovery
- Ziel

  Persistente Speicherung interner Sätze und Realisierung von Zugriffspfaden für effizienten Zugriff
- Schnittstelle
  - Datenstrukturen: Satztypen mit internen Sätzen (Record), Bäume (altern. Hashtabellen)
  - Operationen:
    - create, update, delete satz
    - read satz by id
    - insert into B-Baum

Speicherstruktur für Sätze

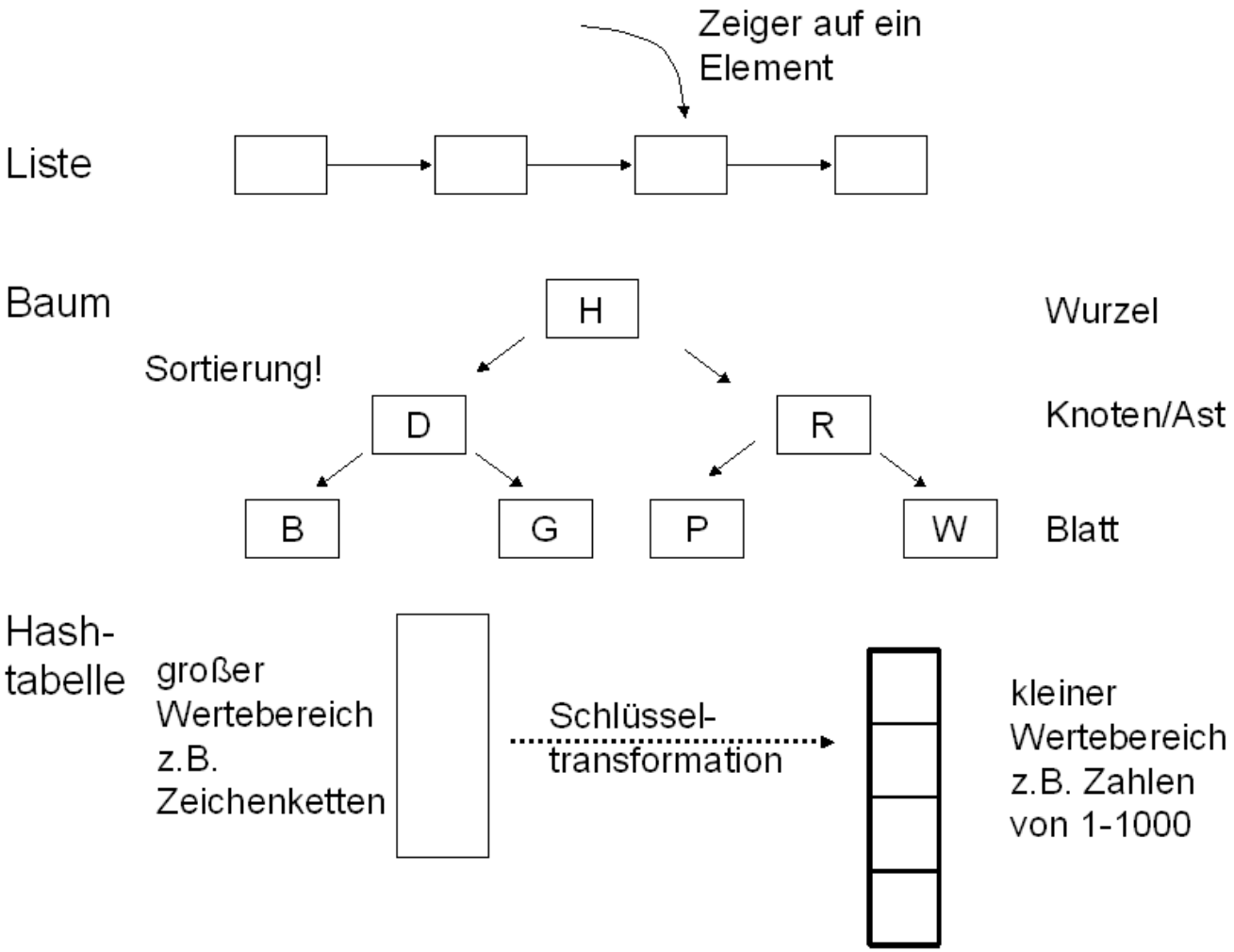


Verwaltung von Sätzen in Seiten



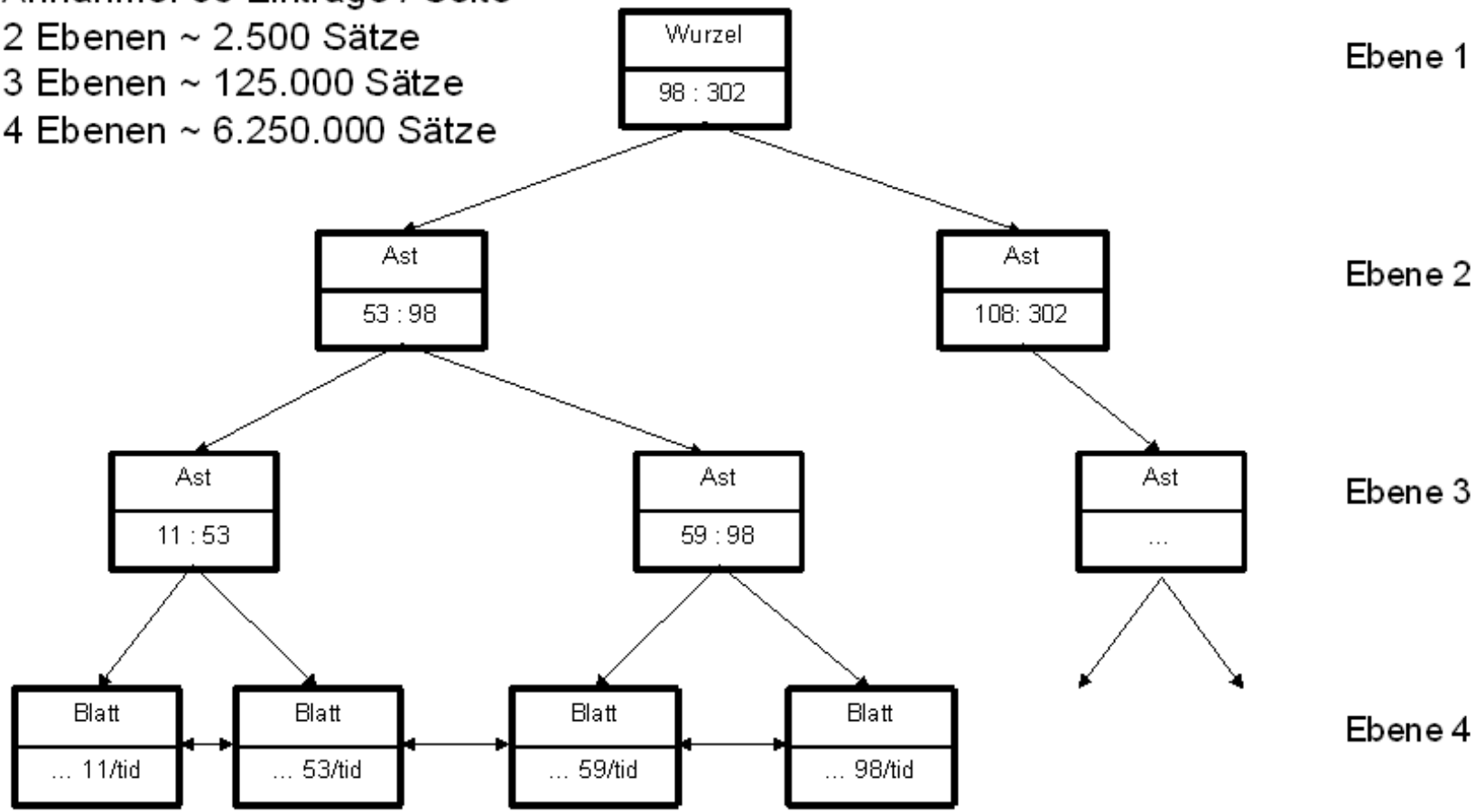
- Tuple-Identifier (TID)
  TID = (Seitennummer, Index im Seitenverzeichnis)

