

9. Mengenorientierte DB-Schnittstelle

- **Ziele**

- Darstellung der prinzipiellen Schritte der Übersetzung von mengenorientierten DB-Sprachen
- Realisierung der verschiedenen Optimierungskonzepte

- **Übersetzungsverfahren für Datenbanksprachen**

- **Anfrageoptimierung**

- Standardisierung und Vereinfachung
- Restrukturierung und Transformation

- **Erstellung und Auswahl von Ausführungsplänen**

- „Fatale“ Annahmen
- Bestimmung des „optimalen“ Ausführungsplans
- Strategien zur Reduzierung des Suchraums

- **Berechnung der Zugriffskosten**

- Kostenmodell
- Berechnungsgrundlagen

- **Übersetzung vs. Interpretation**

- **Beispiele**

Logische Datenstrukturen

- **Charakterisierung der Abbildung**

```
SELECT PNR, ABT-NAME
FROM ABTEILUNG, PERS, FAEHIGKEIT
WHERE BERUF = 'PROGRAMMIERER' &
      FAEHIGKEIT.FA-NR = PERS.FA-NR &
      PERS.ABT-NR = ABTEILUNG.ABT-NR
```

Abbildungsfunktionen

- Sichten ↔ Basistabellen
- Relationale Ausdrücke ↔ Logische Zugriffspfade
- Satzmenge n ↔ Einzelne Sätze, Positionsanzeiger

FETCH FAEHIGKEIT USING ...

FETCH NEXT PERS ...

FETCH OWNER WITHIN ...

- **Eigenschaften der oberen Schnittstelle**

- Zugriffspfad-unabhängiges (relationales) Datenmodell
- Alle Sachverhalte und Beziehungen werden durch Werte dargestellt
- Nicht-prozedurale (deskriptive) Anfragesprachen
- Zugriff auf Satzmenge n

Anfrageoptimierung¹

- Von der Anfrage (Was?) zur Auswertung (Wie?)

↳ Ziel: kostengünstiger Auswertungsweg

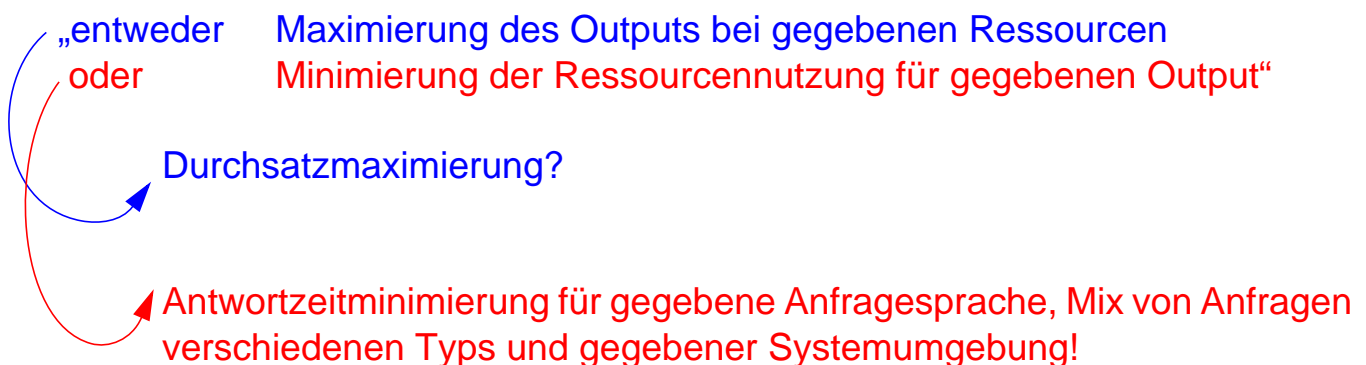
- Einsatz einer großen Anzahl von Techniken und Strategien

- logische Transformation von Anfragen
- Auswahl von Zugriffspfaden
- optimierte Speicherung von Daten auf Externspeichern

- Schlüsselproblem

- genaue Optimierung ist im allgemeinen „nicht berechenbar“
- Fehlen von genauer statistischer Information
- breiter Einsatz von Heuristiken (Daumenregeln)

- Optimierungsziel



¹ Jarke, M., Koch, J.: Query Optimization in Database Systems, in: ACM Comp. Surveys 16:2, 1984, pp. 111-152

Anfrageoptimierung (2)

- Welche Kosten sind zu berücksichtigen?

- **Kommunikationskosten**
(# der Nachrichten, Menge der zu übertragenden Daten)
 - ↳ verteilte DBS!
- **Berechnungskosten** (CPU-Kosten, Pfadlängen)
- **E/A-Kosten** (# der physischen Referenzen)
- **Speicherungskosten** (temporäre Speicherbelegung im DB-Puffer und auf Externspeichern)

↳ Kostenarten sind nicht unabhängig voneinander

↳ in zentralisierten DBS oft
„gewichtete Funktion von Berechnungs- und E/A-Kosten“

- Wie wird am besten vorgegangen?

