9. Mengenorientierte DB-Schnittstelle

- **Übersetzungsverfahren für Datenbanksprachen**
  - Übersetzung vs. Interpretation
- **Anfrageoptimierung**
  - Anfragedarstellung
  - Standardisierung und Vereinfachung
  - Restrukturierung und Transformation
- **Erstellung und Auswahl von Ausführungsplänen**
- **Berechnung der Zugriffskosten Kostenmodell)**
- **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	↔ Basisrelationen
- Relationale Ausdrücke	↔ Logische Zugriffspfade
- Satzmengen	↔ 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 Satzmengen

Anfrageoptimierung[*]

- 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**

  „entweder Maximierung des Outputs bei gegebenen Ressourcen
  oder Minimierung der Ressourcennutzung für gegebenen Output“

  Durchsatzmaximierung ?

  Antwortzeitminimierung für eine gegebene Anfragesprache, einem Mix von Anfragen verschiedenen Typs und einer gegebenen Systemumgebung!

---

* Jarke, M., Koch, J.: Query Optimization in Database Systems, in: ACM Computing Surveys 16:2, June 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?**

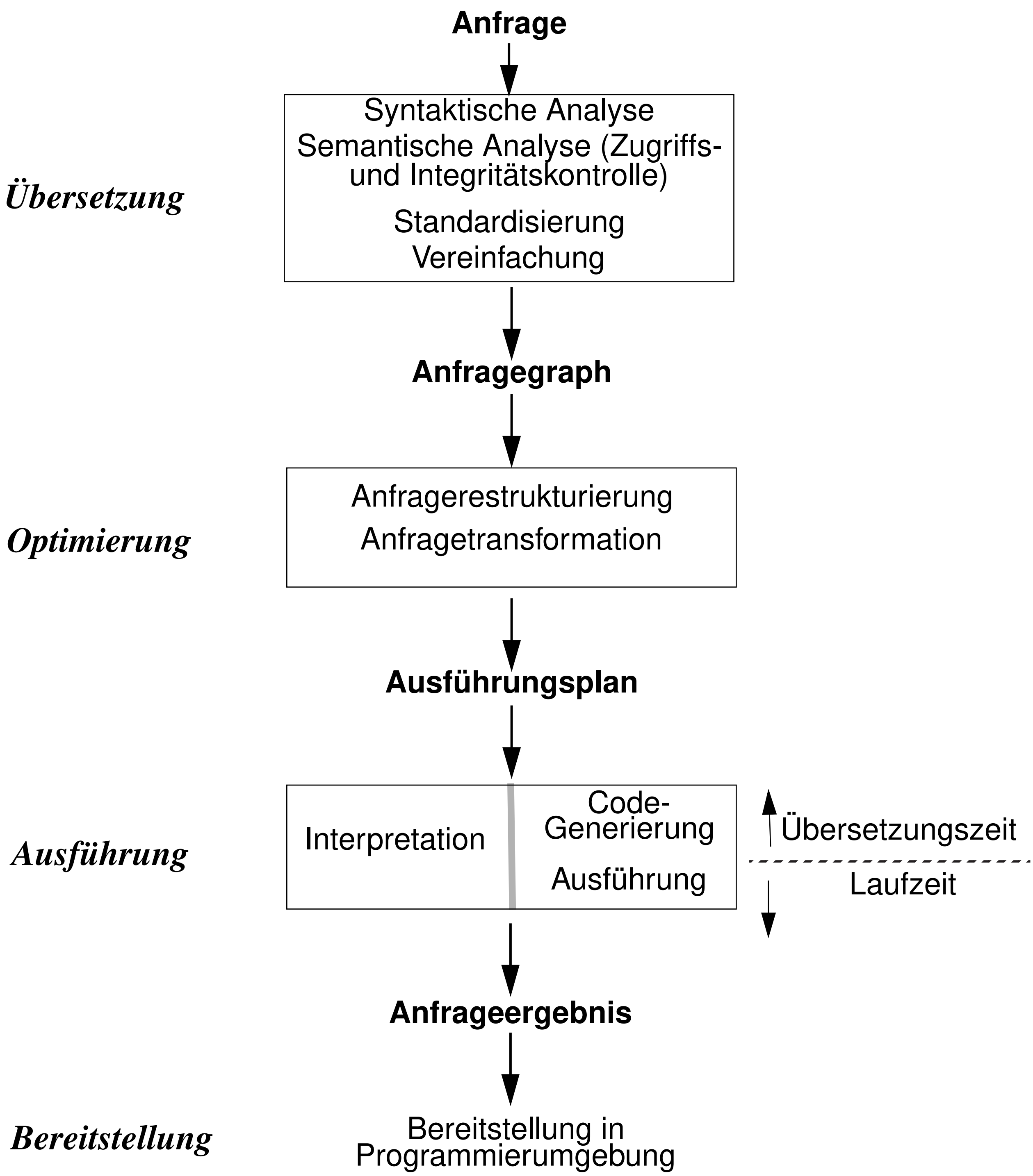
  Schritt 1: Finde nach Übersetzung geeignete Interndarstellung für die Anfrage (Anfragegraph)

