

## 9.2: Die Fünf-Schichten-Architektur im Überblick

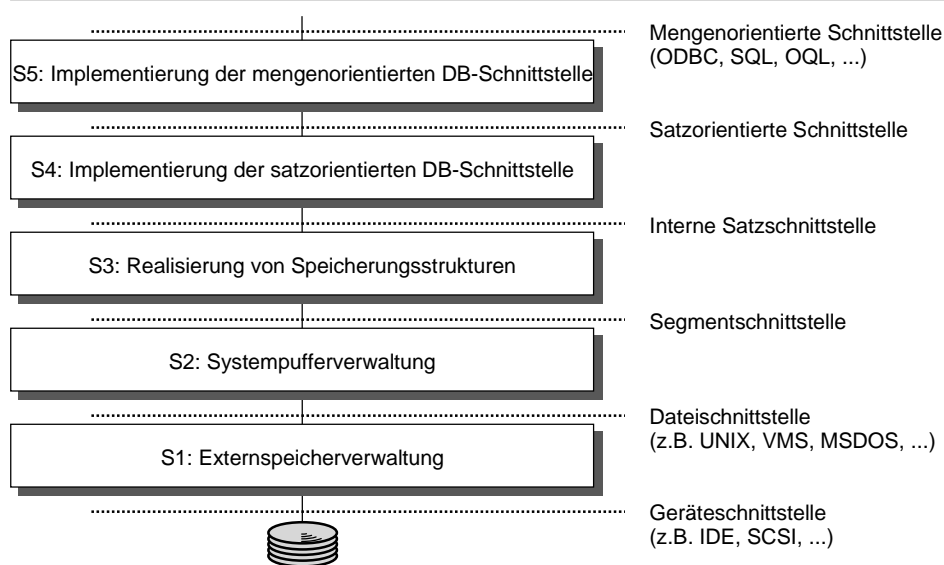
Lernziele:

- Aufgaben und Objekte der sechs Schnittstellen
- Teilaufgaben der Schichten
- Ausgewählte Lösungsansätze (je 1 Konzept pro Schicht)

Datenbanken und Informationssysteme

Die Fünf-Schichten-Architektur 9.2.1

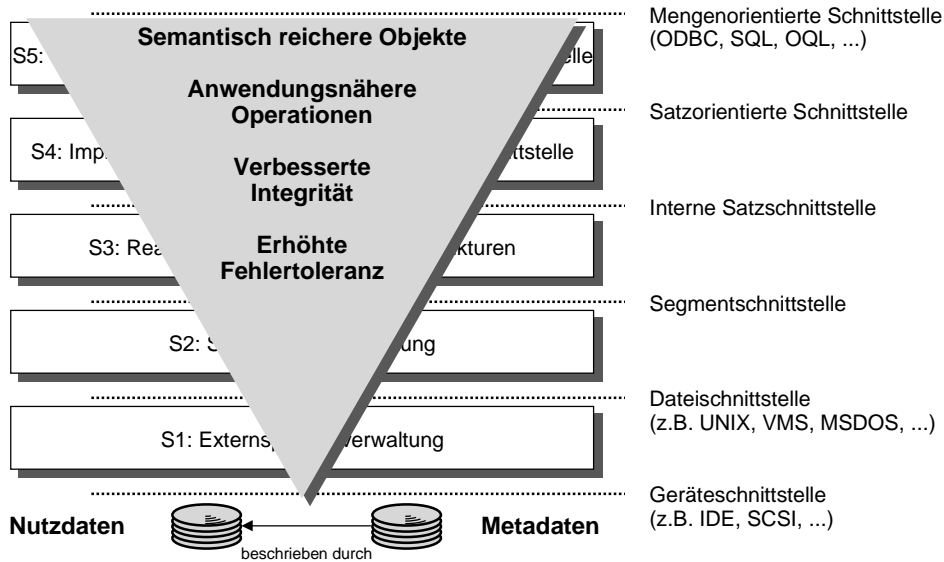
### Fünf-Schichten-Architektur von Senko (1)



Datenbanken und Informationssysteme

Die Fünf-Schichten-Architektur 9.2.2

## Fünf-Schichten-Architektur von Senko (2)



Datenbanken und Informationssysteme

Die Fünf-Schichten-Architektur 9.2.3

## S1: Externspeicherverwaltung

### Aufgabe und Objekte:

- Realisierung von Dateien bestehend aus Blöcken auf physischen Geräten
  - Einfaches, sicheres, stabiles Schreiben (read/write vs. memory mapped) zum Datentransport zwischen Systempuffer und Hintergrundspeicher
  - Direkter Zugriff für Nutzdaten, sequentieller Zugriff für Protokollaten

### Teilaufgaben:

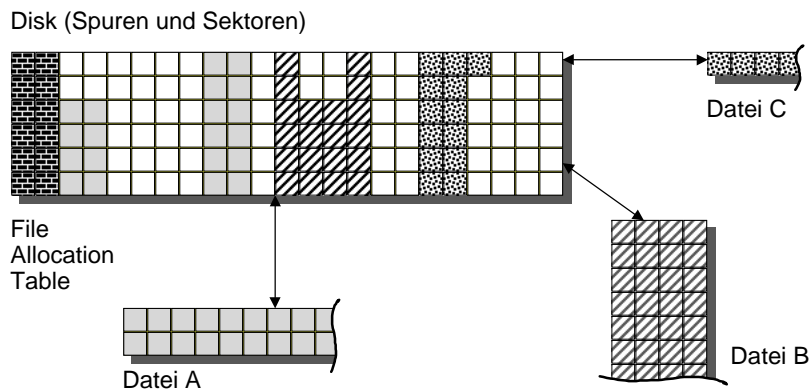
- Verwalten des physischen Speichers auf externen Speichermedien (file allocation tables, directory tables, ...) dabei Verbergen von Speicherparametern
- Blockadressierung: Zuordnung der physischen Blöcke zu Untereinheiten externer Speichermedien (Zylinder, Spuren)
- Einsatz von Spiegelplatten und fehlertoleranter Hardware (RAID)
- Evtl. mehrstufige Speicherungshierarchie: nicht-flüchtiges RAM, Cache im Platten-Controller, Festplatte, Bandroboter
- Optional: Speicherabzug (*Dump*), Verschlüsselung, Entfernter Plattenzugriff, ...

