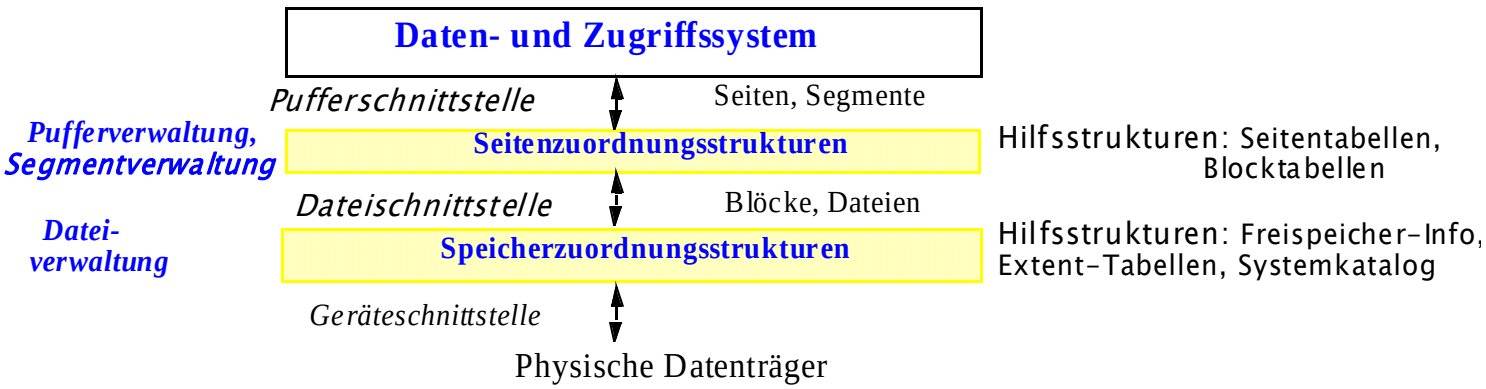
# 3. Speicher- und Seitenzuordnung



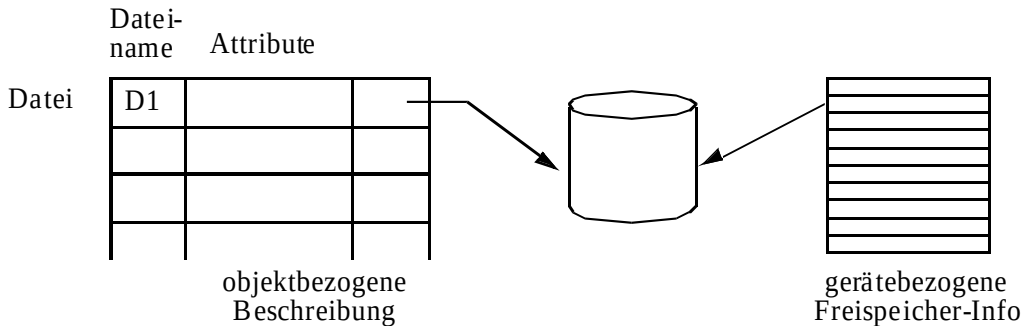

- Speicherzuordnungsstrukturen
  - Realisierung eines Dateikonzeptes
  - Verfahren der Blockzuordnung: statisch, dynamische Extent-Zuordnung, dynamische Block-Zuordnung
- Seitenzuordnungsstrukturen
  - Segmentkonzept
  - Indirekte Einbringstrategien
    - Bsp.: Schattenspeicherkonzept (shadow page mechanism), Zusatzdateikonzept
  - Pufferverwaltung -> Kap. 4



# Dateikonzept

- DB-Speicher = Menge von Dateien
- Dateien repräsentieren externe Speichermedien für Anwendungen in einer geräteunabhängigen Weise
- Dateisystem
  - realisiert Abbildung von physischen Blöcken und Dateien auf Externspeicher
  - verwaltet typischerweise einen hierarchischen Namensraum
- Operationen auf Dateien
  - Anlegen/Löschen (Create / Delete)
  - Verarbeiten (Open / Close)
  - Lesen/Schreiben (Read / Write)
- Dateiarten:
  - permanente vs. temporäre Dateien
  - unstrukturiert vs. strukturiert
  - sequentieller vs. direkter Zugriff
  - ggf. wertbasierter Zugriff

