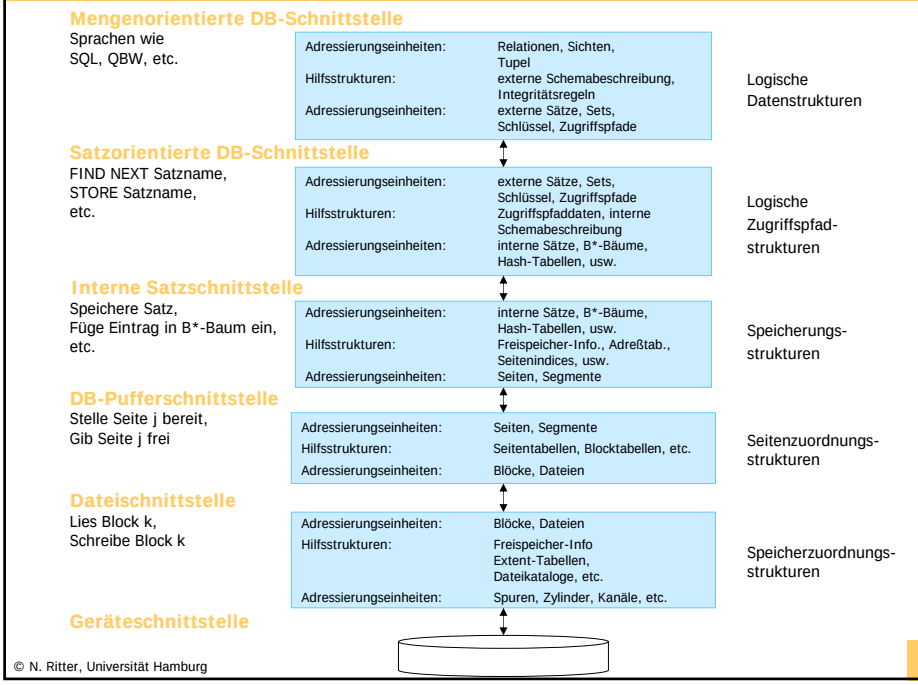
Implementierung von Datenbanksystemen

Kapitel 7: Satzorientierte Schnittstelle

Kapitel 7: Satzorientierte Schnittstelle

- Ziele
  - Entwurfsprinzipien für die satzorientierte Verarbeitung und die Navigation auf logischen Zugriffspfaden
  - Entwicklung einer Scan-Technik und eines Sortieroperators
- Logische Zugriffspfadstrukturen
  - Objekte und Operatoren
  - Abbildung von externen Sätzen
- Satzorientierte Verarbeitung
  - Verarbeitungsprimitive
  - Navigationskonzepte
- Scan-Technik
  - Implementierung durch Scan-Operatoren
  - Scan-Typen
- Sortierkomponente zur Unterstützung relationaler Operationen
  - Einsatz eines Sortieroperators
  - Externes Sortieren



Logische Zugriffspfadstrukturen

- Operationen an der oberen Schnittstelle:

```
STORE <record>
FETCH <record> USING <attr1> = 400 AND <attr2> >=7
CONNECT <record> TO <set>
```

- Abbildungsfunktionen:

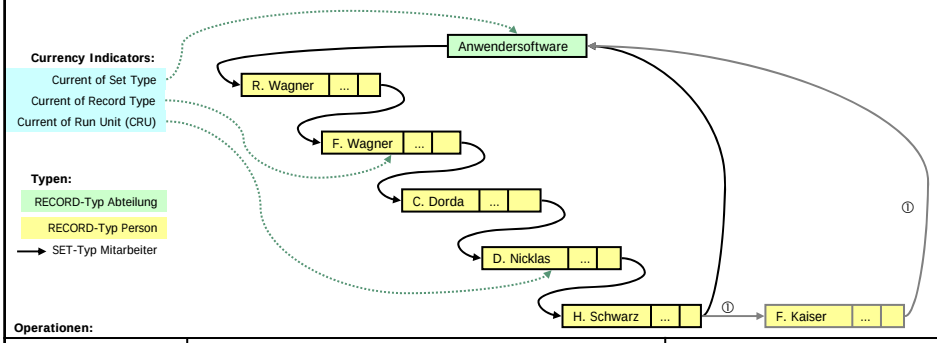
```
Physischer Satz     <-> Externer Satz
Externer Satz       <-> zugehörige Zugriffspfade
Suchausdruck         -> unterstützende Zugriffspfade
```

- Operationen an der unteren Schnittstelle:

```
insert <record> at ...
add <entry> to ...
retrieve <address-list> from...
retrieve <record> with
```

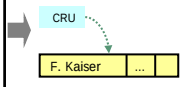
- Eigenschaften der oberen Schnittstelle
  - Zugriff auf logische Sätze in Einheiten von einem Satz pro Aufruf
  - Logische Zugriffspfade (inhaltsadressierbarer Speicher, hierarchische Beziehungen zwischen Satztypen)
  - interne Schnittstelle bei mengenorientierten DBS
  - externe Schnittstelle (Anwendungsprogrammierschnittstelle/API) bei satzorientierten DBS (z.B. Netzwerk-Datenmodell oder objektorientiertes Datenmodell)

Exkurs: Netzwerk-Datenmodell



Operationen:

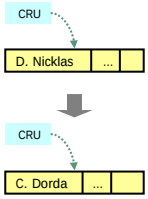
STORE <F. Kaiser, ...>



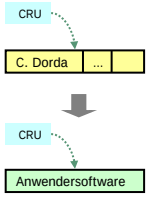
ERASE Person

MODIFY ...