Datenbanken und Informationssysteme

Die Fünf-Schichten-Architektur 9.2.4

## S1: Externspeicherverwaltung

Beispiel: Abbildung einer Datei auf ein einzelnes physisches Gerät



Metadaten (File Allocation Table) und Nutzdaten (Datei X)

## S2: Systempufferverwaltung <sup>(1)</sup>

**Aufgabe und Objekte:**

- Realisierung eines oder mehrerer *Segmente* bestehend aus *Seiten* fester Größe mit sichtbaren Seitengrenzen
- Seiten können in *Systempufferrahmen* (Slots) des *Systempuffers* durch Hauptspeicherzugriffe von *nebenläufigen* Transaktionen gelesen und modifiziert werden
- Segmente sind lineare, unendliche Adreßräume, die verschiedene Charakteristika besitzen können (typisch persistent, typisch mehrbenutzerfähig, typisch fehlererholend)

Motivation für eine *indirekte* Seitenadressierung und *indirekte* Einbringstrategie

- Vermeide "update in place" (unmittelbare, direkte Änderung von Seiten auf Platte)
  - Vereinfachtes Rücksetzen von Transaktionen
  - Erhöhte Parallelität bei lesenden und schreibenden Transaktionen auf der gleichen Seite

## S2: Systempufferverwaltung (2)

### Teilaufgaben:

□ S2a: Systempufferverwaltung

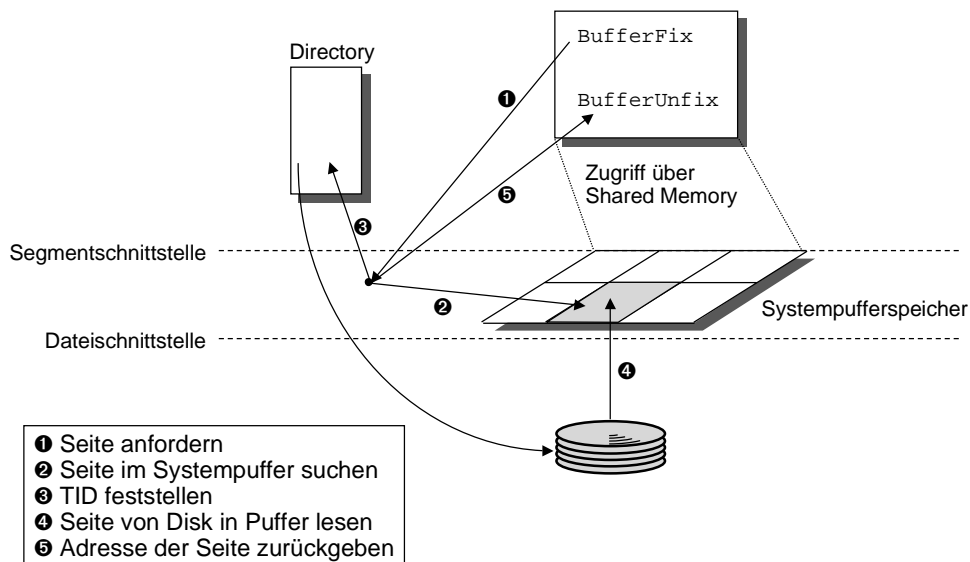
- Effizientes Auffinden einer Seite im Puffer
- Speicherzuteilung im Systempuffer (*fix, unfix, dirty pages*)
- Ersetzungsstrategien für Seiten (Anzahl der Zugriffe, Zeitpunkt der Zugriffe)
- Probleme bei der Verwaltung des Systempuffers (*demand paging, prefetching, preplanning*)
- Nutzung von Anwendungswissen (*hot set model, Prioritätssteuerung*)
- Pufferverwaltung bei variabler Seitengröße

□ S2b: Seitenzuordnung von Segmenten und Seiten auf Dateien und deren Blöcken

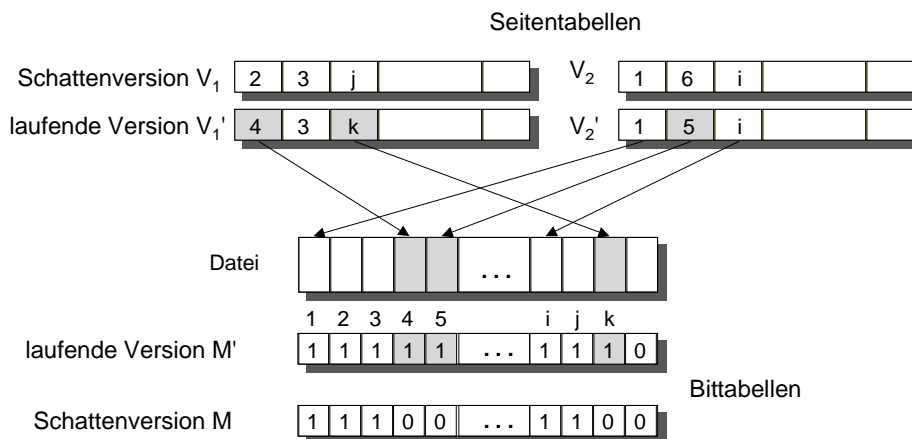
- "Twin-Slot"-Verfahren, *Schattenspeicherkonzept*, Zusatzdatei-Konzept

Problematisch: Interaktion der (virtuellen) Speicherverwaltung des Betriebssystems und der Systempufferverwaltung des DBMS.