# Realisierung eines Dateikonzeptes

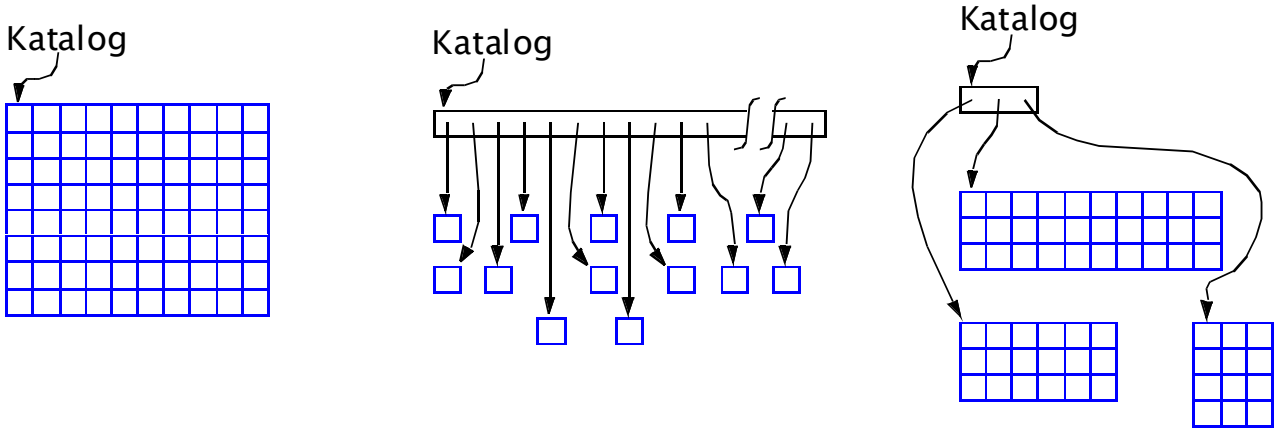



- Katalog für alle Dateien
  - feste Position
- objektbezogene Beschreibung durch Dateideskriptor
  - Dateiname, OwnerID, Zugriffskontrollliste, Zeitinformation über Erzeugung, letzter Zugriff, letzte Archivierung, Dateigröße, Externspeicherzuordnung, . . .
- Freispeicherverwaltung für Externspeicher (z.B. formatierte Bitlisten)
- Anlegen/Reservieren von Speicherbereichen (Erstzuweisung, Erweitern)
- Einheit des physischen Zugriffs: Block
  - feste Blocklänge pro Datei
  - Unterstützung für sequentielle E/A-Vorgänge