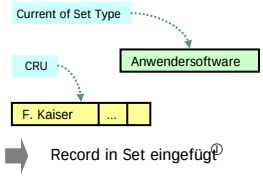
FIND Person USING <C. Dorda>



FIND OWNER WITHIN Mitarbeiter



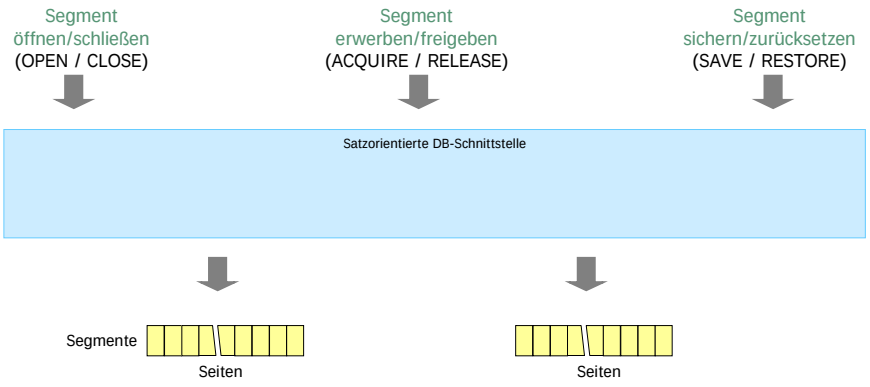
CONNECT Person TO Mitarbeiter



DISCONNECT Person FROM Mitarbeiter

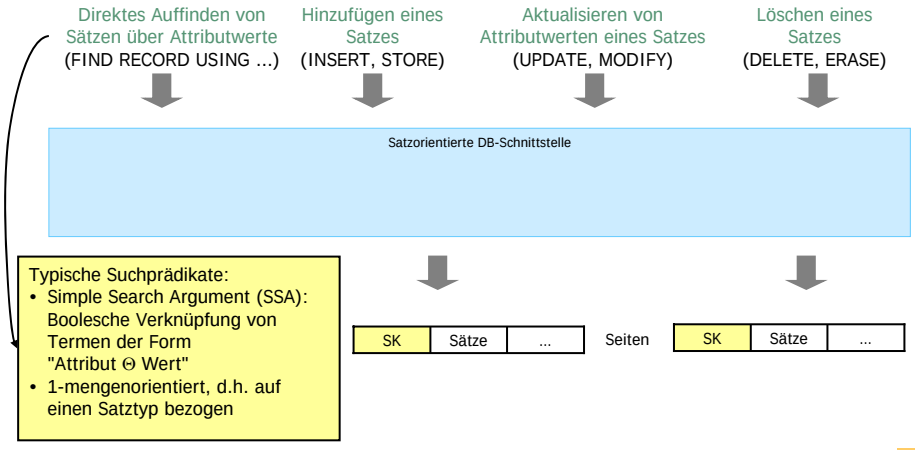
Satzorientierte DB-Schnittstelle

- Speicherverwaltung



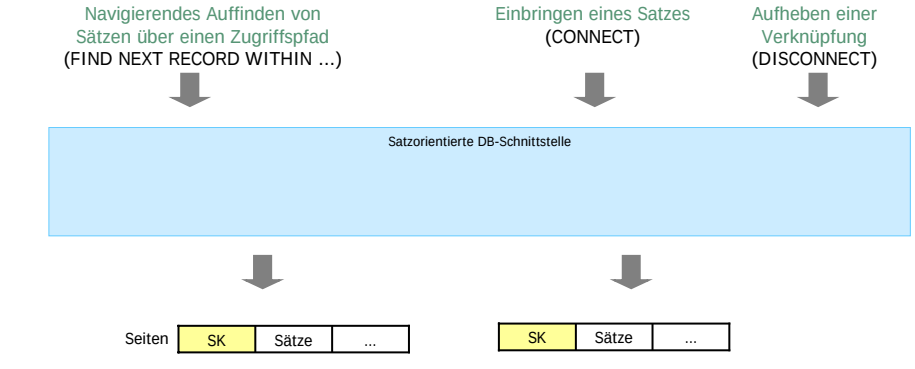
Satzorientierte DB-Schnittstelle

- Operatoren auf Sätzen (Tupel) von Satztypen (Tabellen)



Satzorientierte DB-Schnittstelle

- Operatoren auf Zugriffspfaden
  - Index (Search Key)
  - Set (Link)