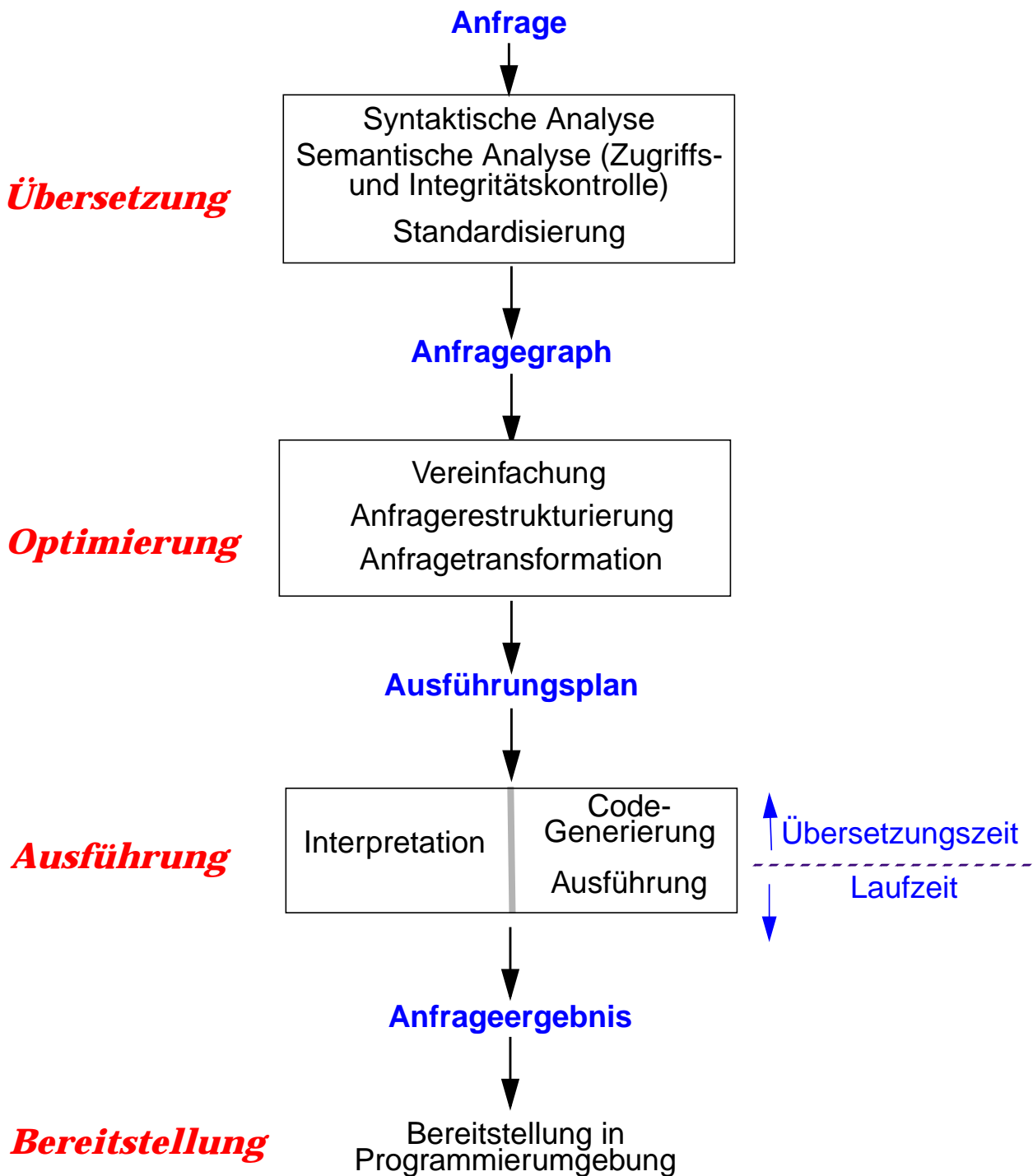
Schritt 1: Finde nach Übersetzung geeignete Interndarstellung für die Anfrage (Anfragegraph)

Schritt 2: Wende die logische Restrukturierung auf den Anfragegraph an

Schritt 3: Bilde die restrukturierte Anfrage auf alternative Folgen von Planoperatoren (Transformation) ab
(↳ Mengen von Ausführungsplänen)

Schritt 4: Berechne Kostenvoranschläge für jeden Ausführungsplan und wähle den billigsten aus

Anfrageoptimierung – Überblick



Standardisierung einer Anfrage

- **Standardisierung**

- Wahl einer Normalform

z.B. konjunktive Normalform

$(A_{11} \text{ OR } \dots \text{ OR } A_{1n}) \text{ AND } \dots \text{ AND } (A_{m1} \text{ OR } \dots \text{ OR } A_{mn})$

- Verschiebung von Quantoren

- **Umformungsregeln für Boolesche Ausdrücke**

Kommutativregeln											
A	OR	B	\Leftrightarrow B OR A								
A	AND	B	\Leftrightarrow B AND A								
Assoziativregeln											
(A	OR	B)	OR C	\Leftrightarrow	A	OR	(B	OR	C)		
(A	AND	B)	AND C	\Leftrightarrow	A	AND	(B	AND	C)		
Distributivregeln											
A	OR	(B	AND C)	\Leftrightarrow	(A	OR	B)	AND	(A	OR	C)
A	AND	(B	OR C)	\Leftrightarrow	(A	AND	B)	OR	(A	AND	C)
De Morgan'sche Regeln											
NOT (A	AND B)	\Leftrightarrow	NOT (A	OR	NOT (B)						
NOT (A	OR B)	\Leftrightarrow	NOT (A	AND	NOT (B)						
Doppelnegationsregel											
NOT (NOT (A))	\Leftrightarrow	A									

- **Idempotenzregeln für Boolesche Ausdrücke**

A	OR	A	\Leftrightarrow	A	
A	AND	A	\Leftrightarrow	A	
A	OR	NOT (A)	\Leftrightarrow	TRUE	
A	AND	NOT (A)	\Leftrightarrow	FALSE	
A	AND	(A	OR B)	\Leftrightarrow	A
A	OR	(A	AND B)	\Leftrightarrow	A
A	OR	FALSE	\Leftrightarrow	A	
A	OR	TRUE	\Leftrightarrow	TRUE	
A	AND	FALSE	\Leftrightarrow	FALSE	