IDBS

# Blockordnungsverfahren

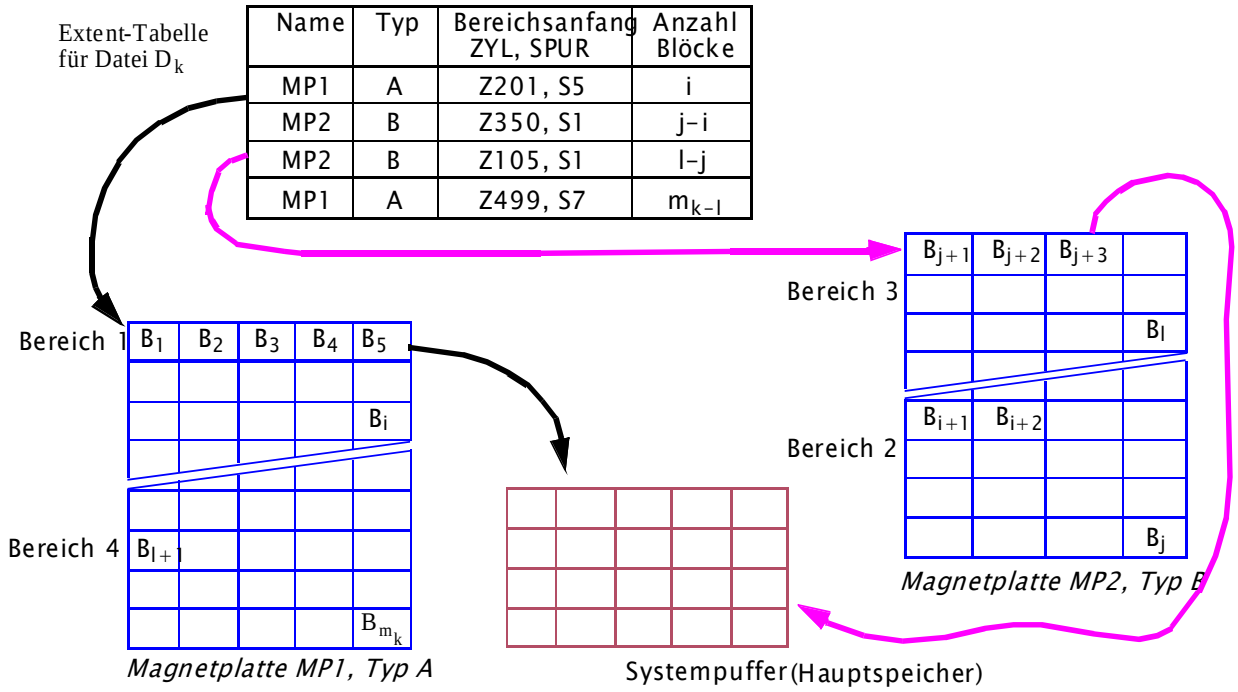



- Statische Datei-Zuordnung
  - direkte Adressierung
  - minimale Zugriffskosten
  - keine Flexibilität
- Dynamische Datei-Zuordnung
  - Adressierung über große Tabelle
  - hohe Zugriffskosten
  - maximale Flexibilität
- Dynamische Extent-Zuordnung
  - Adressierung über kleine Tabelle
  - geringe Zugriffskosten
  - mäßige Flexibilität




IDBS

# Blockzuordnung über Extent-Tabellen




Dargestellte Aktionen der Zugriffsprimitive: Hole Block $B_5$; Hole Block $B_{j+3}$



# Segmentkonzept

- Teil der Seitenzuordnungsstrukturen
- Aufteilung des logischen DB-Adressraumes in Segmente mit sichtbaren Seitengrenzen
- Realisierung eines Segmentkonzeptes:
  - Ermöglichung indirekter Einbringstrategien
  - selektive Einführung von zusätzlichen Attributen, z.B. zur Erhöhung der Fehlertoleranz
  - Segmente als Einheiten des Sperrens, der Recovery und der Zugriffskontrolle
  - Abbildung auf Dateien
- Segmentarten
  - öffentliche vs. private
  - permanente vs. temporäre

# Verwendung von Tablespaces

- Tablespace: Segmenttyp zur Speicherung von Tabellen (Relationen) sowie ggf. Indexstrukturen
  - in manchen DBS werden Indexstrukturen in eigenen Segmentarten verwaltet (Indexspaces)

- Tablespace kann auf mehrere Dateien abgebildet werden
  - Allokation von Tablespaces:

```
CREATE TABLESPACE tablespacename
DATAFILE filename SIZE size { "," filename SIZE size }
```

  - Erweiterung von Tablespaces:

```
ALTER TABLESPACE tablespacename
ADD DATAFILE filename SIZE size { "," filename SIZE size }
```

- Zuordnung von Relationen zu Tablespaces

```
CREATE TABLE tablename ( ...
[ TABLESPACE tablespacename ]
[ STORAGE INITIAL size NEXT size ] [PCTFREE percent ] )
```