Satzorientierte DB-Schnittstelle

- Weitere Operatoren

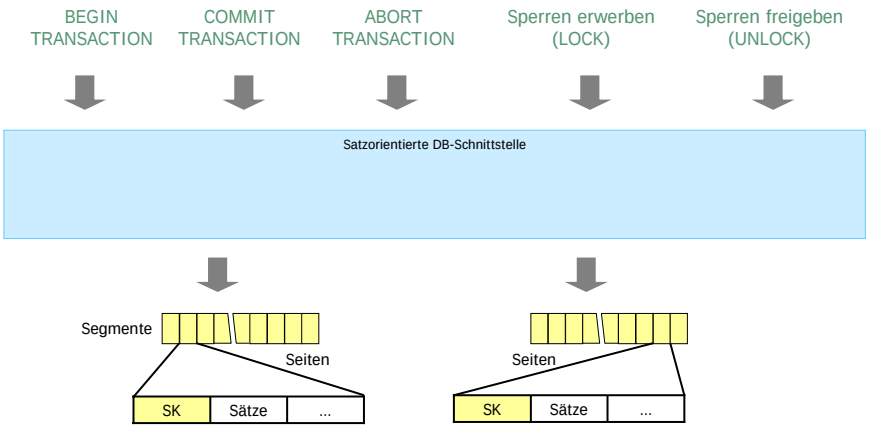


Abbildung von externen Sätzen

- Vorteile der Trennung externer und interner Sätze
  - Realisierung des Sichtenkonzepts (externes Schema, Subschema)
  - Partitionierte Speicherung großer Satzmengen
    - Zuordnung disjunkter Satzmengen zu separaten Speicherungseinheiten
    - Spezifikation der Partitionierung über Werte (Schlüsselbereiche, Hashing) oder über Prozeduren (user exit)
    - Leistungsgründe: E/A-Parallelität
    - Verfügbarkeit: Erstellung von Kopien, Migration, . . .
    - Partition ist Einheit der Reorganisation, des Backup, der Archivierung, des Ladens, von Zugriffsverfahren etc.
  - Möglichkeiten der Abstimmung/Verbesserung des Leistungsverhaltens (Tuning)
    - Aufteilung und Zuordnung der Felder nach Zugriffshäufigkeiten
    - redundante Speicherung
    - Verdichtung von Feldern und Sätzen

Kapitel 7: Satzorientierte Schnittstelle

- Logische Zugriffspfadstrukturen
  - Objekte und Operatoren
  - Abbildung von externen Sätzen
- Satzorientierte Verarbeitung
  - Verarbeitungsprimitive
  - Navigationskonzepte
- Scan-Technik
  - Implementierung durch Scan-Operatoren
  - Scan-Typen
- Sortierkomponente zur Unterstützung relationaler Operationen
  - Einsatz eines Sortieroperators
  - Externes Sortieren

Verarbeitungsprimitive

- Welche Verarbeitungsprimitive werden durch die Schicht der Speicherungsstrukturen direkt unterstützt?
- Welcher Zusatzaufwand entsteht in der Schicht logischer Zugriffspfadstrukturen?

Verarbeitungsprimitiv	Konzept	Logische Zugriffspfadstrukturen	Speicherungs-strukturen
Direkter Zugriff über Primärschlüssel	Kontextfeie Operationen	Passenden Zugriffspfad identifizieren (DB-Katalog)	Zugriffspfade bereitstellen
Satzweise Aktualisierung (INSERT, UPDATE, DELETE)		Wartung betroffener Zugriffspfade	interne Satzschnittstelle
Satzweise Navigation	Navigation	Navigationskonzept, Scan	Zugriffspfade und interne Satzschnittstelle bereitstellen
Sortierung einer Satzmenge	Sortierung	Sortieroperator	

Navigationskonzepte

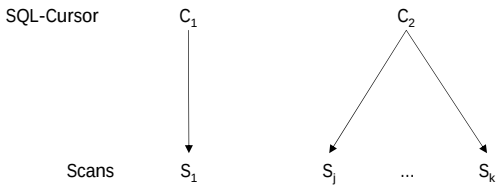
- Bereitstellen und Warten von transaktionsbezogenen Verarbeitungspositionen zur Navigation
- Verschiedene Cursor-Konzepte:
- Implizite Cursor
  - Bsp.: Currency-Konzept bei CODASYL
  - Änderung mehrerer Cursor durch Ausführung einer DB-Anweisung
  - Benutzerkontrolle durch RETAINING-Klausel
    - Hochgradige Komplexität
    - Fehleranfälligkeit
- Explizite Cursor
  - Bsp.: SQL-Cursor
  - Cursor werden durch AP definiert, verhalten sich wie normale Programmvariablen und werden unter expliziter Kontrolle des AP verändert
  - Definition von n Cursor auf einer Tabelle
  - Eindeutige Änderungssemantik