Vereinfachung einer Anfrage

- **Äquivalente Ausdrücke**

- können einen unterschiedlichen Grad an Redundanz besitzen
- Behandlung/Eliminierung gemeinsamer Teilausdrücke

$$\begin{array}{l} (A_1 = a_{11} \text{ OR } A_1 = a_{12}) \\ \text{AND } (A_1 = a_{12} \text{ OR } A_1 = a_{11}) \end{array}$$

- Vereinfachung von Ausdrücken, die an "leere Tabellen" gebunden sind

- **Konstanten-Propagierung**

$$A \text{ op } B \text{ AND } B = \text{const.}$$

→ $A \text{ op const.}$

- **Nicht-erfüllbare Ausdrücke**

$$A \geq B \text{ AND } B > C \text{ AND } C \geq A$$

→ $A > A \rightarrow \text{false}$

- **Nutzung von Integritätsbedingungen (IB)**

- IB sind wahr für alle Tupel der betreffenden Tabelle
- A ist Primärschlüssel: $\pi_A \rightarrow$ keine Duplikateliminierung erforderlich
- Regel: FAM-STAND = 'verh.' AND STEUERKLASSE ≥ 3

→ Ausdruck: $(\text{FAM-STAND} = \text{'verh.'} \text{ AND } \text{STEUERKLASSE} = 1) \rightarrow \text{false}$

- **Verbesserung der Auswertbarkeit**

- Hinzufügen einer IB zur WHERE-Bedingung verändert den Wahrheitswert eines Auswahlausdrucks nicht

→ Einsatz zur verbesserten Auswertung (knowledge-based query processing)

- einfachere Auswertungsstruktur, jedoch effiziente Heuristiken benötigt

Anfragerestrukturierung

• Wichtigste Regeln für Restrukturierung und Transformation

- Frühzeitige Ausführung von Selektion (σ) und Projektion (π) ohne Duplikateliminierung
- Unäre Operatorfolgen (wie σ und π) zu einer Operation zusammenzufassen
- Gleiche Teile im AG nur einmal auswerten
- Binäre Operatorfolgen (wie \cap , \cup , $-$, \times , $|\times|$): Zwischenergebnisse minimieren

➔ selektive Operationen (σ , π) vor konstruktiven Operationen (\times , $|\times|$)

• Zusammenfassung von Operationsfolgen

$$R1: \pi_{A_n}(\dots\pi_{A_2}(\pi_{A_1}(\text{Tab}))\dots) \Leftrightarrow \pi_{A_n}(\text{Tab})$$

$$R2: \sigma_{p_n}(\dots\sigma_{p_2}(\sigma_{p_1}(\text{Tab}))\dots) \Leftrightarrow \sigma_{p_1 \text{ AND } p_2 \dots \text{ AND } p_n}(\text{Tab})$$

• Restrukturierungsalgorithmus

- (1) Zerlege komplexe Verbundprädikate so, daß sie binären Verbunden zugeordnet werden können (Bilden von binären Verbunden).
- (2) Teile Selektionen mit mehreren Prädikatstermen in separate Selektionen mit jeweils einem Prädikatsterm auf.
- (3) Führe Selektionen so früh wie möglich aus, d. h., schiebe Selektionen hinunter zu den Blättern des AG (selection push-down).
- (4) Fasse einfache Selektionen zusammen, so daß aufeinanderfolgende Selektionen (derselben Tabelle) zu einer verknüpft werden.
- (5) Führe Projektionen ohne Duplikateliminierung so früh wie möglich aus, d. h., schiebe sie soweit wie möglich zu den Blättern des AG hinunter (projection push-down).
- (6) Fasse einfache Projektionen (einer Tabelle) zu einer Operation zusammen.

Anfragetransformation

- **Aufgabe**

Zusammenfassung von logischen Operatoren (Ein- und Zwei-Variablen-Ausdrücke) und ihre Ersetzung durch **Planoperatoren**

- **Typische Planoperatoren in relationalen Systemen**

- **auf einer Tabelle:**

Selektion, Projektion, Sortierung, Aggregation, Änderungsoperationen (IUD) und ACCESS zum Zugriff auf Basistabellen

+

Erweiterungen: Rekursion, Gruppierung, . . .

- **auf zwei Tabellen:**

Verbund- und Mengen-Operationen, Kartesisches Produkt.

- **Anpassungen im AG zum effektiven Einsatz von Planoperatoren**

(1) Gruppierung von direkt benachbarten Operatoren;

z. B. lassen sich durch einen speziellen Planoperator ersetzen:
Verbund (oder Kartesisches Produkt) mit Selektionen und/oder Projektionen auf den beteiligten Tabellen.

(2) Bestimmung der Verknüpfungsreihenfolge bei binären Operationen;

dabei sollen die minimalen Kosten für die Operationsfolge erzielt werden. Als Heuristik ist dazu die Größe der Zwischenergebnisse zu minimieren, d. h., die kleinsten (Zwischen-)Tabellen sind immer zuerst zu verknüpfen.

(3) Erkennung gemeinsamer Teilbäume,

die dann nur jeweils einmal zu berechnen sind. Allerdings steht dieser Einsparung die Zwischenspeicherung der Ergebnistabelle gegenüber.

Bewertung von Ausführungsplänen – Grundsätzliche Probleme

- **Anfrageoptimierung beruht i.a. auf zwei „fatalen“ Annahmen**

1. Alle Datenelemente und alle Attributwerte sind gleichverteilt
2. Suchprädikate in Anfragen sind unabhängig

➔ **Beide Annahmen sind (im allgemeinen Fall) falsch!**

- **Beispiel**

(GEHALT \geq '100K') AND (ALTER BETWEEN 20 AND 30)

Bereiche: 10K – 1M 20 – 65

1. Lineare Interpolation:

2. Multiplikation von Wahrscheinlichkeiten

- **Lösung?**

- Verbesserung der Statistiken/Heuristiken (Histogramme)
- Berechnung/Bewertung von noch mehr Ausführungsplänen?