  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

    (A11 OR ... OR A1n) AND ... AND (Am1 OR ... OR Amn)
  - Verschiebung von Quantoren

- **Umformungsregeln für Boole'sche Ausdrücke**

Kommutativregeln

A	**OR**	B			⇔	B	**OR**	A				
A	**AND**	B			⇔	B	**AND**	A				

Assoziativregeln

(A	**OR**	B)	**OR**	C	⇔	A	**OR**	(B	**OR**	C)		
(A	**AND**	B)	**AND**	C	⇔	A	**AND**	(B	**AND**	C)		

Distribtivregeln

A	**OR**	(B	**AND**	C)	⇔	(A	**OR**	B)	**AND**	(A	**OR**	C)
A	**AND**	(B	**OR**	C)	⇔	(A	**AND**	B)	**OR**	(A	**AND**	C)

De Morgan'sche Regeln

NOT (A	**AND**	B)	⇔	**NOT** (A)	**OR**	**NOT** (B)
NOT (A	**OR**	B)	⇔	**NOT** (A)	**AND**	**NOT** (B)

Doppelnegationsregel

NOT (**NOT** (A))	⇔	A

- **Idempotenzregeln für Boole'sche Ausdrücke**

A	**OR**	A			⇔	A
A	**AND**	A			⇔	A
A	**OR**	**NOT** (A)			⇔	TRUE
A	**AND**	**NOT** (A)			⇔	FALSE
A	**AND**	(A	**OR**	B)	⇔	A
A	**OR**	(A	**AND**	B)	⇔	A
A	**OR**	**FALSE**			⇔	A
A	**OR**	**TRUE**			⇔	TRUE
A	**AND**	**FALSE**			⇔	FALSE

Vereinfachung einer Anfrage

- **Äquivalente Ausdrücke** können einen unterschiedlichen Grad an Redundanz besitzen

- **Behandlung/Eliminierung gemeinsamer Teilausdrück**e

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

- Vereinfachung von Ausdrücken, die an “leere Relationen” gebunden sind

- **Konstanten-Propagierung**

  A op B AND B = const.

  ➥ A op const.

- **Nicht-erfüllbare Ausdrücke**

  $A \geq B$ AND $B > C$ AND $C \geq A$

  ➥ $A > A \rightarrow$ false

- **Nutzung von Integritätsbedingungen (IB)**

  IB sind wahr für alle Tupel der betreffenden Relation

  - A ist Primärschlüssel: $\pi_A \rightarrow$ keine Duplikateliminierung erforderlich
  - Regel: FAM-STAND = ‘verh.’ AND STEUERKLASSE ≥ 3

  ➥ Ausdruck: (FAM-STAND = ‘verh.’ AND STEUERKLASSE = 1) $\rightarrow$ 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

Interndarstellung einer Anfrage

- **Problematik:**
  Finden eines entsprechenden ***Darstellungsschemas***, mit dem dann geeignete Interndarstellungen einer Anfrage möglich sind.
  - zentrale Datenstruktur für Übersetzung und Optimierung
  - entscheidend für die Effizienz und Erweiterbarkeit des AP

- **Eigenschaften eines guten Darstellungsschemas**
  - **Prozeduralität**

    Externe Anfrage: deskriptiven Form
    (Relationenkalkül oder **SQL-Notation**)

    Interndarstellung: prozedurale Darstellung der Anfrage

    Deskriptive DB-Sprache in eine an die Relationenalgebra angelehnte Darstellung umsetzen:

    ➟ eine **deklarative Beschreibung des Anfrageergebnisses** wird übersetzt in einen **Algorithmus oder Plan**, dargestellt als Folge **von Algebraoperatoren**.