Kapitel 7: Satzorientierte Schnittstelle

- Logische Zugriffspfadstrukturen
  - Objekte und Operatoren
  - Abbildung von externen Sätzen
- Satzorientierte Verarbeitung
  - Verarbeitungsprimitive
  - Navigationskonzepte
- Scan-Technik
  - Implementierung durch Scan-Operatoren
  - Scan-Typen
- Sortierkomponente zur Unterstützung relationaler Operationen
  - Einsatz eines Sortieroperators
  - Externes Sortieren

Scan-Technik

- Was ist ein Scan?
  - Ein Scan erlaubt das satzweise Durchmustern und Verarbeiten einer physischen Satzmenge.
  - Ein Scan kann an einen Zugriffspfad (Index, Link, ...) gebunden werden.
  - Ein Scan definiert eine Verarbeitungsposition, die als Kontextinformation für die Navigation dient.
- Wie unterstützt das Scan-Konzept die mengenorientierte Verarbeitung?



Scan-Typen

- Table-Scan (Satztyp-Scan, Tabellen-Scan)
  - zum Aufsuchen aller Sätze eines Satztyps (einer Tabelle)
- Index-Scan
  - zum Aufsuchen von Sätzen in einer wertabhängigen Reihenfolge
- Link-Scan
  - zum Aufsuchen von Sätzen in benutzerkontrollierter Einfügereihenfolge
- k-d-Scan
  - zum Aufsuchen von Sätzen über einen k-dimensionalen Index
  - Anmerkung:
    Es ist wünschenswert, für alle mehrdimensionalen Zugriffspfade ein einheitliches Auswertungsmodell anbieten zu können. Dadurch würde sich eine DBS-Erweiterung um einen neuen mehrdimensionalen Zugriffspfadtyp lokal begrenzen lassen. Jedoch dürfte die direkte Abbildung des Auswertungsmodells bei manchen Strukturen sehr komplex und gar unmöglich sein. Als Ausweg bietet sich hier an, das Anfrageergebnis mit Hilfe der verfügbaren Operationen des physischen Zugriffspfads abzuleiten, ggf. zu sortieren und in einer temporären Speicherungsstruktur zu materialisieren. Auf dieser Speicherungsstruktur könnte dann der k-d-Scan nachgebildet werden, um das abstrakte Auswertungsmodell für die satzweise Verarbeitung zu realisieren. Falls ein k-d-Scan nur ungeordnete Treffermengen abzuliefern braucht, sind sicherlich einfachere Auswertungsmodelle, die sich stärker an den Eigenschaften der darunter liegenden physischen Strukturen orientieren, denkbar.

Implementierung von Scans

- explizite Definition/Freigabe: OPEN/CLOSE SCAN
- Navigation: NEXT TUPLE
- Scans werden auf Zugriffspfaden definiert
- Optionen:
  - Start-, Stopp- und Suchbedingung (Simple Search Argument)
  - Suchrichtung: NEXT/PRIOR, FIRST/LAST, n-th
- Scan-Kontrollblock (SKB bzw. SCB):
  Angaben über Typ, Status, Position etc.

Tabellen-Scan

- Anfragebeispiel:

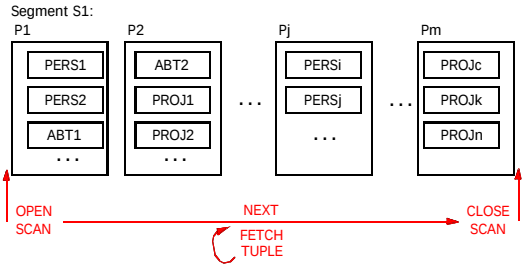
```
SELECT   *
FROM     PERS
WHERE    ANR BETWEEN K28 AND K67
AND      BERUF = 'Programmierer'
```

- Scan-Optionen
  - Startbedingung (SB): BOS (Beginn von Segment S1)
  - Stoppbedingung (STB): EOS (Ende von Segment S1)
  - Suchrichtung: NEXT
  - Suchbedingung (SSA): ANR ≥ 'K28' AND ANR ≤ 'K67' AND BERUF = 'Programmierer'

Ablauf eines Tabellen-Scans

- Tabellen-Scan:

```
OPEN SCAN (PERS, BOS, EOS)                                   /* SCB1 */
WHILE (NOT FINISHED)
DO
        FETCH TUPLE (SCB1, NEXT,
        ANR ≥ 'K28' AND ANR ≤ 'K67' AND BERUF = 'Programmierer')
        . . .
END
CLOSE SCAN (SCB1)
```



Index-Scan

- Anfragebeispiel:

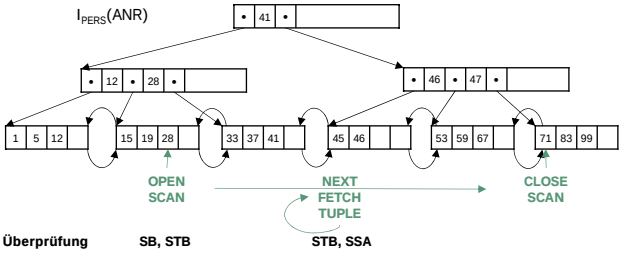
```
SELECT    *
FROM      PERS
WHERE     ANR BETWEEN K28 AND K67
AND       BERUF = 'Programmierer'
```