## S2a: Systempufferverwaltung (3)



## S2b: Schattenspeicherkonzept (1)



## S2b: Schattenspeicherkonzept (2)

**Idee:** Der Inhalt aller Seiten eines Segmentes bleibt während eines Sicherungsintervalls unverändert in einem konsistenten Zustand.

Geänderte Seiten werden immer in freie Blöcke geschrieben.

- Konsistente Zustände werden durch Sicherungspunkte gekennzeichnet.
- Ein Sicherungspunkt umfaßt die Seitentabelle  $V$  und die Bitliste  $M$ .
- Seitentabellen bilden Seiten auf Dateiblöcke ab.
- Die Bittabelle gibt für jede Seite an, ob sie frei oder belegt ist.
- Alle Änderungen werden mit Hilfe von Kopien der Seiten- und Bittabellen durchgeführt ( $V'$  und  $M'$ ). Nach der Ausführung wird
  - im Fehlerfall auf die ursprünglichen Versionen  $V$  und  $M$  zurückgegriffen,
  - ansonsten wird  $V'$  zu  $V$  und  $M'$  zu  $M$  kopiert.
- Ein Flag zeigt die aktuelle Version an.

## S2b: Schattenspeicherkonzept (3)

---

### Bewertung:

- + Einfachen Rücksetzen auf einen konsistenten Zustand im letzten Sicherungspunkt.
- + Bei katastrophalen Fehlern, bei dem die Log-Dateien zerstört wurden, ist die Wahrscheinlichkeit, mit Hilfe der Schattenspeicher einen "brauchbaren" Zustand zu rekonstruieren, recht groß im Vergleich zu direkt modifizierten Datenbanken.
- Keine Clusterbildung auf Seitenebene

## S3: Realisierung von Speicherungsstrukturen (1)

---

### Aufgabe und Objekte:

- Bereitstellung von *physischen* Speicherungsstrukturen (vgl. ANSI/SPARC)
  - S3a: *Rekordmanager*: Abbildung der physischen *Rekords* teilweise dynamischer Größe (z.B. 100 Bytes) auf Seiten und Systempufferrahmen (z.B. 8K)
  - S3b: *Zugriffspfadverwaltung*: Aktualisierung und Verwendung von spezifizierten physischen Zugriffspfaden (z.B. Verkettung aller Rekords einer Relation, Primärschlüsselzugriff und Sekundärschlüsselzugriff)

Die Zugriffspfadverwaltung sitzt innerhalb der Schicht häufig "oberhalb" des Rekordmanagers und verwaltet nur Referenzen auf Rekords. Aus Effizienzgründen werden die Zugriffspfadstrukturen selbst aber auch direkt auf Seiten abgebildet.

Beispiel: B-Bäume, Hashtabellen

## S3a: Teilaufgaben des Rekordmanagers

---

- Freispeicherverwaltung innerhalb von Segmenten und Seiten
- Adressierung von Sätzen innerhalb von Seiten
  - Externspeicherbasierte Adressierung (*TID-Konzept, DBK-Konzept*) oder
  - Hauptspeicherbasierte Adressierung (OID = Hauptspeicheradressen) in modernen objektorientierten Systemen (→ *Swizzling-Verfahren* zur Adreßumsetzung wg. Persistenz)
- Abbildung von Datensätzen in Seiten
  - Speicherung von "flachen" Tupeln (HDM, NDM, RDM)
  - Darstellung komplexer Objekte (ODDM, teilweise NDM): Mengen, Listen, Bags
  - Unterstützung der Clusterbildung für komplexe Objekte
    - Objektbezogene Cluster
    - Objektübergreifende Cluster
  - Realisierung "langer Felder" (Rekords dynamischer Größe > Seitengröße)

## S3a: Rekordmanager - Freispeichertabelle

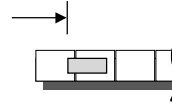
---

- Eine Freispeichertabelle F pro Segment mit der Anzahl freier Bytes (oder freier n Bytes) pro Seite
- F ist entweder als Teil des Segmentes (bei direkter Adressierung) oder der Seitentabelle (bei indirekter Adressierung).
- Ein Eintrag pro Seite über Beginn und aktuelle Länge des freien Seitenanteils

## S3a: Rekordmanager - Satzadressierung

---

- Sehr wichtige Implementierungsentscheidung
- Anforderungen an Adressierungstechnik:
  - Schneller, möglichst direkter Satzzugriff
  - Hohe "Stabilität" gegen "kleine" Verschiebungen



### Alternativen :

- Direkte Adressierung
  - Relative Byteadresse ab Segmentanfang und schneller Zugriff
  - Inflexibel gegen Satzverschiebung
  - Zwang zu häufiger Reorganisation
  - Für DBMSs ungeeignet
- Indirekte Adressierung
  - Abbildung der logischen auf die physischen Adressen mittels Tabellen