Exkurs: Datenstrukturen



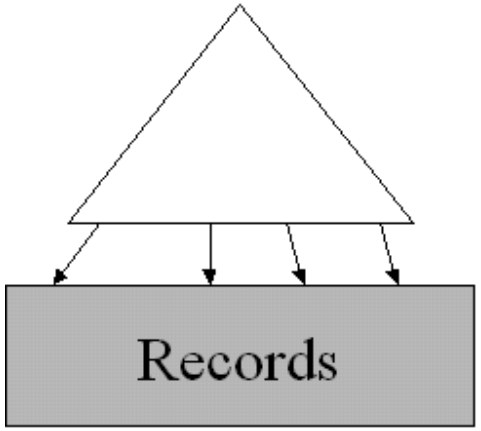
Zugriffspfad

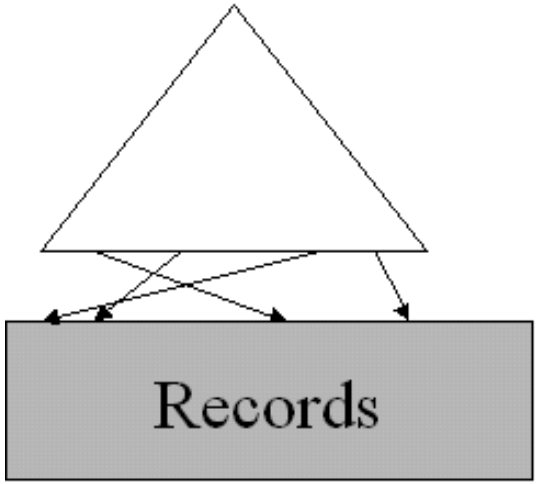
Annahme: 50 Einträge / Seite
2 Ebenen ~ 2.500 Sätze
3 Ebenen ~ 125.000 Sätze
4 Ebenen ~ 6.250.000 Sätze



Geclusterte/Nichtgeclusterte Indizes

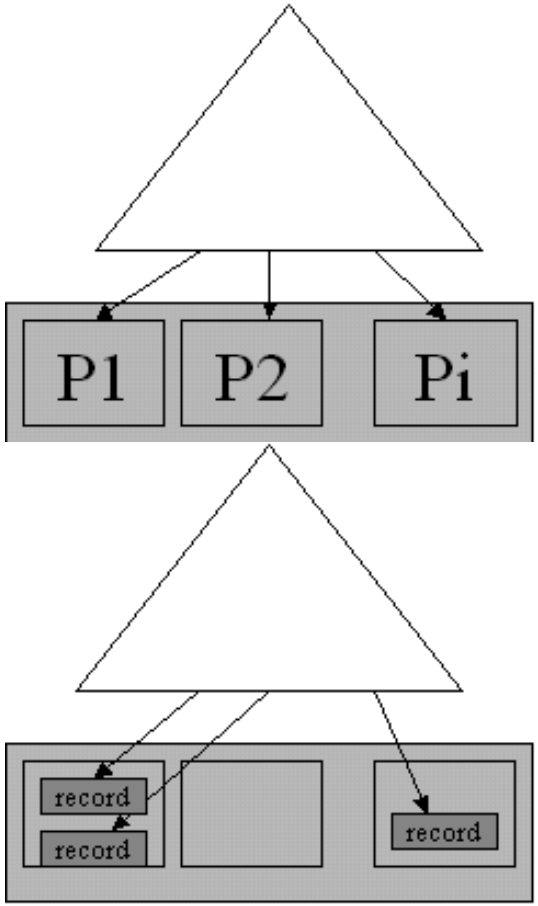
- Geclusterter Index
  - Physische Speicherung der Datensätze entsprechend der Indexsortierung
  - D. h. Datensätze, deren Indexwerte nahe bei einander liegen, werden physisch nahe bei einander gespeichert (auf der gleichen Seite)
  - Nur einer pro Tabelle
- Nichtgeclusterter Index
  - Physische Speicherung der Datensätze unabhängig von der Indexsortierung
  - Beliebig viele pro Tabelle