- Scan-Optionen
  - Startbedingung (SB): ANR ≥ 'K28'
  - Stoppbedingung (STB): ANR > 'K67'
  - Suchrichtung: NEXT
  - Suchbedingung (SSA): BERUF = 'Programmierer'

Ablauf eines Index-Scans

- Index-Scan:

```
OPEN SCAN (I_PERS(ANR), ANR ≥ 'K28', ANR > 'K67')          /* SCB1 */
WHILE (NOT FINISHED)
DO
        FETCH TUPLE (SCB1, NEXT, BERUF = 'Programmierer')
        . . .
END
CLOSE SCAN (SCB1)
```



Die Verweise (TIDs) auf die PERS-Tupel sind in I_{PERS}(ANR) weggelassen. Die Suchbedingung (SSA = simple search argument) darf nur Wertvergleiche „Attribut Θ Wert“ (mit $\Theta \in \{<, =, >, \leq, \neq, \geq\}$) enthalten und wird auf jedem Tupel überprüft, das über den Index-Scan erreicht wird. Der Operator FETCH TUPLE liefert also nur Tupel zurück, die die WHERE-Bedingung erfüllen.

Link-Scan

- Anfragebeispiel:

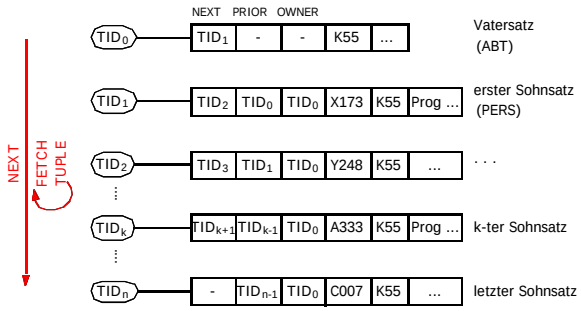
```
SELECT   *
FROM     PERS
WHERE    ANR BETWEEN K28 AND K67
AND      BERUF = 'Programmierer'
```

- Hierarchischer Zugriffspfad $L_{ABT\text{-}PERS}$
- Auffinden des Vaters
  - Startposition bereits gefunden (z.B. für ANR=K55 gegeben)
  - Fortschalten in ABT erforderlich (ANR BETWEEN K28 AND K67)

Ablauf eines Link-Scans

- Einzelner Link-Scan:

```
OPEN SCAN (L_ABT-PERS(ANR), NONE, EOL)                    /* SCB1 */
WHILE (NOT FINISHED)
        DO
        FETCH TUPLE (SCB1, NEXT, BERUF = 'Programmierer')
        . . .
END
CLOSE SCAN (SCB1)
```



Schachtelung von Index- und Link-Scan

- Index I_{ABT}(ANR) und hierarchischer Zugriffspfad $L_{ABT-PERS}$
- Auffinden des Vaters
  - Nutzung der Indexstruktur I_{ABT}(ANR)
  - Schachtelung von Index-Scan (ANR BETWEEN K28 AND K67) und Link-Scan (BERUF = 'Programmierer')
- Scan-Optionen

	Index-Scan	Link-Scan
Startbedingung:	ANR ≥ 'K28'	-
Stoppbedingung:	ANR > 'K67'	EOL
Suchrichtung:	NEXT	NEXT
Suchbedingung:	-	BERUF = 'Programmierer'

Schachtelung von Index- und Link-Scan

- Index- und Link-Scan:

```
OPEN SCAN (I_ABT(ANR), ANR ≥ 'K28', ANR > 'K67')                    /* SCB1 */
. . .
        WHILE (NOT FINISHED)
        DO
        FETCH TUPLE (SCB1, NEXT, NONE)
        . . .
        OPEN SCAN (L_ABT-PERS(ANR), NONE, EOL)                      /* SCB2 */
        . . .
        WHILE (NOT FINISHED)
        DO
                FETCH TUPLE (SCB2, NEXT, BERUF = 'Programmierer')
                . . .
        END
        CLOSE SCAN (SCB2)
END
CLOSE SCAN (SCB1)
```


k-d-Scan

- Anfragebeispiel:

```
SELECT    *
FROM      PERS
WHERE     ANR BETWEEN 'K28' AND 'K67'
          AND ALTER BETWEEN 20 AND 30
          AND BERUF = 'Programmierer'
```

- 2-dimensionaler Zugriffspfad I_{PERS}(ANR,ALTER)
- Scan-Optionen

	Dimension 1	Dimension 2
▪ Startbedingung:	ANR ≥ 'K28'	ALTER ≥ 20
▪ Stoppbedingung:	ANR > 'K67'	ALTER > 30
▪ Suchrichtung:	NEXT	NEXT
▪ Suchbedingung:	BERUF = 'Programmierer' (wird auf den PERS-Sätzen ausgewertet)	