## S3a: Rekordmanager - TID-Konzept (1)

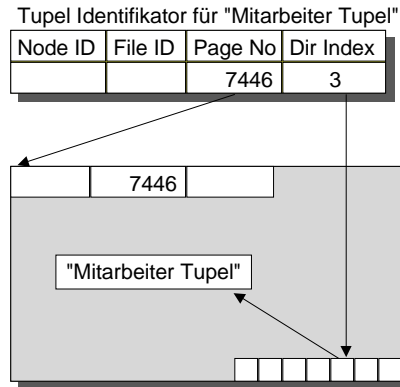
---

### TID-Konzept (Tuple Identifier):

- Die Tabelle zur Abbildung von Rekords innerhalb von Seiten ist auf die einzelnen Seiten verteilt, d.h. lokale Lösung der Überlaufproblematik.
- Zugriffsfaktor ca. 1.1 bei 10% Überlauf
- Entwickelt für System R (vgl. Astrahan 1976)
- Beispiel s. nächste Folie

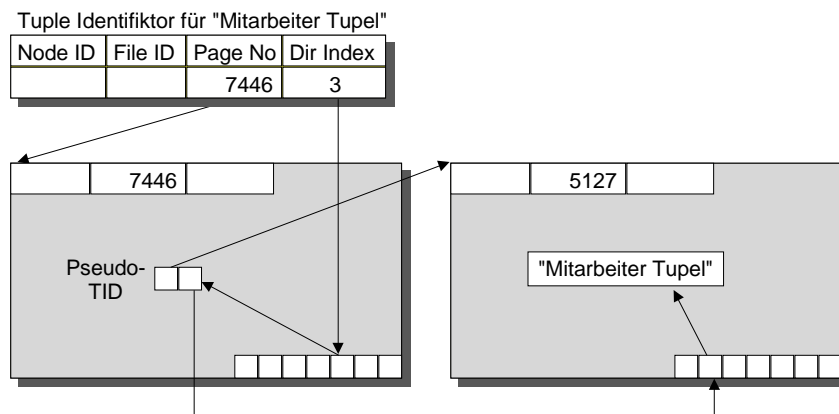
### S3a: Rekordmanager - TID-Konzept (2)

- ❑ 1. Fall: Der Tupel befindet sich in der Seite, auf die der TID zeigt.



### S3a: Rekordmanager - TID-Konzept (3)

- ❑ 2. Fall: Das Tupel "Mitarbeiter Tupel" ist in Folge einer Modifikation auf die Seite 5127 verschoben worden. Die ursprüngliche TID bleibt unverändert.



## S3b: Zugriffspfade - Einbettung oder Trennung

---

### 1. Alternative:

- Einbetten der Zugriffspfadstrukturen in die Speicherungsstrukturen der Sätze, etwa in Form physischer Nachbarschaft oder von Zeigerverkettung
  - Folge: Enge Zusammenarbeit von Rekordmanager und Zugriffspfadverwaltung
  - Konsequenz für die Praxis: Verwaltung und Wartung von Zugriffspfaden muß vom Rekordmanager übernommen werden.

### 2. Alternative:

- Separate Speicherung der Zugriffsinformation und Verweis auf die Sätze durch geeignete Adressierungstechnik. Diese Zugriffsinformation wird von der Zugriffspfadverwaltung angelegt, ausgewertet und gepflegt.

## S3b: Zugriffspfadverwaltung: Eindimensional <sup>(1)</sup>

---

### Zugriffspfade für Primärschlüssel

- Sequentielle Speicherungsstrukturen: Schlüsselwert → Rekordadresse / NIL  
Sequentielle Listen, verkettete Listen
- Baumstrukturierte Speicherungsstrukturen: Binärbäume, Mehrwegbäume als *B-Bäume*, Mehrwegbäume als *B\*-Bäume*, Digitalbäume
- Statische Hash-Verfahren: kollisionsfreie Satzzuordnung, Hash-Verfahren mit Kollisionsbehandlung, externes Hashing mit Separatoren
- Dynamische Hash-Verfahren: Verfahren mit Indexnutzung, Verfahren ohne Indexnutzung

## S3b: Zugriffspfadverwaltung: Eindimensional (2)

### Zugriffspfade für Satzmengen

- Verknüpfungsstrukturen für Satzmengen
- Zugriffspfade für Sekundärschlüssel: Implementationsformen der Invertierung, Erweiterung der Invertierungsverfahren
- Hierarchische Zugriffspfade

Wert → Liste(Rekordadresse)

### Zugriffspfade über mehrere Relationen

- Verallgemeinerte Zugriffspfadstrukturen
- Verbund-Index
- Mehrverbund-Index, Pfad-Index

Rekordadresse → Liste(Rekordadresse)

```
select x.company.products
from x in X where ...
```

## S3b: Zugriffspfadverwaltung: Mehrdimensional

### Mehrattributzugriff über mehrere eindimensionale Zugriffspfade

- Separate Attribute als Schlüssel
- Konkatenierte Attribute als Schlüssel

Wert, Wert, ... Wert → List(Rekordadresse)

### Mehrdimensionale Zugriffspfade für punktförmige Objekte

- Organisation der Datensätze (Quadranten-Baum, Mehrschlüssel-Hashing, Mehrdimensionale binäre Suchbäume)
- Organisation des umgebenen Datenraums durch Divide and Conquer bei lokaler Ordnungserhaltung (*Heterogener k-d-Baum, k-d-B-Baum, hB-Baum*)
- Organisation des umgebenen Datenraums durch Dimensionsverfeinerung (Grid-File, Interpolationsbasiertes Grid-File, Mehrdimensionales dynamisches Hashing)