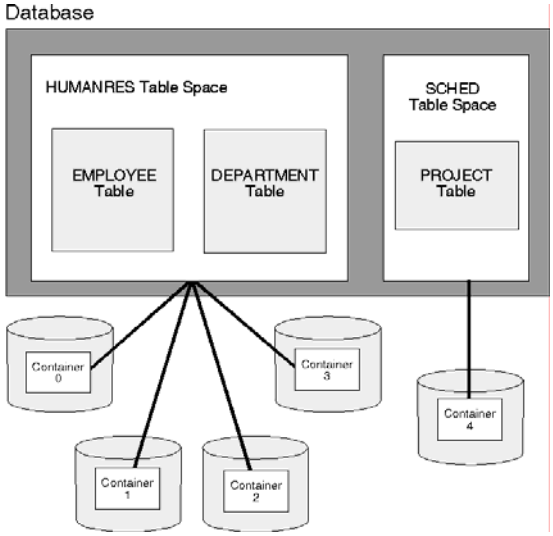


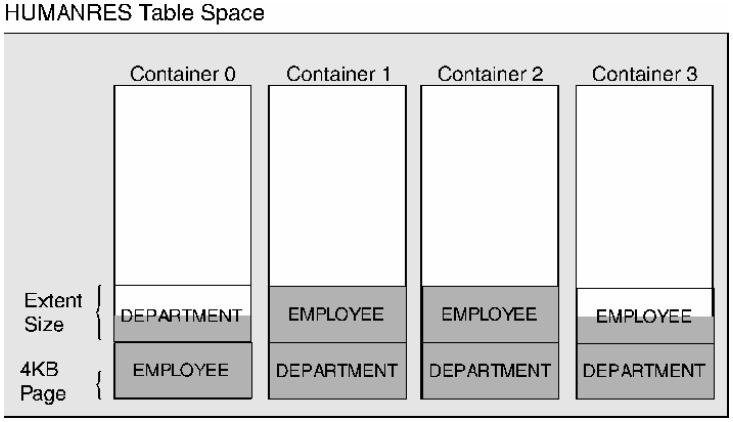
# Tablespaces in DB2

- 3 Segmentarten: Catalog, User, Temporary Tablespace

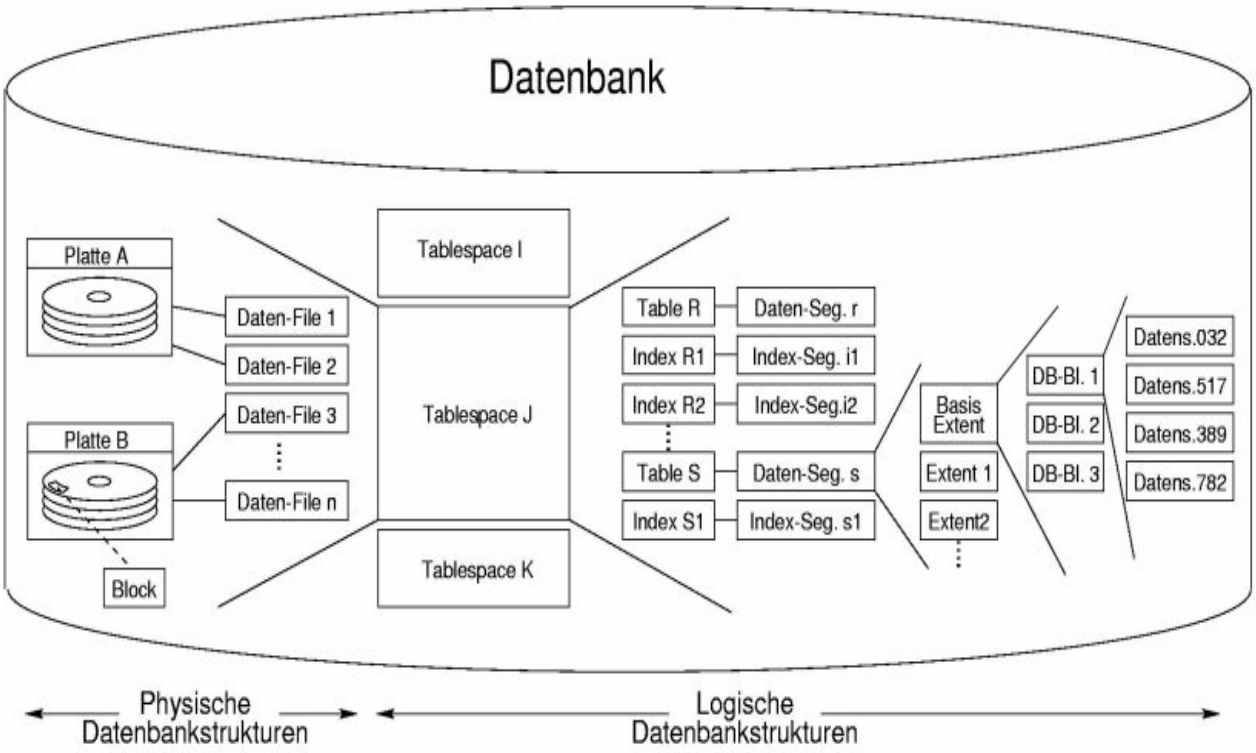
```
CREATE DATABASE PERSONL
    CATALOG TABLESPACE MANAGED BY SYSTEM USING ('d:\pcatalog','e:\pcatalog')
                EXTENTSIZE 32 PREFETCHSIZE 16
    USER TABLESPACE MANAGED BY DATABASE USING (FILE'd:\db2data\personl' 5000,
                FILE'd:\db2data\personl' 5000) EXTENTSIZE 32 PREFETCHSIZE 64
    TEMPORARY TABLESPACE MANAGED BY SYSTEM USING ('f:\db2temp\personl')
                WITH "Personal DB for BSchiefer"
```