Obwohl die Kostenabschätzungen meist falsch sind . . .

Erstellung und Auswahl von Ausführungsplänen

- **Eingabe:**

- optimierter Anfragegraph (AG)
- existierende Speicherstrukturen und Zugriffspfade
- Kostenmodell

- **Ausgabe: optimaler Ausführungsplan (oder wenigstens „gut“)**

- **Vorgehensweise:**

1. **Generiere alle „vernünftigen“ logischen Ausführungspläne** zur Auswertung der Anfrage
2. **Vervollständige die Ausführungspläne** durch Einzelheiten der physischen Datenrepräsentation (Sortierreihenfolge, Zugriffspfadmerkmale, statistische Information)
3. **Wähle den billigsten Ausführungsplan** gemäß dem vorgegebenen Kostenmodell aus

➔ Alternative Ausführungspläne für einen AG entstehen vor allem dadurch, daß für jeden Planoperator verschiedene Methoden (Implementierungen) vorliegen, und daß Operationsreihenfolgen (z. B. bei Mehrfachverbunden) variiert werden können. So bilden sich bei komplexen Anfragen sehr große Suchräume mit Alternativen (z. B. 10^{70} mögliche Ausführungspläne bei einer Anfrage mit 15 Verbunden).

- **Generierung durch Anfrageoptimierer**

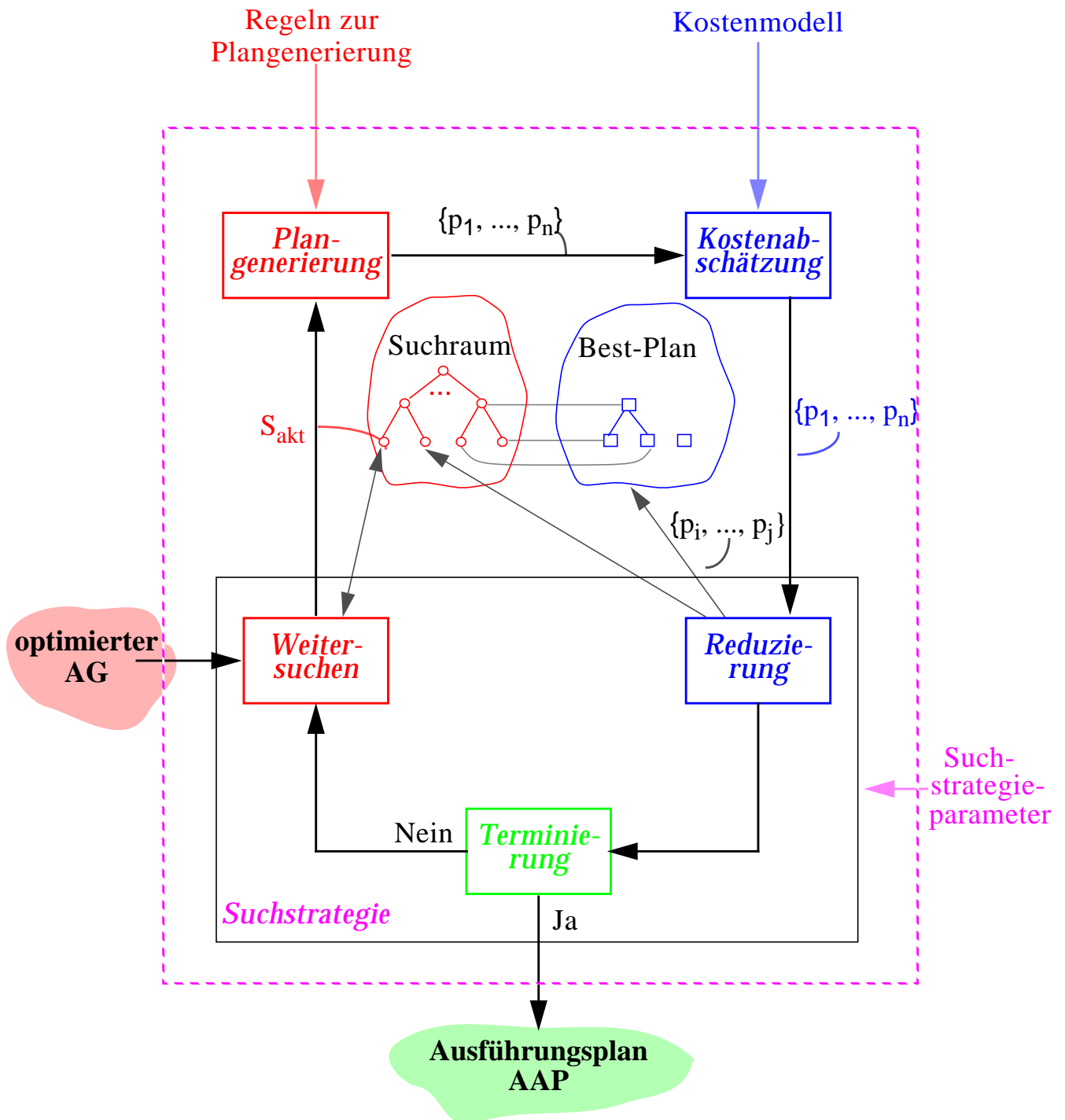
➔ **kleine Menge der Pläne, die den optimalen Plan enthält**