    Diese Vorgehensweise wird von den meisten DBS übernommen, wobei die Menge an verfügbaren (Algebra-)Operatoren von einem zum anderen DBS durchaus differieren kann.

- **Flexibilität**

  Erweiterungen des Datenmodells und der DB-Sprache

  Transformationen im Rahmen des nachfolgenden Optimierungsschritts

  - **Effizienz**

    effiziente Datenstruktur mit geeigneten Zugriffsfunktionen

Interndarstellung einer Anfrage (2)

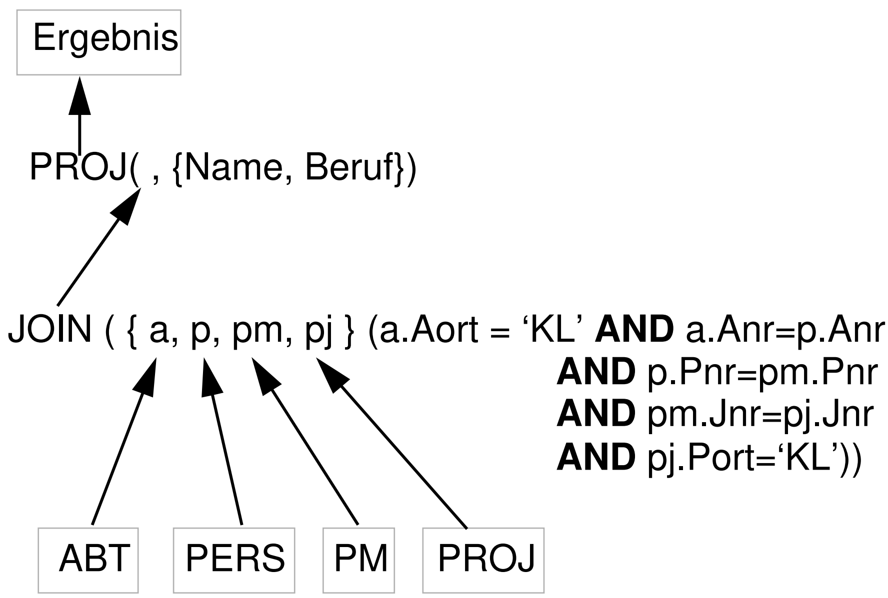
- **Klassen von Darstellungsschemata**
  - lineare oder auch matrixförmige Interndarstellung
    Relationenalgebra
    Relationenkalkül
  - strukturierte Interndarstellung
    Zerlegungsbaum
    Objektgraph
    Operatorgraph

- **Beispiel**

  Finde Name und Beruf von Angestellten, die Projekte in ‘KL’ durchführen und deren zugehörige Abteilung sich ebenfalls in ‘KL’ befindet”

```
SELECT   Name, Beruf
FROM     ABT a, PERS p, PM pm, PROJ pj
WHERE    a.Anr = p.Anr AND a.Aort = ‘KL’ AND
         p.Pnr = pm.Pnr AND
         pm.Jnr = pj.Jnr AND pj.Port = ‘KL’;
```

Operatorgraph



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 $\cap$, $\cup$, –, $\times$, $\bowtie$): Zwischenergebnisse minimieren

  ➥ selektive Operationen (σ, π) vor konstruktiven Operationen ($\times$, $\bowtie$)

- **Zusammenfassung von Operationsfolgen**

  R1: $\pi_{An}(\ldots\pi_{A2}(\pi_{A1}(Rel))\ldots) \Leftrightarrow \pi_{An}(Rel)$

  R2: $\sigma_{pn}(\ldots\sigma_{p2}(\sigma_{p1}(Rel))\ldots)\ldots) \Leftrightarrow \sigma_{p1 \text{AND } p2 \ldots \text{ AND } pn}(Rel)$

- **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 Relation) 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 Relation) zu einer Operation zusammen.