- Tablespace kann auf mehrere Container/Platten aufgeteilt werden



# Speicherorganisation in Oracle

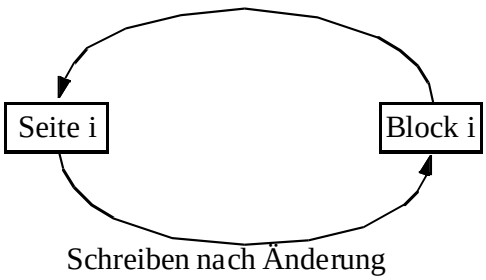




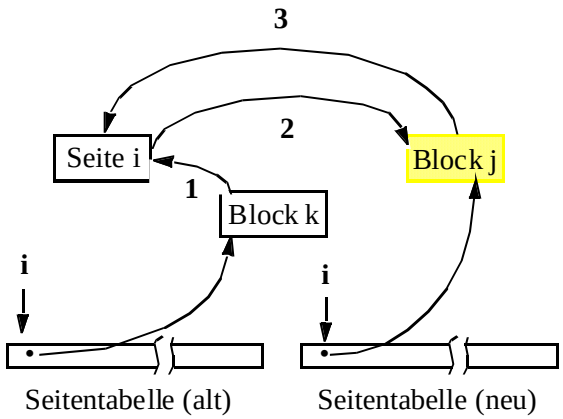

# Seitenzuordnungsverfahren

## Direkte Seitenzuordnung

(Update in Place)