Dünn gestreute/Dichte Indizes

- Dünn gestreuter (sparse) Index
  - Zeiger auf Seiten
  - Suche nach Datensatz innerhalb der Seite
  - Benötigt wenig Speicher, dadurch wenige Seitenzugriffe, bei der Indexsuche
- Dichter (dense) Index
  - Direkter Zugriff über TID auf den Datensatz
  - Nichtgeclusterte Indizes müssen dicht sein



Weitere Aspekte von Indizes

- Überdeckende (covering) Indizes
  - Speicherung weiterer Attributwerte im Index
  - Vermeidung von Zugriffen auf Datenseiten, da alle benötigten Daten im Index vorliegen
- Kosten für Einfüge-, Lösch- und Änderungsoperationen
  - Bei Datenänderungen in den Tabellen müssen die Indizes angepasst werden
  - Wartung von Indizes
- Indizes sind kein Allheilmittel
  - Keine Vorteile bei kleinen Tabellen
  - TableScan kann schneller sein als IndexScan (abhängig von der Selektivität der Indexattribute)

Spezielle Zugriffspfade

- Bitmap-Indexstrukturen
  - Ein Bitvektor pro Spaltenwert, wobei jede Bitposition für einen Datensatz steht.
  - Steht an der Bitposition der Wert 1, so hat der entsprechende Datensatz den gegebenen Spaltenwert, ansonsten nicht.
- Multidimensionale Indexstrukturen
  - Erweiterung eindimensionaler Indexstrukturen
  - Addressierung von Datensätzen über mehrere Dimensionen

Datensystem

- Aufgaben
  - Data Dictionary
  - Currency Pointer
  - Sortierung
  - Transaktionsverwaltung
- Schnittstelle
  - Datenstrukturen: Satztypen externe Sätze, Index
  - Operationen:
    - insert/delete/update (externer) satz
    - find satz by id, (find satz by attributwert)
    - find satz by index key

Relationales Datensystem

- Aufgaben
  - Realisiert die relationalen Operationen (z. B. Join)
  - Zugriffskontrolle
  - Integritätskontrolle
  - Zugriffspfadauswahl
- Schnittstelle
  - Datenstrukturen: Zeilen, Relationen, Sichten
  - Operationen: SQL: select ... from ..., ...

Systemtabellen (Systemkatalog, Data Dictionary)

- Tabellen, die die Objekte innerhalb der Datenbank beschreiben
  - Tabellen
  - Spalten
  - Schlüssel
  - Indizes
  - Gespeicherte Prozeduren
  - Trigger
  - ...
- Werden für den laufenden Betrieb benötigt
  - Kontextanalyse bei Anfragen: Existieren die in der SQL-Anweisung angegebenen Tabellen und Spalten
  - Anfrageoptimierung: Welche Indizes existieren
  - Datenzugriff: Wo beginnt eine Spalte im Datensatz
  - Berechtigungsüberprüfung

Beispiel: Oracle-Server

- Tabellen

```
select T.table_name
from user_tables T
```

TableName
STUDENT
VERANSTALTUNG
STUDIENGANG
BEWERTUNG

- Spalten

```
select TC.column_name, TC.column_id
from user_tab_columns TC
where tc.table_name = 'STUDENT'
```

ColumnName	ColumnId
MATRNR	1
VORNAME	2
NAME	3
STUDIENGANG	4