### Zugriffspfad für ausgedehnte räumliche Objekte (*R-Baum, R+-Baum*)

## S3: B-Bäume (1)

---

### Definition:

- Ein B-Baum vom Typ  $(k, h)$  mit Füllungsgrad  $k$  und Baumhöhe  $h$  ist ein Baum mit folgenden drei Eigenschaften:
  - Jeder Weg von der Wurzel zu einem Blatt hat die gleiche Länge  $h$ .
  - Alle Knoten (außer Wurzel und Blätter) haben mindestens  $k+1$  Kinder. Die Wurzel ist entweder ein Blatt oder sie hat mindestens zwei Kinder.
  - Alle Knoten haben höchstens  $2k+1$  Kinder.
- Vgl. Bayer, McCreight, 1972

## S3: B-Bäume (2)

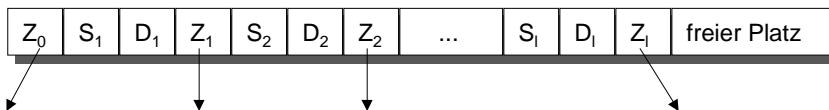
---

### Implementierung:

- 1:1-Zuordnung zwischen Knoten und Seiten (Knoten "sind" Seiten; Baumhöhe  $h$  identisch Zahl der Seitenzugriffe).
- Interne Seitenstruktur:
  - (Schlüssel, Datum, Zeiger) geordnet nach aufsteigenden Schlüsselwerten. Sei  $n$  die max. Zahl der Einträge pro Seite, (also  $k = n / 2$ ), so gilt:
    - Jede Seite ist höchstens voll (!?).
    - Jede Seite (außer der Wurzel) ist mind. halb voll.
    - Ein B-Baum ist vollständig balanciert.
  - Beispiel s. nächste Folie

### S3: B-Bäume (3)

Format einer Seite eines B-Baums ( $k \leq l \leq 2k$ ):



$(S_i, D_i, Z_i) = \text{Eintrag}$

$S_i = \text{Schlüssel}$

$D_i = \text{Daten des Satzes oder Verweis auf den Satz (typisch!)}$

$Z_i = \text{Zeiger zu einer Kinderseite}$

### S3: B-Bäume (4)

**Bedeutung der Zeiger:**

- $Z_0$  verweist auf Teilbaum mit Schlüssel  $n$  kleiner  $S_1$
- $Z_i$  verweist auf Teilbaum mit Schlüssel  $n$  zwischen  $S_i$  und  $S_{i+1}$  ( $i = 1, 2, \dots, l-1$ )
- $Z_l$  verweist auf Teilbaum mit Schlüssel  $n$  größer  $S_l$
- In den Blättern sind die Zeiger undefiniert (fehlen).

Für die Höhe  $h$  eines B-Baumes, der  $N$  Datenelemente verwaltet, gilt also für  $N \geq 1$ :

$$\log_{2k+1}(N + 1) \leq h \leq 1 + \log_{k+1}((N + 1) / 2)$$

## S3: B-Bäume (5)

Beispiel (s. nächste Folie):

- Das Zeichen "•" symbolisiert entweder den Datensatz selbst oder einen Zeiger auf ihn:
  - Satztyp ABT: Abteilungsdaten
  - Primärschlüssel ANR: Abteilungsnummer  $ANR \in \{ K01, K02, \dots, K99 \}$
  - B-Baum vom Typ  $(k = 2, h = 3)$

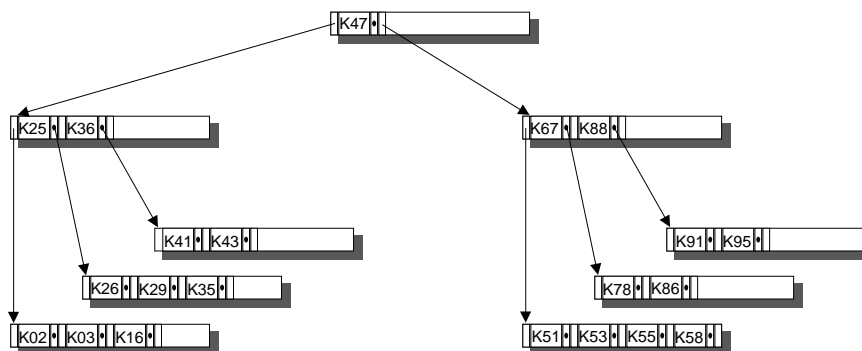
Entwurfsziel: Hoher Verzweigungsgrad (fan out) und damit niedrige Baumhöhe.

Beispiel:

- Seitengröße 4KByte, Zeiger- und Schlüssellänge 4 Byte, Datenlänge 92 Byte
- Alternative 1: Speicherung der Daten im Baum ergibt Verzweigungsgrad von 40;
- Alternative 2: Speicherung von Zeigern auf die Daten ergibt Verzweigungsgrad von über 330.

## S3: B-Bäume (6)

**B-Baumstruktur als Zugriffspfad für den Primärschlüssel ANR:**



## S3: B\*-Bäume (1)

### Unterschiede zum B-Baum:

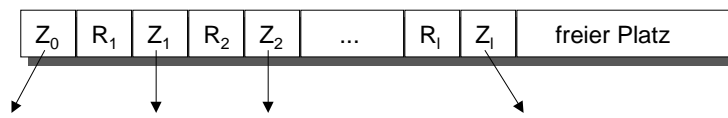
- ❑ Redundante Schlüsselspeicherung, Schlüssel in den bauminternen Knoten haben nur organisatorische Aufgaben
- ❑ Keine Dateneinträge in baum-internen Knoten
- ❑ Konsequenzen:
  - Höherer Verzweigungsgrad und geringere Höhe
  - Zwei unterschiedliche Knotenformate mit dem Ziel der verbesserten Unterstützung der *sequentiellen Verarbeitung* durch Blattverkettung.
- ❑ Ausgezeichnete Stellung der Blattknoten:
  - Nur sie enthalten Dateneinträge, D;
  - Für ihren Füllungsgrad gilt:  $k^* \leq j \leq 2k^*$ ;  $k^*$  ist im allg. verschieden von  $k$ .