Anfragetransformation

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

- **Typische Planoperatoren in relationalen Systemen**

  - **auf einer Relation:**
    Selektion, Projektion, Sortierung, Aggregation, Änderungsop. (Einfügen, Löschen, Modifizieren) und ACCESS zum Zugriff auf Basisrelationen
    +
    Erweiterungen: Rekursion, Gruppierung, . . .

  - **auf zwei Relationen:**
    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 Relationen.

(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-)Relationen sind immer zuerst zu verknüpfen.

(3) *Erkennung gemeinsamer Teilbäume*,

die dann nur jeweils einmal zu berechnen sind. Allerdings steht dieser Einsparung die Zwischenspeichung der Ergebnisrelation 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 falsch (im allgemeinen Fall)

- **Beispiel**

  (GEHALT ≥ ‘100K’) AND (ALTER BETWEEN 20 AND 30)

  Bereiche: 10K - 1M 20 - 65

  ➥ lineare Interpolation, Multiplikation von Wahrscheinlichkeiten

- **Lösung ?**
  - Verbesserung der Statistiken/Heuristiken
  - Berechnung/Bewertung von noch mehr Ausführungsplänen ?

  Obwohl die Kostenabschätzungen meist falsch sind . . .

Erstellung und Auswahl von Ausführungsplänen

- **Eingabe:**
  - optimierter Anfragegraph (AG)
  - existierende Speicherungsstrukturen 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 Optimizer**

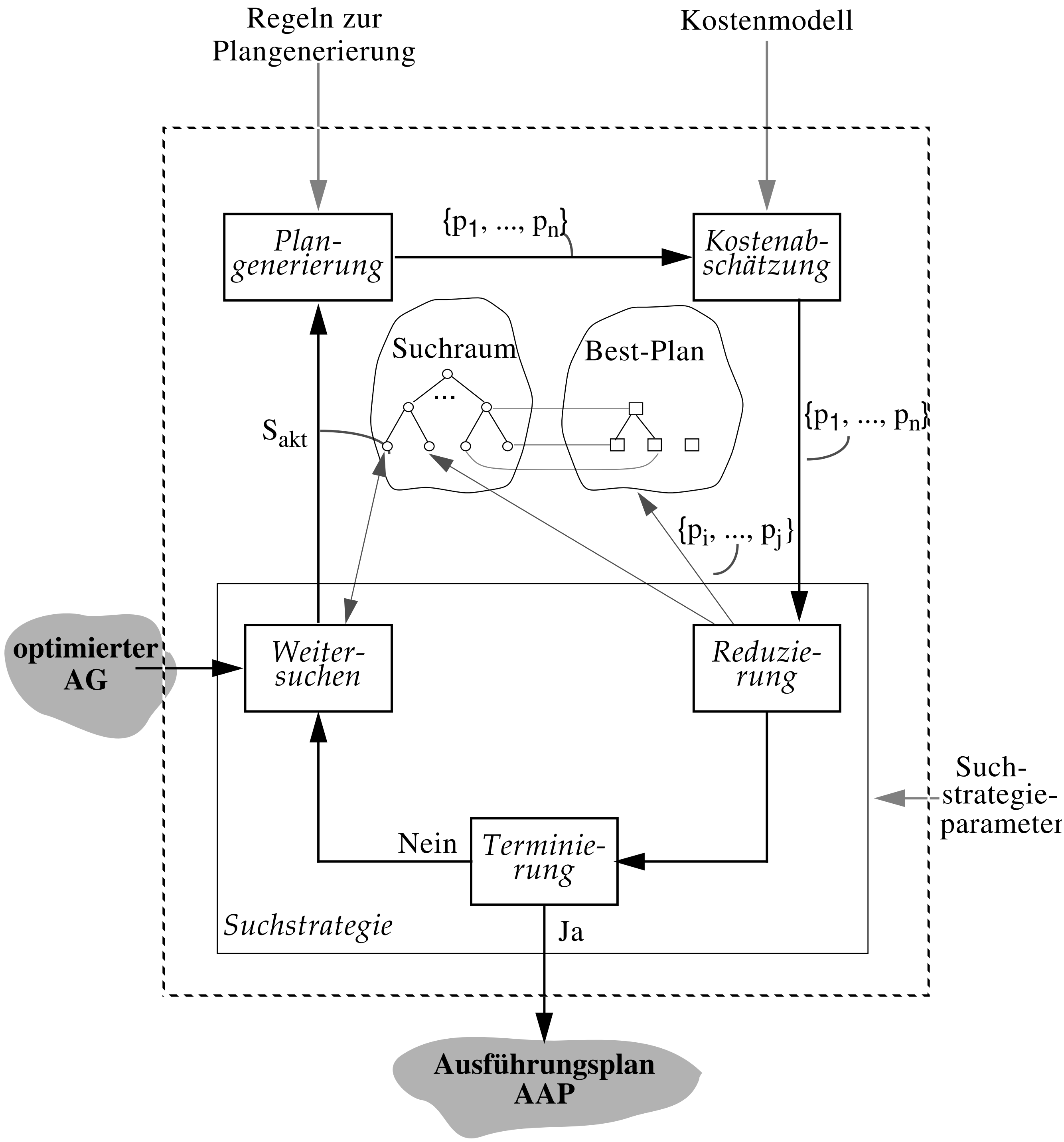
  ➥ kleine Menge der Pläne, die den optimalen Plan enthält