➔ **Einschränkung durch Heuristiken**

- hierarchische Generierung basierend auf dem Schachtelungskonzept von SQL
- Zerlegung in eine Menge von Teilanfragen mit höchstens Zwei-Variablen-Ausdrücken

Erstellung und Auswahl von Ausführungsplänen (2)²

- Zusammenspiel der Komponenten



AAP: Anfrageausführungsplan (engl. QEP: Query Evaluation Plan)

2 Mitschang, B.: Anfrageverarbeitung in Datenbanksystemen: Entwurfs- und Implementierungskonzepte, Reihe Datenbanksysteme, Vieweg, 1995

Erstellung und Auswahl von Ausführungsplänen (3)

- **Plangenerierung** soll

- immer und möglichst schnell den „optimalen“ Plan finden
- mit einer möglichst kleinen Anzahl generierter Pläne auskommen

- **Suchstrategien**

- voll-enumerativ
- beschränkt-enumerativ
- zufallsgesteuert

↳ **Reduzierung:** Bestimmte Suchpfade zur Erstellung von AAPs werden nicht weiter verfolgt

- **Kostenabschätzung**

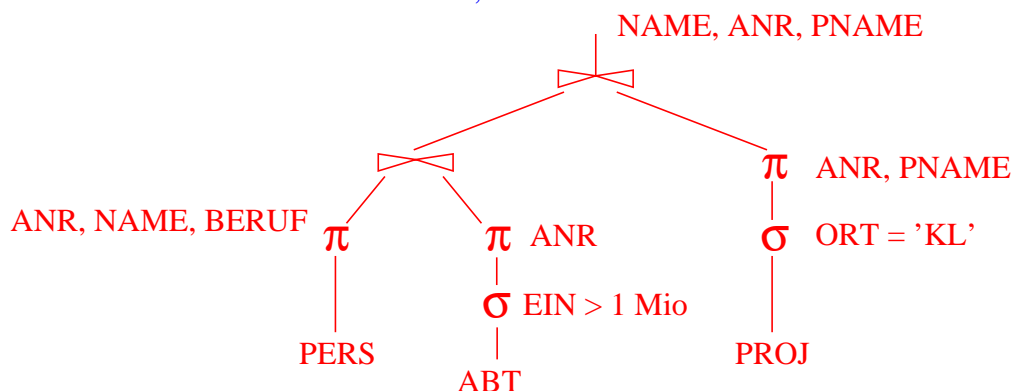
- verlangt hinreichend genaues Kostenmodell
- wird bei allen Suchverfahren inkrementell durchgeführt

- **Problemdarstellung – Beispiel**

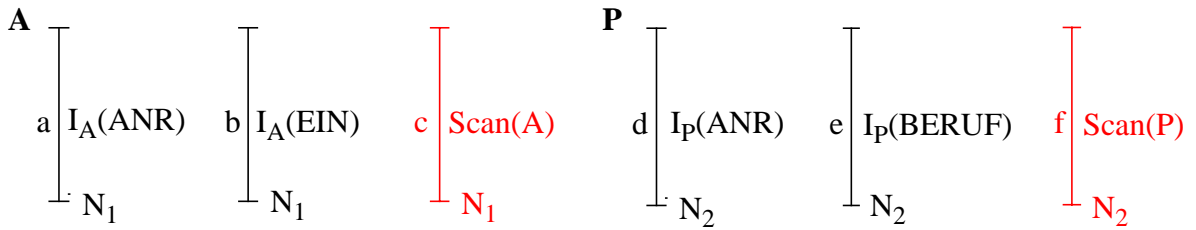
SQL:

```
SELECT  P.NAME, P.BERUF, J.PNAME
FROM    PERS P, ABT A, PROJ J
WHERE   A.EIN > 1000000 AND J.ORT = 'KL'
        AND  A.ANR = P.ANR AND A.ANR = J.ANR;
```

Zugehöriger Anfragegraph

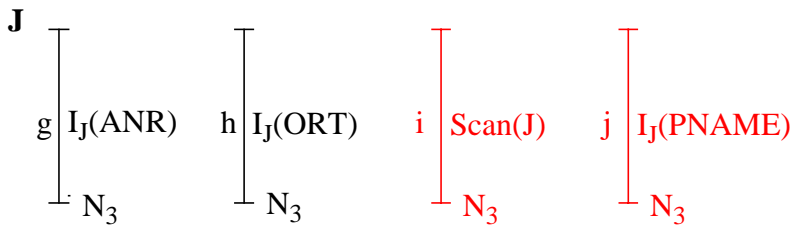


Erstellung und Auswahl von Ausführungsplänen (4)