## Indirekte Seitenzuordnung

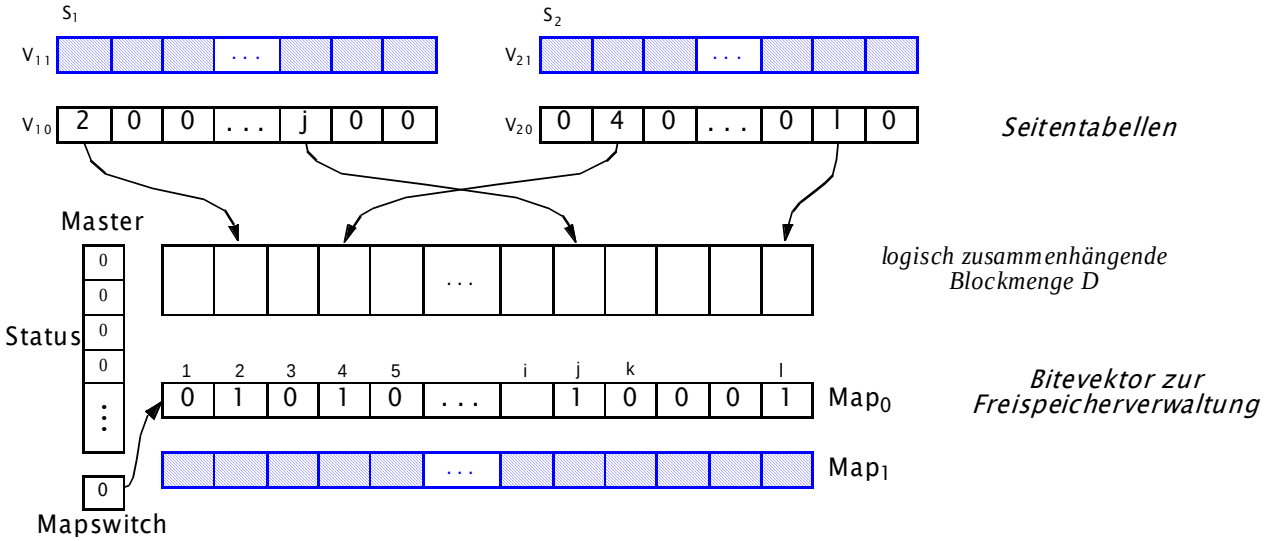



1) Lesen vor Änderung
2) Schreiben nach Änderung
3) Lesen nach Änderung

- Indirekte Zuordnung
  - Seitentabelle: Abbildung von Seitennr. → Blocknr.
  - Einbringen (mengenorientiert): Umschalten der Seitentabellen

# Einsatz des Schattenspeicher-Verfahrens

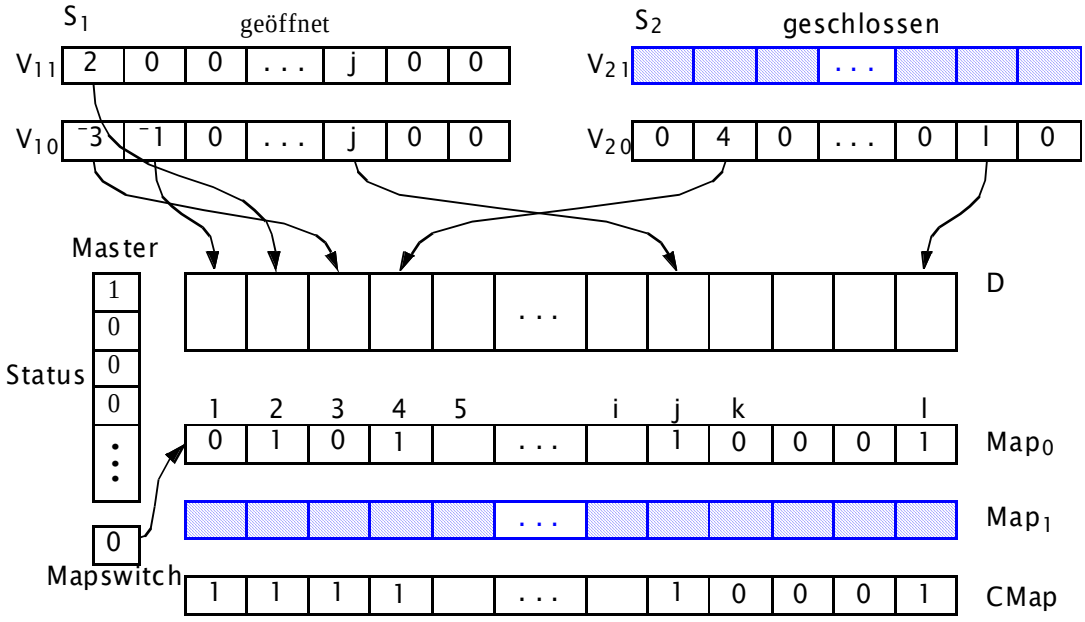
– Die schraffierten Strukturen sind in der Ausgangssituation nicht benutzte Speicherbereiche, die erst nach der Eröffnung für Änderungsbetrieb erforderlich werden.




– Status (i) enthält den Eröffnungszustand für Segment i (hier: **alle Segmente geschlossen**).
– MAPSWITCH zeigt an, welche der beiden (gleichberechtigten) Tabellen Map0 und Map1 das aktuelle Verzeichnis belegter Blöcke enthält.
– Wenn ein Segment geschlossen ist, erfüllt sein Inhalt bestimmte, von höheren Schichten kontrollierte Konsistenzbedingungen.