Es gibt also zwei unterschiedliche Knotenformate in einem B\*-Baum vom Typ  $(k, k^*, h)$  (s. Abbildung nächste Folie)

## S3: B\*-Bäume (2)

### Knotenformate (Seitenformate) eines B\*-Baums:

Innerer Knoten:



$R_i = \text{Referenzschlüssel } k \leq l \leq 2k$

Blattknoten:



$P = \text{PRIOR-Zeiger, } N = \text{NEXT-Zeiger, } k^* \leq j \leq 2k^*$

Die Zeiger P und N dienen der Blattverkettung zur Unterstützung der sequentiellen Verarbeitung .

### S3: B\*-Bäume (3)

Höhe eines B\*-Baumes:

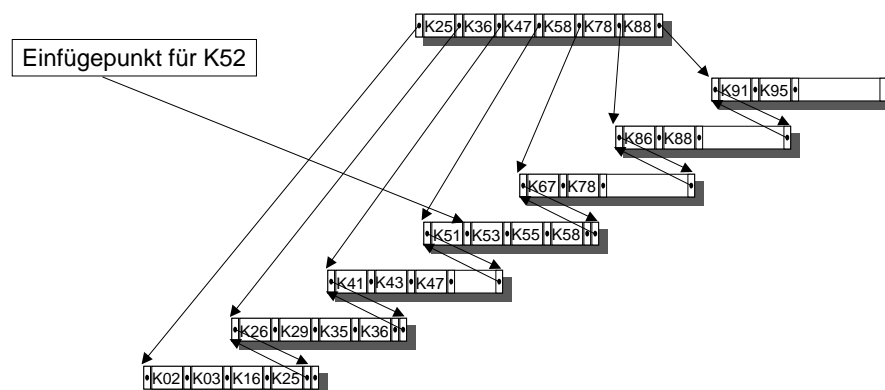
$$1 + \log_{2k+1} (N / 2k^*) \leq h \leq 2 + \log_{k+1} (N / 2k^*) \text{ für } N \geq 2$$

Beispiel:

- Verzweigungsgrad > 500.
- Bei  $10^5$  bis  $10^7$  Einträgen  $N$  ergibt sich  $h$  die Zahl der Seitenzugriffe bzw. Baumhöhe zu 3 bis 4.
- Lesen: Baumtraversierung;
- Einsetzen, Löschen: s. Abbildung nächste Folien
  - **Überlauf:** Seitenteilung, d.h. Leerseitenanforderung, Modifikation der Elternseite, evtl. rekursive Fortsetzung nach oben.
  - **Unterlauf:** Seitenverschmelzung, d.h. Leerseitenabgabe, Modifikation der Elternseite, evtl. rekursive Fortsetzung nach oben.

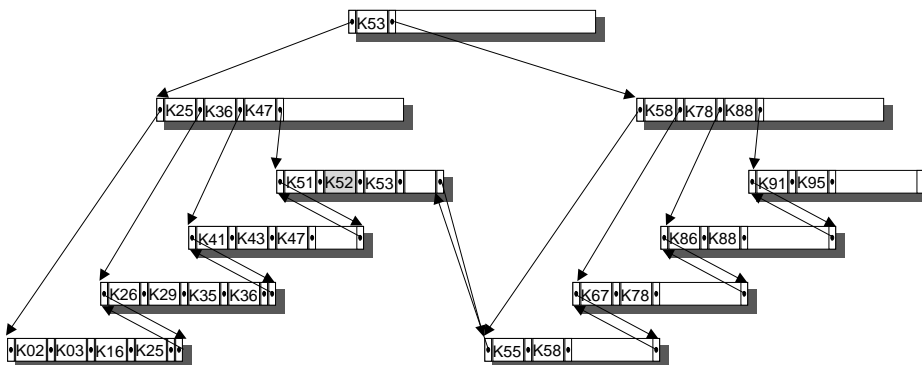
### S3: B\*-Bäume (4)

B\*-Baum vom Typ  $(k = 3, k^* = 2, h = 2)$ :



### S3: B\*-Bäume (5)

B\*-Bäume nach Einfügen von K52: ( $k=3, k^*=2, h=3$ ):



### S4: Die satzorientierte Schnittstelle (1)

**Aufgaben:**

- Realisierung von *logischen Sätzen* und *logischen Zugriffspfaden* auf physische Sätze und physische Zugriffspfade
- Bereitstellen der *Typinformationen* für die logischen Sätze einschließlich aller im *Datenwörterbuch* abgelegten Beschreibung
- Bereitstellen der *Arbeitsumgebung* (*Aktualitätsanzeiger*, *Statusinformation*, *kontrollierte Ausnahmebehandlung*)
- Bereitstellen generischer Funktionen auf Rekordattributen (*Sortierung*, *Mittelwert*, *Extremwert*, ...)
- Transaktionsunterstützung (BEGIN WORK, COMMIT WORK, ROLLBACK WORK) für Anwendungsprozesse und Datenbankmonitor