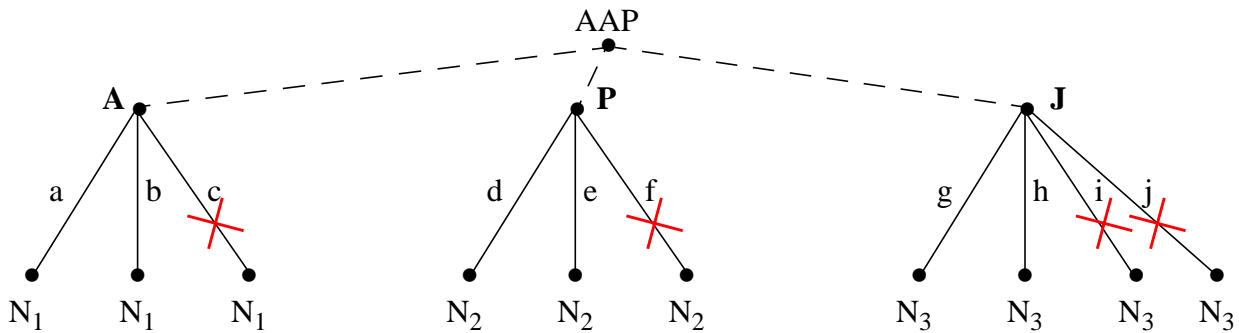
Kostenabschätzung: $C(A.ANR) \dots$

$C(P.ANR) \dots$



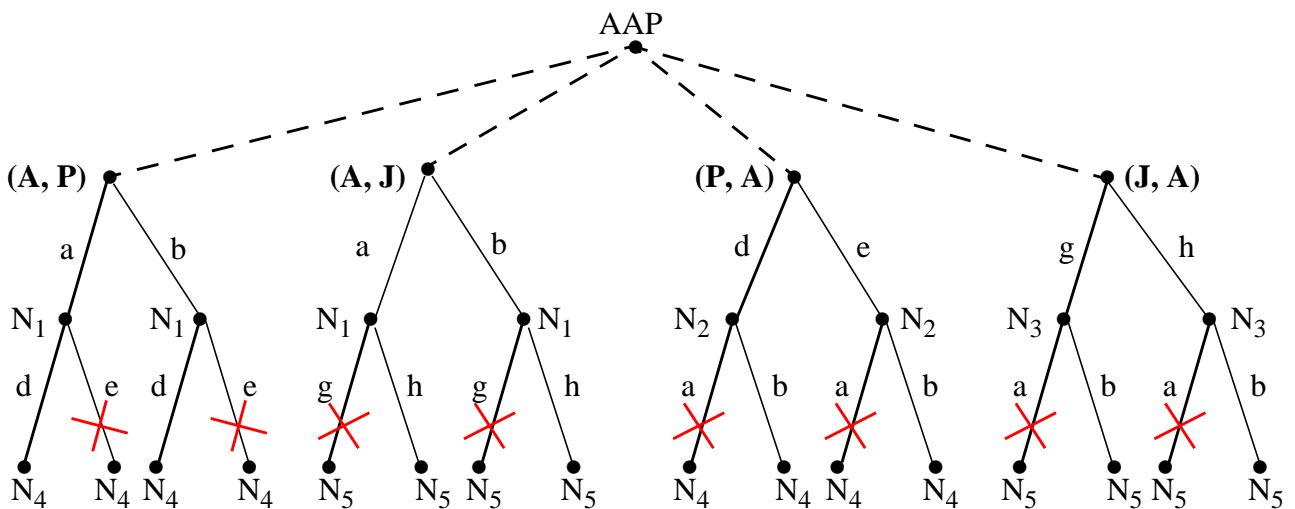
Kostenabschätzung: $C(J.ANR) \dots$

a) mögliche Zugriffspfade für die einzelnen Tabellen



b) Lösungsbaum für einzelne Tabellen:

Reduzierung durch Abschneiden von Teilbäumen



c) Erweiterter Lösungsbaum für den Nested-Loop-Verbund mit der zweiten Tabelle

Kostenabschätzung pro Pfad: z. B. durch $C(C(A.ANR) + C(P.ANR) + \text{Verbundkosten})$

Berechnung der Zugriffskosten

- **Optimiereraufgabe**

- erstellt Kostenvoranschlag für jeden „aussichtsreichen“ Ausführungsplan
- Einsatz einer gewichteten Kostenformel:

$$C = \#physischer \text{ Seitenzugriffe} + W * (\#Aufrufe \text{ des Zugriffssystems})$$

- gewichtetes Maß für E/A- und CPU-Auslastung
 - W ist das Verhältnis des Aufwandes von ZS-Aufruf zu Seitenzugriff
- **Ziel der Gewichtung: Minimierung der Kosten in Abhängigkeit des Systemzustandes**
 - System "I/O-bound": → sehr kleiner W-Wert

$$W_{I/O} = \frac{\#Instr. \text{ pro ZS-Aufruf}}{\#Instr. \text{ pro E/A} + \text{Zugriffszeit} \cdot \text{MIPS-Rate}}$$

$$\text{Bsp. } W_{I/O} = \frac{1000 \text{ I.}}{2500 \text{ I.} + 12 \text{ msec} \cdot 10^8 \text{ I./sec}} = 0,0008$$

- System "CPU-bound": → relativ großer W-Wert

$$W_{CPU} = \frac{\#Instr. \text{ pro ZS-Aufruf}}{\#Instr. \text{ pro E/A}}$$

$$\text{Bsp. } W_{CPU} = \frac{1000}{2500} = 0,4$$

Kostenmodell – statistische Werte

- **Statistische Größen für Segmente:**

M_S Anzahl der Datenseiten des Segmentes S

L_S Anzahl der leeren Seiten in S

- **Statistische Größen für Tabellen:**

N_R Anzahl der Tupel der Tabelle R (Card(R))

$T_{R,S}$ Anzahl der Seiten in S mit Tupel von R

C_R Clusterfaktor (Anzahl Tupel pro Seite)

- **Statistische Größen pro Index I auf Attributen A einer Tabelle R:**

j_I Anzahl der Attributwerte / Schlüsselwerte im Index
(=Card ($\pi_A(R)$))

B_I Anzahl der Blattseiten (B*-Baum)

...