Ablauf eines k-d-Scans

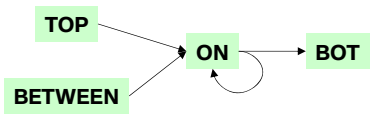
- 2-d-Scan:

```
OPEN SCAN (I_PERS(ANR, ALTER), ANR ≥ 'K28' AND ALTER ≥ 20,
           ANR > 'K67' AND ALTER > 30)                          /* SCB1 */
           . . .
WHILE (NOT (ALTER > 30))
DO
           /* Zwischenspeichern der SCB1-Position in SCANPOS      */
           WHILE (NOT (ANR > 'K67'))
           DO
           FETCH TUPLE (SCB1, NEXT(ANR), BERUF = 'Programmierer')
           . . .
           END
           /* Zurücksetzen der SCB1-Position auf SCANPOS          */
           . . .
           MOVE SCB1 TO NEXT(ALTER)
END
CLOSE SCAN (SCB1)
```

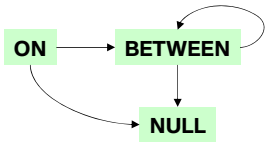

Scan-Semantik

- Scan-Zustände:
  - in Abhängigkeit von der Position in der Satzmenge
  - vor dem ersten Satz (TOP)
  - auf einem Satz (ON)
  - in einer Lücke zwischen zwei Sätzen (BETWEEN)
  - hinter dem letzten Satz (BOT)
  - in einer leeren Menge (NULL)
- Scan-Semantik durch geeignete Übergangsregeln festlegen!

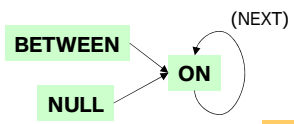
- Zustandsübergänge beim Scan
  - Suche



  - Delete



  - Insert

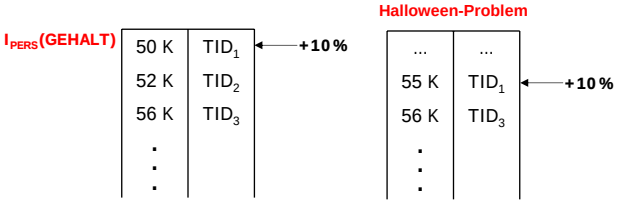


Anwendung von Scans

- Problem:
  Mengenorientierte Spezifikation <---> satzorientierte Auswertung
- Die navigierende Auswertung einer deklarativen SQL-Anweisung über einen Scan kann Verarbeitungsprobleme verursachen, insbesondere wenn das zu aktualisierende Objekt (Tabelle, Zugriffspfad) vom Scan benutzt wird (Verbot?)
- Beispiel:

```
UPDATE PERS
SET GEHALT = 1.1 * GEHALT
```

- Ausführung:
  Gehaltsverbesserung um 10% werde mit Hilfe eines Scans über Index GEHALT durchgeführt



Quelle: [Gra93]

Kapitel 7: Satzorientierte Schnittstelle

- Logische Zugriffspfadstrukturen
  - Objekte und Operatoren
  - Abbildung von externen Sätzen
- Satzorientierte Verarbeitung
  - Verarbeitungsprimitive
  - Navigationskonzepte
- Scan-Technik
  - Implementierung durch Scan-Operatoren
  - Scan-Typen
- Sortierkomponente zur Unterstützung relationaler Operationen
  - Einsatz eines Sortieroperators
  - Externes Sortieren

Einsatz eines Sortieroperators

- Explizite Umordnung der Sätze gemäß vorgegebenem Suchschlüssel (ORDER BY-Klausel)
- Umordnung mit Durchführung einer Restriktion

```
SELECT * FROM  PERS
WHERE ANR > 'K50'
ORDER BY GEHALT DESC
```

Partitionierung von Satzmengen

```
SELECT ANR, AVG (GEHALT)
FROM PERS
GROUP BY ANR
```

Duplikateliminierung in einer Satzmenge

```
SELECT DISTINCT BERUF
FROM PERS
WHERE ANR > 'K50' AND ANR < 'K56'
```

- Unterstützung von Mengen- und Verbundoperationen
- Reduktion der Komplexität von $O(N^2)$ nach $O(N \log N)$ bei Mengen- und Verbundoperationen
- Umordnen von Zeigern zur Optimierung der Auswertung oder Zugriffsreihenfolge
- Dynamische Erzeugung von Indexstrukturen ("bottom-up"-Aufbau von B*-Bäumen)
- Erzeugen einer Clusterbildung beim Laden und während der Reorganisation