  ➥ 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)*

- **Zusammenspiel der Komponenten**



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

* 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

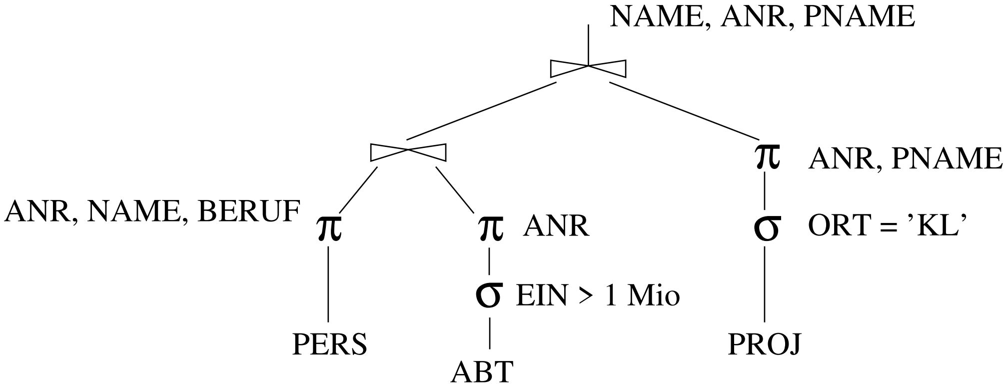
- **Kostenabschätzung**
  - verlangt hinreichend genaues Kostenmodell
  - wird bei allen Suchverfahren inkrementell durchgeführt