## S4: Die satzorientierte Schnittstelle (2)

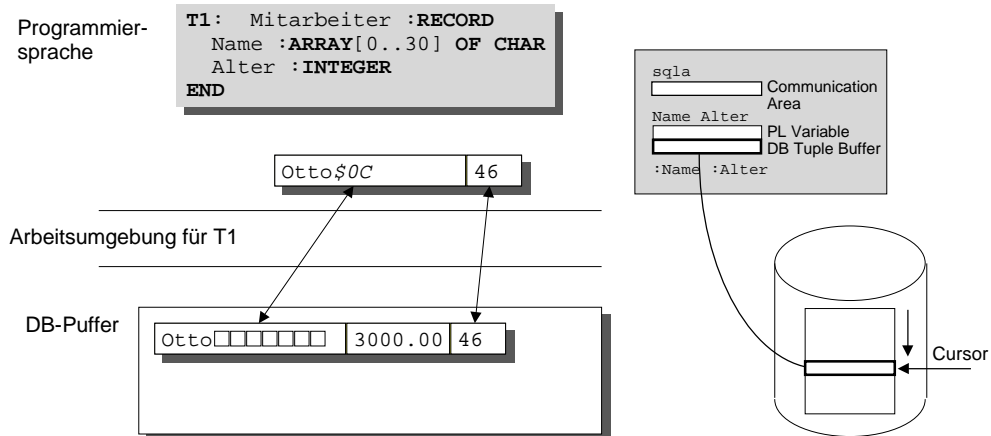
### Teilaufgaben:

- ❑ Pflege des Datenwörterbuchs (*data dictionary*), das einen uniformen lesenden und schreibenden Zugriff auf alle logischen und internen Schemainformationen (Namen, Besitzer, Gerätezuordnung, Zugriffsrechte, Änderungsdatum, ...) liefert.
- ❑ Abbildung der externen Sätze auf interne Sätze
  - Transformation zwischen den Typmechanismen von Programmiersprache und Datenbank
  - Umordnen und Ausblenden von Attributen
  - Realisierung redundanter und komprimierter Speicherung
  - Unterstützung der Evolution externer Typen
- ❑ Wartung aller durch das Datenwörterbuch definierten Zugriffsstrukturen
- ❑ Currency-Konzepte zur satzweisen Navigation
- ❑ Sitzungsverwaltung und Koordination der (Mehrschichten-)Transaktionsverwaltung

Datenbanken und Informationssysteme

Die Fünf-Schichten-Architektur 9.2.35

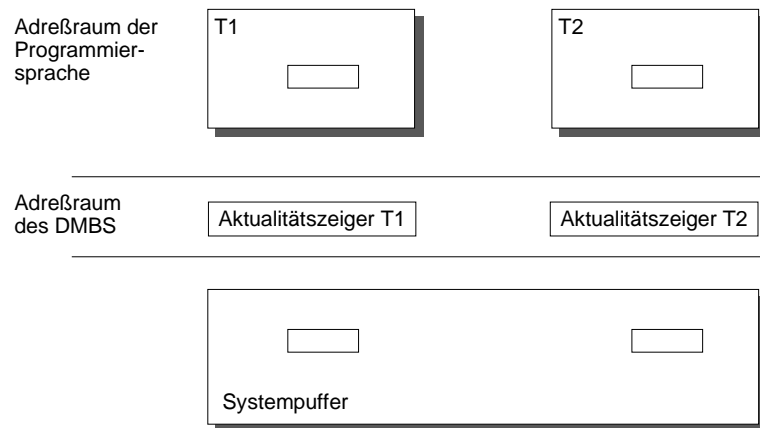
## S4: Die satzorientierte Schnittstelle (3)



Datenbanken und Informationssysteme

Die Fünf-Schichten-Architektur 9.2.36

## S4: Die satzorientierte Schnittstelle (4)



Datenbanken und Informationssysteme

Die Fünf-Schichten-Architektur 9.2.37

## S5: Die mengenorientierte Schnittstelle (1)

### Aufgabe und Objekte (nur im RDM und OODM vorhanden):

- Abbildung von *mengenorientierten Anfragen* in Folgen von Aufrufen der satzorientierten Schnittstelle
- Verwaltung von *mengenwertigen Anfrageergebnissen*

### Teilaufgaben:

- Evtl. Prüfung deklarativer benutzerdefinierter Integritätsbedingungen
- Evtl. iterative Berechnung rekursiver Anfragen
- Anfrageanalyse
  - Bindungen an Programmiersprachenvariablen
  - Auflösung und Bindung der externen an interne Objektnamen
  - Elimination von Sichten