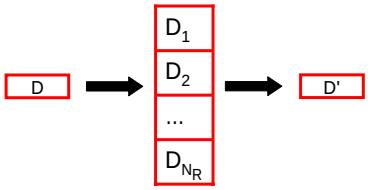
SORT-Operator - Optionen und Anwendung

- Scans können mit Suchbedingungen eingeschränkt sein
  - (SSA = Simple Search Arguments)
- SORT-Optionen zur Duplikateliminierung:
  - N = keine Eliminierung
  - K = Duplikateliminierung bezüglich Sortierkriterium
  - S = STOPP, sobald Duplikat entdeckt wird
- SORT dient als Basisoperator für Operationen auf höherer Ebene
- Beispiel: Einsatz von Scan- und Sortier-Operator bei einem Sort-Merge-Verbund

```
OPEN SCAN (R1, SB1, STB1)        /* SCB1 */
SORT R1 INTO S1 USING SCAN(SCB1)
CLOSE SCAN(SCB1)
OPEN SCAN (R2, SB2, STB2)        /* SCB2 */
SORT R2 INTO S2 USING SCAN(SCB2)
CLOSE SCAN(SCB2)
OPEN SCAN (S1, BOS, EOS)         /* SCB3 */
OPEN SCAN (S2, BOS, EOS)         /* SCB4 */
WHILE (NOT FINSHED)
DO
        FETCH TUPLE (SCB3, NEXT, NONE)
        FETCH TUPLE (SCB4, NEXT, NONE)
        ...
END
```

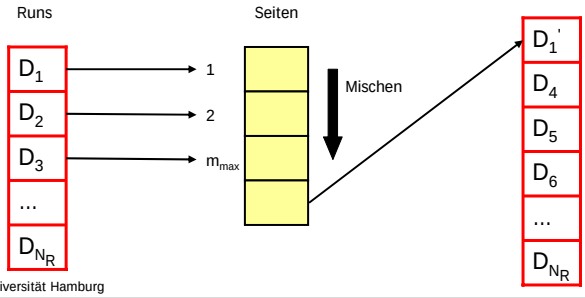
Externes Sortieren

- Problem:
  - Zu sortierende Datenmenge (n Sätze) viel größer als der zum Sortieren verfügbare HSP (q Sätze)
- Wie wird sortiert?
  - Lesen der Eingabedaten aus Datei D und Schreiben sog. Runs nach D_i
  - Erzeugung der Runs ($N_R \geq 1$) mit einem internen Sortierverfahren
  - Mischen aller N_R anfänglichen Runs erforderlich



Externes Sortieren: Mischen

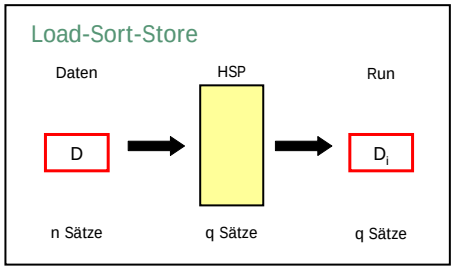
- m-Wege-Mischen bei Magnetplatten
  - Es stehen m_{max} + 1 Plätze (Seiten) im HSP zur Verfügung
  - m_{max} bestimmt die maximale Mischordnung
  - N_R anfängliche Runs sind i.A. über mehrere Magnetplatten verteilt
  - Jeder Run wird nach Möglichkeit sequentiell geschrieben
  - Er kann jederzeit wahlfrei gelesen werden
  - Typischerweise sind die anfänglichen Runs gleich lang

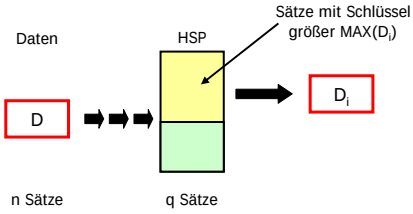


Externes Sortieren: Runs erstellen

- Erzeugung sog. Runs ($N_R \geq 1$) mit einem internen Sortierverfahren
- Einfache Vorgehensweise: Load-Sort-Store
- Geeignete Sortierverfahren?

- Ziel: Möglichst umfangreiche Runs erstellen.





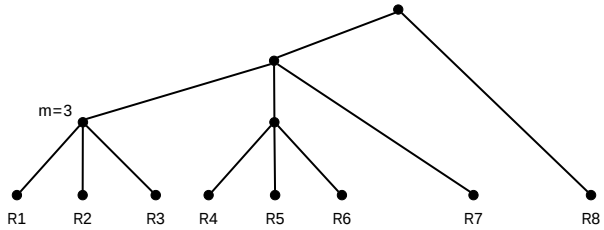
MAX(D_i): Größter nach D_i geschriebener Schlüsselwert

1. q Sätze in HSP lesen und sortieren
2. Nächsten Satz d aus D lesen
3. Sätze mit Schlüsseln größer MAX(D_i) nach D_i schreiben; solange bis ausreichend Platz für d in HSP
4. Sobald keine Sätze mehr ausgeschrieben werden können:
   D_i schließen und mit D_{i+1} fortfahren
5. D_i enthält im Mittel 2q Sätze

Quelle: [LG98]