→ **Statistiken müssen im DB-Katalog gewartet werden**

- **Aktualisierung bei jeder Änderung sehr aufwendig**

- zusätzliche Schreib- und Log-Operationen
- DB-Katalog wird zum Sperr-Engpaß

- **Alternative:**

- Initialisierung der statistischen Werte zum Lade- oder Generierungszeitpunkt von Tabellen und Indexstrukturen
- periodische Neubestimmung der Statistiken durch eigenes Kommando/ Dienstprogramm (DB2: RUNSTATS)

Kostenmodell – Berechnungsgrundlagen

Mit Hilfe der statistischen Werte kann der Anfrageoptimierer jedem Verbundterm im Qualifikationsprädikat einen Selektivitätsfaktor ($0 \leq SF \leq 1$) zuordnen (erwarteter Anteil an Tupel, die das Prädikat erfüllen): $Card(\sigma_p(R)) = SF(p) \cdot Card(R)$

• Selektivitätsfaktor SF bei:

$$\begin{array}{l}
 A_i = a_i \\
 A_i = A_k \\
 A_i \geq a_i \quad (\text{oder } A_i > a_i) \\
 A_i \text{ BETWEEN } a_i \text{ AND } a_k \\
 A_i \text{ IN } (a_1, a_2, \dots, a_r)
 \end{array}
 \quad
 \begin{array}{l}
 SF = \begin{cases} 1/j_i & \text{wenn Index auf } A_i \\ 1/10 & \text{sonst} \end{cases} \\
 SF = \begin{cases} 1 / \text{Max}(j_i, j_k) & \text{wenn Index auf } A_i, A_k \\ 1 / j_i & \text{wenn Index auf } A_i \\ 1 / j_k & \text{wenn Index auf } A_k \\ 1/10 & \text{sonst} \end{cases} \\
 SF = \begin{cases} (a_{\max} - a_i) / (a_{\max} - a_{\min}) & \text{wenn Index auf } A_i \\ & \text{und Wert interpolierbar} \\ 1/3 & \text{sonst} \end{cases} \\
 SF = \begin{cases} (a_k - a_i) / (a_{\max} - a_{\min}) & \text{wenn Index auf } A_i \\ & \text{und Wert interpolierbar} \\ 1/4 & \text{sonst} \end{cases} \\
 SF = \begin{cases} r / j_i & \text{wenn Index auf } A_i \text{ und} \\ & SF < 0.5 \\ 1/2 & \text{sonst} \end{cases}
 \end{array}$$

• Berechnung von Ausdrücken

- $SF(p(A) \wedge p(B)) = SF(p(A)) \cdot SF(p(B))$
- $SF(p(A) \vee p(B)) = SF(p(A)) + SF(p(B)) - SF(p(A)) \cdot SF(p(B))$
- $SF(\neg p(A)) = 1 - SF(p(A))$

• Join-Selektivitätsfaktor (JSF)

- $Card(R \bowtie S) = JSF * Card(R) * Card(S)$
- bei (N:1)-Verbunden (verlustfrei): $Card(R \bowtie S) = \text{Max}(Card(R), Card(S))$

Zusammenfassung

- **Interpretation einer DB-Anweisung**

- Allgemeines Programm (Interpreter) akzeptiert Anweisungen der DB-Sprache als Eingabe und erzeugt mit Hilfe von Aufrufen des Zugriffssystems Ergebnis
- hoher Aufwand zur Laufzeit (v.a. bei wiederholter Anweisungsausführung)

- **Übersetzung, Code-Erzeugung und Ausführung einer DB-Anweisung**

- Für jede DB-Anweisung wird ein zugeschnittenes Programm erzeugt (Übersetzungszeit), das zur Laufzeit abgewickelt wird und dabei mit Hilfe von Aufrufen des Zugriffssystems das Ergebnis ableitet
- Übersetzungsaufwand wird zur Laufzeit soweit wie möglich vermieden

- **Anfrageoptimierung: Kernproblem**

der Übersetzung mengenorientierter DB-Sprachen

- **"fatale" Annahmen:**

- Gleichverteilung aller Attributwerte
- Unabhängigkeit aller Attribute

- **Kostenvoranschläge für Ausführungspläne:**

- CPU-Zeit und E/A-Aufwand
- Anzahl der Nachrichten und zu übertragende Datenvolumina (im verteilten Fall)

- gute Heuristiken zur Auswahl von Ausführungsplänen sehr wichtig

- **Kostenmodell**

- Minimierung der Kosten in Abhängigkeit des Systemzustandes
- Problem: Aktualisierung der statistischen Kenngrößen

Beispiel: Einfache Anfrage

- **SQL-Anfrage**

```
SELECT  NAME, GEHALT
FROM    PERS
WHERE   BERUF = 'PROGRAMMIERER'
        AND GEHALT ≥ 100.000
```

- **Vorhandene Zugriffspfade**

- Tabellen-Scan im Segment von PERS
- $I_{PERS}(BERUF)$
- $I_{PERS}(GEHALT)$
- Link von FAEHIGKEIT nach PERS

- **Statistische Kennwerte**

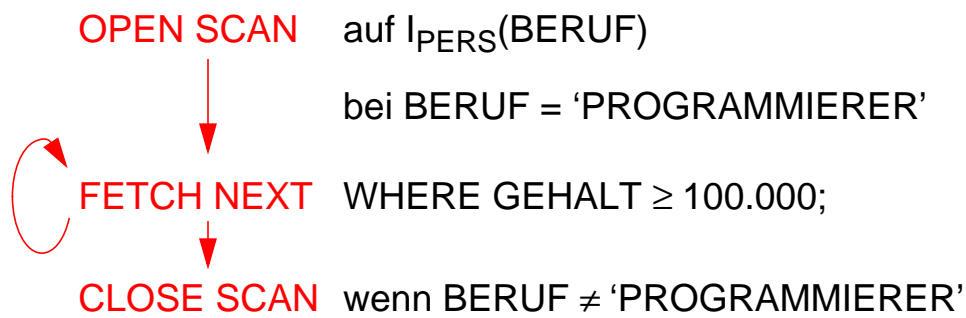
Der Anfrageoptimierer findet folgende Parameter im DB-Katalog:

- N = # der Tupel in Tabelle PERS
- C = durchschn. # von PERS-Tupel pro Seite
- j_i = Index-Kardinalität (#Attributwerte für A_i)
- ...
- + Information über Clusterbildung

- **Annahmen**

- Jeder 10. Programmierer hat ein Gehalt > 100 K
- Jeder 2. Angestellte mit Gehalt > 100 K ist Programmierer

Methode 1: Scan über I_{PERS}(BERUF)



• Kosten:

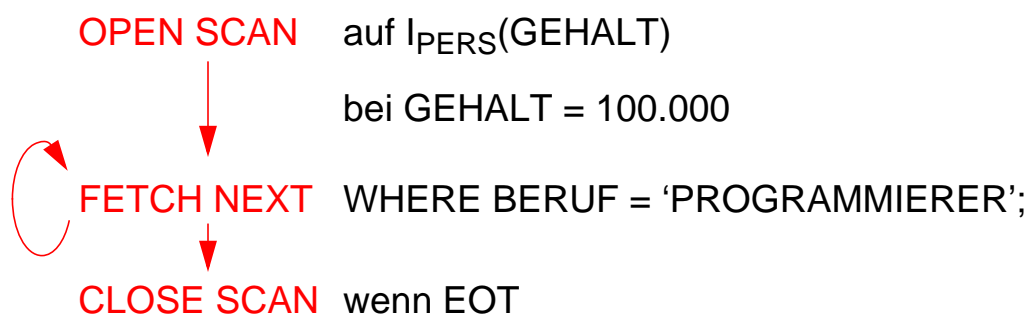
- Clusterbildung auf I_{PERS}(BERUF)

$$K \approx 3 + \frac{N}{C \cdot j_{\text{BERUF}}} + W \cdot \frac{N}{j_{\text{BERUF}} \cdot 10}$$

- keine Clusterbildung

$$K \approx 3 + \frac{N}{j_{\text{BERUF}}} + W \cdot \frac{N}{j_{\text{BERUF}} \cdot 10}$$

Methode 2: Scan über I_{PERS}(GEHALT)



• Kosten:

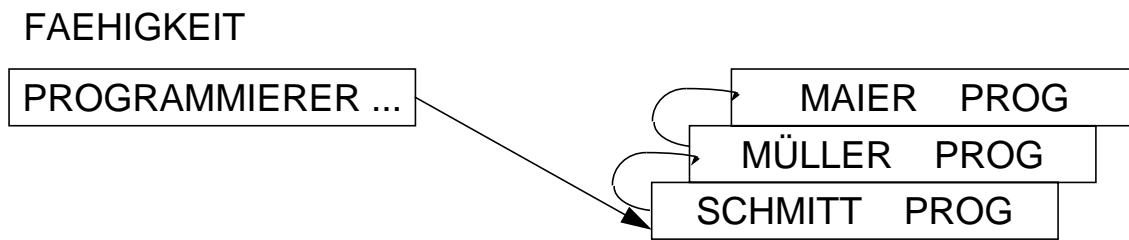
- Clusterbildung auf I_{PERS}(GEHALT)

$$K \approx 3 + \frac{N}{3 \cdot C} + W \cdot \frac{N}{3 \cdot 2}$$

- keine Clusterbildung

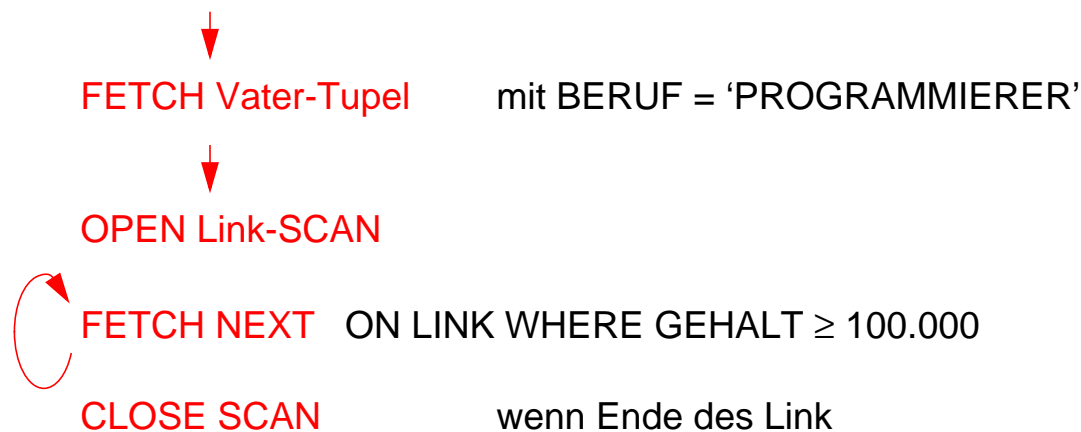
$$K \approx 3 + \frac{N}{3} + W \cdot \frac{N}{3 \cdot 2}$$

Methode 3: Benutze einen hierarchischen Zugriffspfad (LINK) von einer anderen Tabelle



- Annahme:**

Schneller Zugriff auf Tabelle FAEHIGKEIT als Einstieg in LINK möglich, z. B. über $I_{FAEHIGKEIT}(BERUF)$



- Kosten:**

- Clusterbildung auf Link

$$K \approx 3 + \frac{N}{C \cdot j_{BERUF}} + W \cdot \frac{N}{j_{BERUF} \cdot 10}$$

- keine Clusterbildung

$$K \approx 3 + \frac{N}{j_{BERUF}} + W \cdot \frac{N}{j_{BERUF} \cdot 10}$$