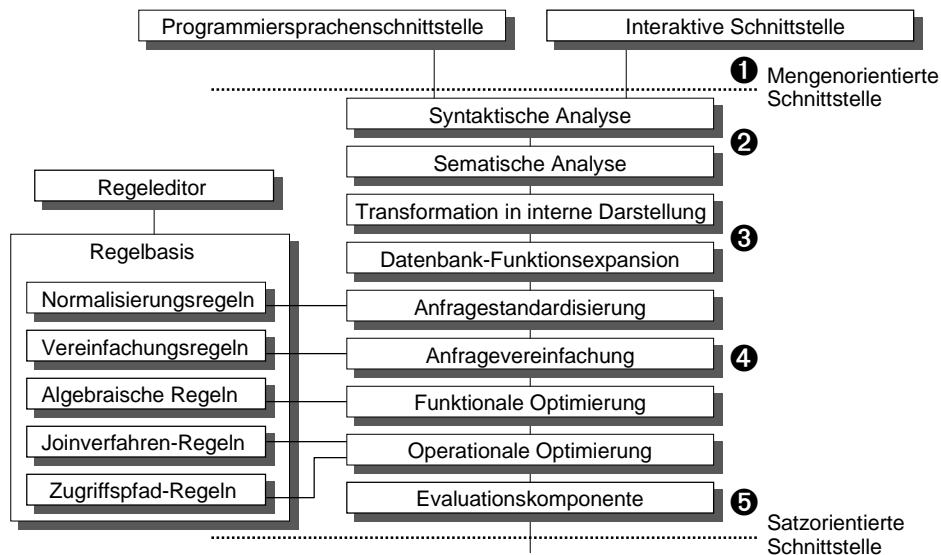
Datenbanken und Informationssysteme

Die Fünf-Schichten-Architektur 9.2.38

## S5: Die mengenorientierte Schnittstelle (2)

- ❑ Algebraische Optimierung
  - Heuristische Regeln
  - Anfragestrukturierung (Verfeinerung)
- ❑ Nicht-algebraische Optimierung
  - Realisierung der algebraischen Operatoren (Selektion, Projektion, 3 Verbundtypen)
  - Kostenmodelle
  - Planoptimierung
- ❑ Codegenerierung für mengenorientierte Anforderungen
- ❑ Ausführung von DB-Anforderungen
  - Ausführung von vorübersetzten Zugriffsroutinen
  - Behandlung von ad hoc-Anfragen

## S5: Überblick Anfrageoptimierung & -auswertung

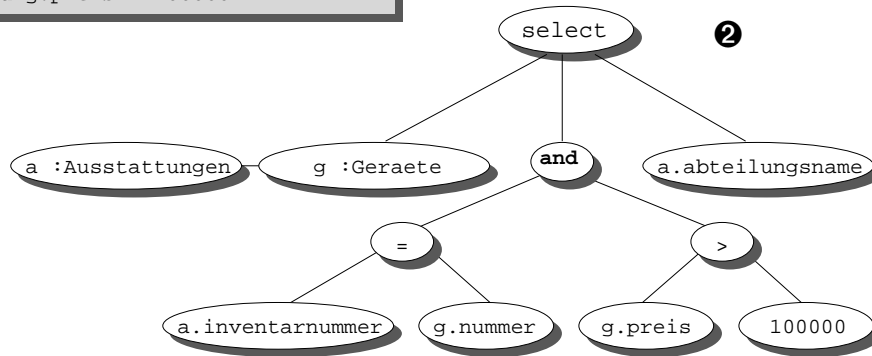


## S5: Syntaktische Analyse

```

select a.abteilungsname
from a in Ausstattungen,
      g in Geraete
where a.inventarnummer = g.nummer
      and g.preis > 100000;
    
```

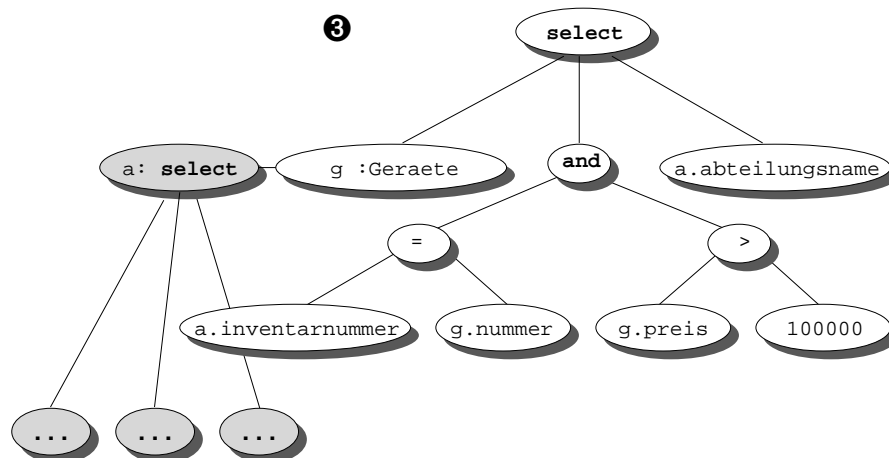
①



Datenbanken und Informationssysteme

Die Fünf-Schichten-Architektur 9.2.41

## S5: Datenbank-Funktionsexpansion



Datenbanken und Informationssysteme

Die Fünf-Schichten-Architektur 9.2.42

## S5: Algebraische Optimierung

Nutzung der folgenden Äquivalenzregeln mit dem Ziel der Anfragevereinfachung und Optimierung:

④

Q1:  $A \text{ AND SOME } r \text{ IN } R (B(r))$   
 $\Leftrightarrow$   
 $\text{SOME } r \text{ IN } R (A \text{ AND } B(r))$

Q3:  $A \text{ OR SOME } r \text{ IN } R (B(r))$   
 $\Leftrightarrow$   
 [a]  $\text{SOME } r \text{ IN } R (A \text{ OR } B(r)) \quad | R \neq \{\}$   
 [b]  $A \quad | R = \{\}$

Q2:  $A \text{ OR ALL } r \text{ IN } R (B(r))$   
 $\Leftrightarrow$   
 $\text{ALL } r \text{ IN } R (A \text{ OR } B(r))$

Q4:  $A \text{ AND ALL } r \text{ IN } R (B(r))$   
 $\Leftrightarrow$   
 [a]  $\text{ALL } r \text{ IN } R (A \text{ AND } B(r)) \quad | R \neq \{\}$   
 [b]  $A \quad | R = \{\}$

## S5: Evaluationskomponente

Ergebnisse der Optimierung sind Ausführungspläne, z.B.:

⑤

"Nested Loop" Evaluation:

```

resultat := {}
forEach a in Ausstattungen do
  forEach g in Geraete do
    if g.Preis > 10000 and
      a.inventarnr = g.nummer
    then
      resultat := resultat
        + {a.Abteilungen}
    end
  end
end
end;
```

Primärschlüsselindexzugriff:

```

resultat := {}
forEach g in Geraete do
  if g.Preis > 10000 then
    a = lookup(Ausstattungen,
      g.nummer)
    if found then
      resultat := resultat
        + {a.abteilungen}
    end
  end
end
end;
```