# Änderungsbetrieb beim Schattenspeicher-Verfahren




Segment $S_1$ geändert

– Auf einer Blockmenge D können gleichzeitig mehrere Segmente für Änderungen geöffnet sein
– CMAP enthält für alle Segmente die mit Schatten- bzw. aktuellen Seiten belegten Blöcke
– Einer Seite wird nur bei der erstmaligen Änderung nach Eröffnen des Segmentes ein neuer Block zugewiesen

# Funktionsprinzip des Schattenspeicher-Verfahrens

- Öffnen von **Segment k für Änderungen**:
  - kopiere $V_{k0}$ nach $V_{k1}$
  - STATUS(k) := 1
  - Schreibe MASTER in einer ununterbrechbaren Operation aus
  - Lege im Hauptspeicher eine Arbeitskopie CMAP von $MAP_0$ an.
- Erstmalige Änderung (seit Segmentöffnung) von **Seite $P_i$**:
  - Lies Seite Pi aus Block j = $V_{k0}(i)$
  - Finde einen freien Block j' in CMAP
  - $V_{k0}(i)$ = j'
  - Markiere Seite $P_i$ in $V_{k0}(i)$ als geändert (setze Schattenbit)
  - Bei weiteren Änderungen von $P_i$ wird Block j' verwendet.
- Beenden eines Änderungsintervalls:
  - Erzeuge Bitliste mit aktueller Speicherbelegung in MAP1 (neue Blöcke belegt, alte freigegeben)
  - Schreibe $MAP_1$ (kein Überschreiben von $MAP_0$)
  - Schreibe $V_{k0}$
  - Schreibe alle geänderten Blöcke
  - STATUS(k) := 0, MAPSWITCH = 1 ($MAP_1$ ist aktuell)
  - Schreibe MASTER in einer ununterbrechbaren Operation aus.
- **Zurücksetzen geöffneter Segmente**:
  lediglich $V_{k1}$ in $V_{k0}$ zu kopieren und STATUS(k) := 0




IDBS

# Bewertung des Schattenspeicher-Konzeptes

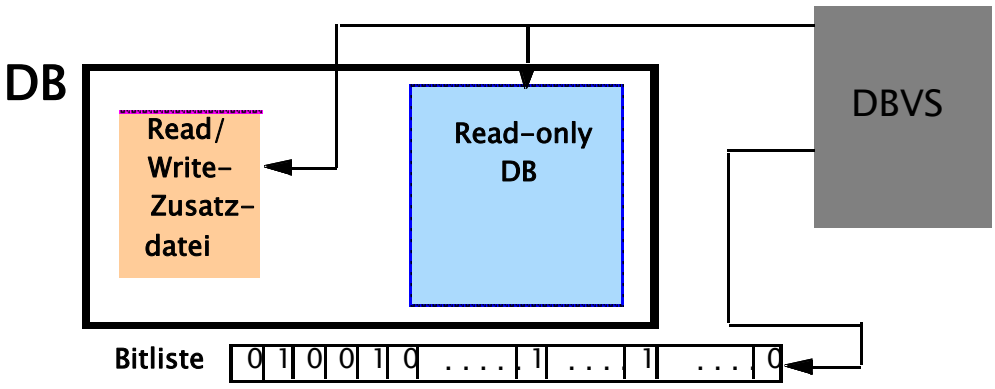
- Vorteile
  - Rücksetzen auf letzten konsistenten Zustand sehr einfach
  - flexiblere Schreibprotokolle für Log-Daten: Pufferung bis zum Umschalten auf einen neuen Zustand möglich
  - physische DB kann operationskonsistent gehalten werden => Operationen-Logging möglich
  - bei katastrophalem Fehler kann eher noch einen brauchbarer DB-Zustand rekonstruiert werden
- Nachteile
  - Hilfsstrukturen (MAP und Seitentabellen $V_i$) werden so groß, dass Blockzerlegung notwendig wird
  - Die Seitentabellen $V_i$ belegen etwa 0.1-0.2% der DB-Größe, was bei größeren Datenbanken viele zusätzliche E/A-Operationen verursachen kann
  - periodische Sicherungspunkte erzwingen Ausschreiben des gesamten DB-Puffers
  - Doppelspeicherung ungünstig für lange Batch- (Änderungs-)Programme
  - physische Clusterbildung logisch zusammengehöriger Seiten wird beeinträchtigt bzw. zerstört