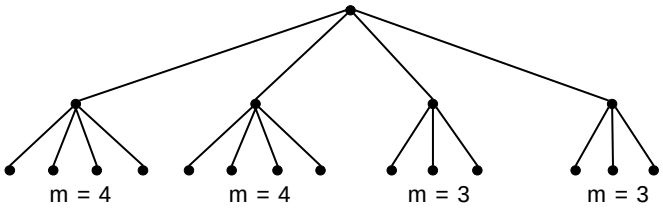
Externes Sortieren

- Optimale Mischbäume
  - Optimierung der Zugriffsbewegungen auf den Magnetplatten ist sehr schwierig: wird gewöhnlich nicht berücksichtigt
  - Ziel: Minimierung der E/A und der Vergleiche
  - einfach bei idealen Anzahlen N_R
    $N_R = m_{max}^{\ p}$
  - Es ergibt sich ein vollständiger Mischbaum der Höhe p vom Grad m_{max}
- Wann sind unvollständige Mischbäume optimal?
  Z.B.:



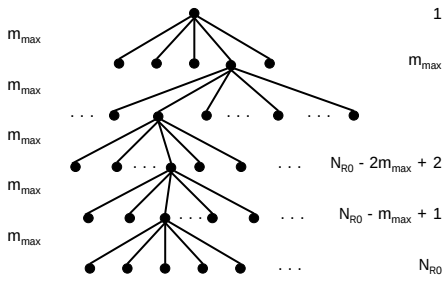
Externes Sortieren

- Allgemein gilt: $N_R \leq (m_{max})^{p_{min}}$
- Annahmen:
  - nur 2 Dateien für Ein-/Ausgabe, ungewichtete Runs
- Heuristik 1: Harmonisiere Mischbaum
  - Bestimme p_{min}
  - Finde kleinstes m, so dass gilt: $N_R \leq m^{p_{min}}$
  - Wende nur Mischordnungen von m und m-1 an.
- Beispiel: $N_R = 14$, $m_{max} = 8$



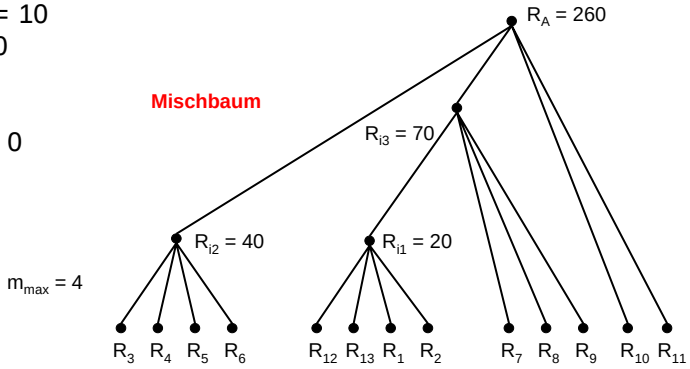
Externes Sortieren

- Annahmen:
  - keine Beschränkung für Ein-/ Ausgabe, gewichtete Runs
- Heuristik 2: Minimiere Anzahl der Ein-/ Ausgaben und Vergleiche
  - $N_R \leq (m_{max} - 1)*p + 1$; bestimme minimales p
  - Erzeuge zusätzliche leere Runs, so dass gilt: $N_{R0} = (m_{max} - 1)*p + 1$
  - Wähle bei jedem Durchlauf jeweils die m_{max} kürzesten Runs und bilde daraus einen neuen Run, bis ein Run übrig bleibt.
- Schema:



Externes Sortieren

- Annahmen: keine Beschränkung für Ein-/Ausgabe, gewichtete Runs
- Beispiel für Heuristik 2:
  $N_R = 11$, $m_{max} = 4 \Rightarrow p = 4$, $N_{R0} = 13$
- Längen der Runs:
  R1, ..., R7 = 10
  R8, R9 = 20
  R10 = 50
  R11 = 100
  R12, R13 = 0



Zusammenfassung

- Abbildung von externen Sätzen
  - Optionen für die Satzspeicherung
  - Trennung von internen und externen Sätzen und flexible Abbildungskonzepte erforderlich
- Transaktionsbezogene Kontroll- und Überwachungsaufgaben
  - Sie verlangen einen schichtenübergreifenden Informationsfluss
  - Lastkontrolle und -balancierung ist komplexes Forschungsthema
- Satzorientierte Schnittstelle
  - Aufgaben des Subschema-Konzeptes bei satzorientierten Datenmodellen
  - Optionen für die Satzspeicherung, vor allem Partitionierung
  - Verarbeitungskonzepte: kontextfreie Operationen, Navigation, Sortierung
  - Implizites und explizites Cursorkonzept
- Scan-Technik
  - Scan-Technik zur satzweisen Navigation auf Zugriffspfaden
  - flexibler Einsatz durch Start-, Stopp- und Suchbedingung sowie Suchrichtung
- Sortierkomponente
  - wichtig zur Implementierung relationaler Operationen
  - große Tabellen (Relationen) erfordern Sortier-/Mischverfahren

Literatur zu diesem Kapitel

[Gra93] Goetz Graefe: Query Evaluation Techniques for Large Databases. ACM Comput. Surv. Vol. 25, No. 2, 1993.

[LG98] P.-A. Larson, G. Graefe: Memory Management during Run Generation in External Sorting. ACM Conf. 1998.