- **Problemdarstellung - Beispiel**

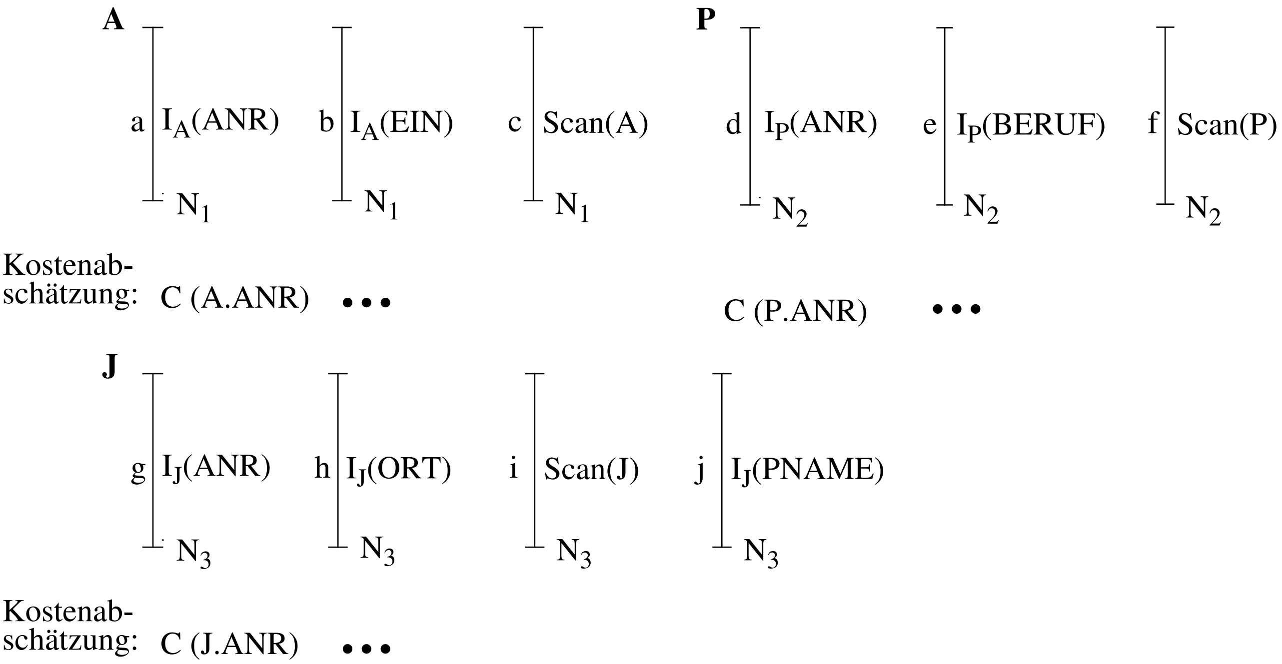
SQL:

```
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;
```

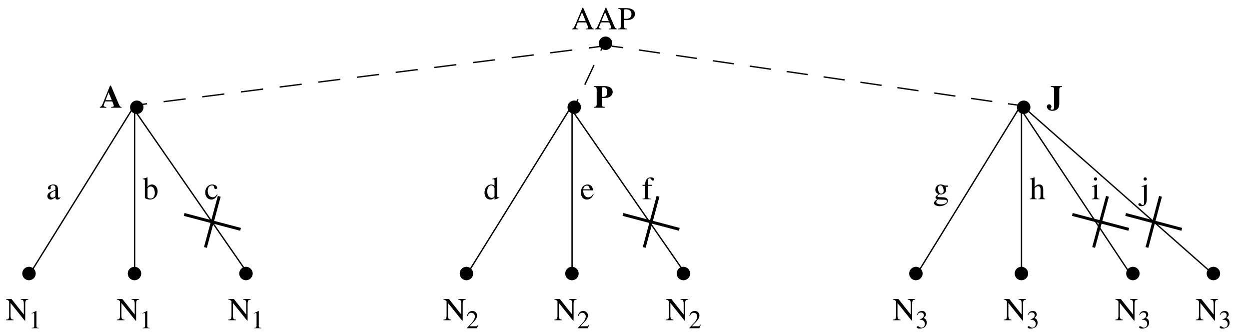
Zugehöriger Anfragegraph



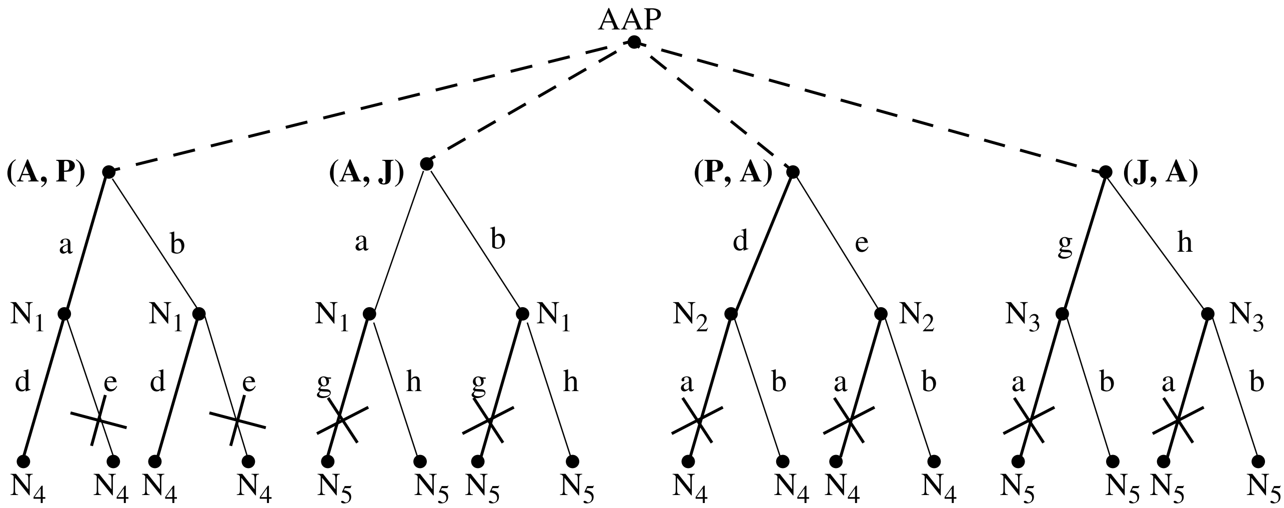
Erstellung und Auswahl von Ausführungsplänen (4)