IDBS

# Zusatzdatei-Verfahren

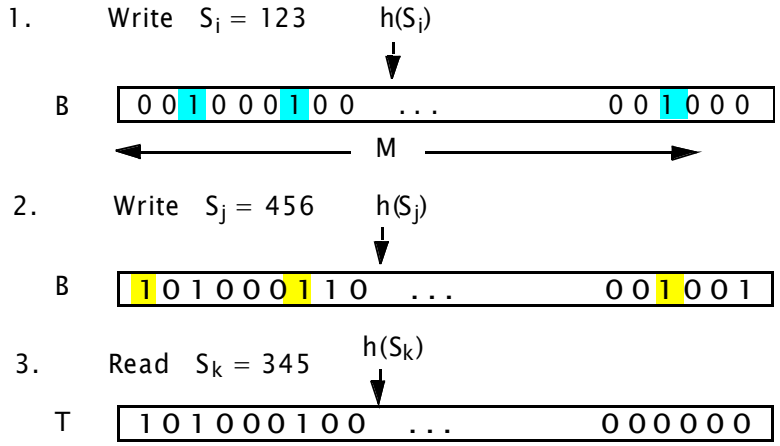
- Idee:
  - wenn Datei für Änderungsbetrieb geöffnet wird, wird eine zusätzliche, temporäre Datei angelegt (Zusatzdatei, differential file) für alle modifizierten Blöcke pro Änderungsintervall
  - eigentliche Datei wird nicht verändert
  - Ende des Änderungsintervalles: Kopieren aller veränderten Blöcke aus der temporären in permanente Datei
- wesentliches Problem: Entscheidung für eine gegebene Seiten-Nr., ob die aktuelle Version in der temporären oder permanenten Datei steht
  - Nutzung einer Bitliste, die anzeigt, ob eine Seite möglicherweise geändert wurde
  - Hash-Abbildung erlaubt Begrenzung des Speicherplatzbedarfs






# Bloom-Filter

- Wirkungsweise
  - Bitliste B der Länge M im Hauptspeicher
  - Abbildung des Schlüssels (Seiten-Nr) bei Änderung über Hash-Funktionen
  - Hash-Funktionen adressieren x Bits, die in B auf 1 gesetzt werden
- Aufsuchen von Satz $S_i$
  - Erzeugen der charakteristischen x Bits in temporärer Bitliste T
  - AND-Operation von T und B in Erg
  - wenn alle x Bits in Erg gesetzt: VIELLEICHT; sonst: NEIN
- Beispiel:




Wenn (B AND T) = T,
dann Antwort VIELLEICHT !

# Zusammenfassung

- Speicherzuordnungsstrukturen erfordern effizientes Dateikonzept
  - viele Dateien variierender, nicht statisch festgelegter Größe
  - Wachstum und Schrumpfung erforderlich
  - permanente und temporäre Dateien
- empfohlene Datei-Eigenschaften:
  - direkter und sequentieller Blockzugriff
  - Blockgröße pro Datei definierbar
  - Blockzuordnung über dynamische Extents
- Segmentkonzept
  - erlaubt die Realisierung zusätzlicher Attribute für die DB-Verarbeitung (Recovery, Clusterbildung für Relationen usw.)
  - zweistufige Abbildung von Segment/Seite auf Datei/Block und diese auf Slots der Magnetplatten erlaubt indirekte Einbringstrategien
- Indirekte Einbringstrategien (Bsp.: Schattenspeicherkonzept)
  - sind teurer als direkte Einbringstrategien (Update in Place)
  - sie belasten den Normalbetrieb zu Gunsten der Recovery