a) mögliche Zugriffspfade für die einzelnen Relationen



b) Lösungsbaum für einzelne Relationen:
Reduzierung durch Abschneiden von Teilbäumen



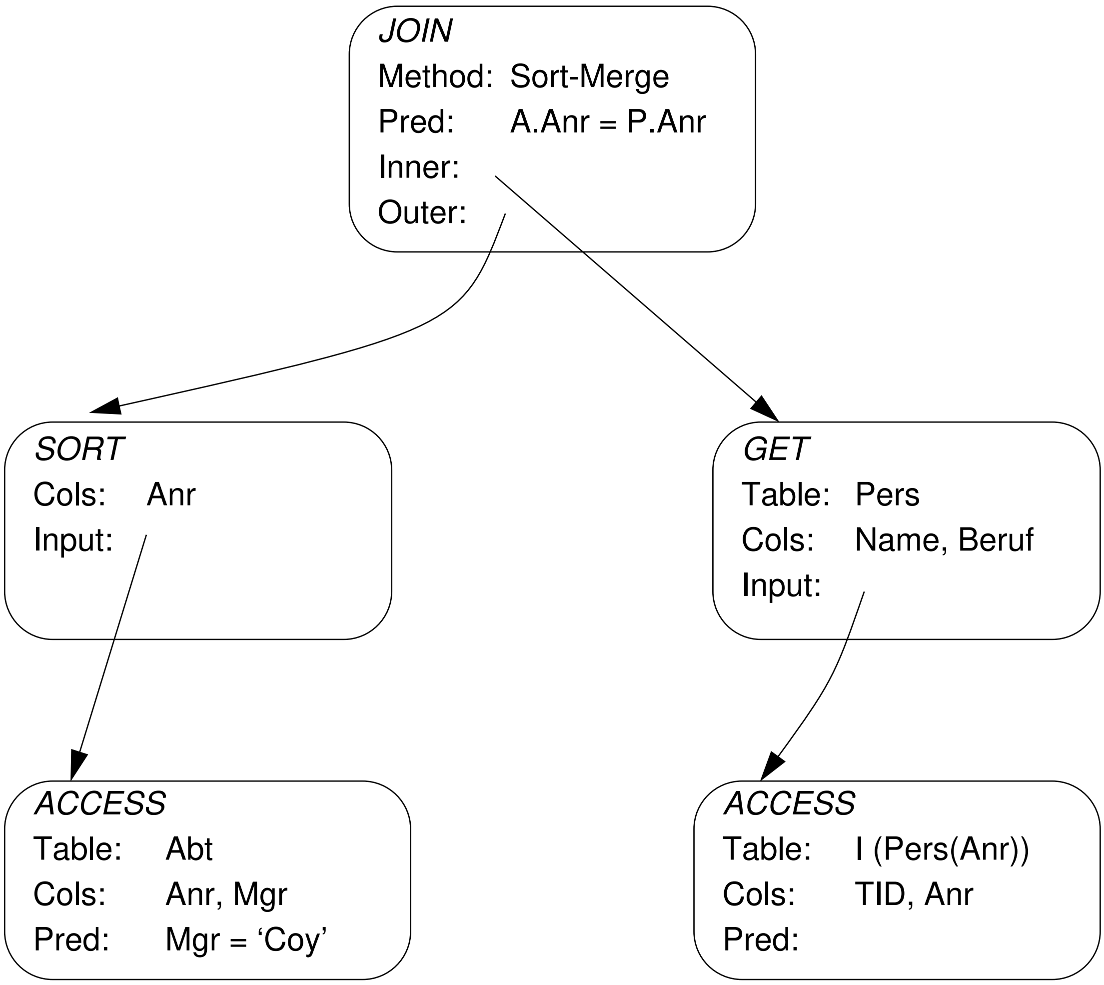
c) Erweiterter Lösungsbaum für den Nested-Loop-Verbund mit der zweiten Relation
Kostenabschätzung pro Pfad: z. B. durch C(C(A.ANR) + C(P.ANR) + Verbundkosten)

Ausführungsplan – Beispiel

- **Anfrage-Beispiel**

```
SQL:    SELECT    Name, Beruf
        FROM      Pers P, Abt A
        WHERE     P.Anr = A.Anr
        AND       A.Mgr = 'Coy'
```

- **Ein möglicher Operatorbaum**



- **Dazugehöriges „Programm“**

```
JOIN (Sort-Merge, A.Anr = P.Anr,
        SORT (ACCESS (Abt, {Anr, Mgr}, {Mgr = 'Coy'}), Anr),
        GET (ACCESS (I (Pers(Anr)), {TID, Anr}, ∅),
                  Pers, {Name, Beruf} , ∅)).
```

Berechnung der Zugriffskosten

- Optimizer erstellt Kostenvoranschlag für jeden Ausführungsplan (möglicher Lösungsweg)

- **Gewichtete Kostenformel:**

  *C = #physischer Seitenzugriffe + W * (#Aufrufe des Zugriffssystems)*

  - gewichtetes Maß für E/A- und CPU-Auslastung
  - W ist das Verhältnis des Aufwandes für einen ZS-Aufruf zu einem Seitenzugriff

- **Ziel der Gewichtung: Minimierung der Kosten in Abhängigkeit des Systemzustandes**

  - System "I/O-bound": ➥ sehr kleiner W-Wert

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

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

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

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

$$\text{Bsp. } W_{CPU} = \frac{1000}{2500} = 0.4$$

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 Relationen:**

  N_R Anzahl der Tupeln der Relation R (Card(R))

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

  C_R Clusterfaktor (Anzahl Tupel pro Seite)

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

  j_I Anzahl der Attributwerte / Schlüsselwerte im Index (=Card (π_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 Relationen und Indexstrukturen
  - periodische Neubestimmung der Statistiken durch eigenes Kommando/ Dienstprogramm (DB2: RUNSTATS)

Kostenmodell – Berechnungsgrundlagen

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

- **Selektivitätsfaktor SF bei:**

$A_i = a_i$ $$SF = \begin{cases} 1/j_i & \text{wenn Index auf } A_i \\ 1/10 & \text{sonst} \end{cases}$$

$A_i = A_k$ $$SF = \begin{cases} 1 / Max(j_i, j_k) & \text{wenn Index auf } A_i, A_k \\ 1 / j_i & \text{wenn Index auf } A_i \\ 1/10 & \text{sonst} \end{cases}$$

$A_i \geq a_i$ $$SF = \begin{cases} (\text{high-key} - a_i) / (\text{high-key} - \text{low-key}) & \text{bei linearer Interpolation} \\ 1/3 & \text{sonst} \end{cases}$$

$A_i \geq a_i \wedge A_i \leq a_k$ $$SF = \begin{cases} (a_k - a_i) / (\text{high-key} - \text{low-key}) & \text{Index auf } A_i \\ 1/4 & \text{sonst} \end{cases}$$

A_i IN (Liste von Werten) $$SF = \begin{cases} r / j_i & \text{bei r Werten auf Index} \\ 1/2 & \text{sonst} \end{cases}$$

- **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)-Joins (verlustfrei)
    Card ($R \bowtie S$) = 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 mengen-orientierter 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**
  - Relationen-Scan im Segment von PERS
  - I_{PERS}(BERUF)
  - I_{PERS}(GEHALT)
  - Link von FAEHIGKEIT nach PERS

- **Statistische Kennwerte**

  Der Optimizer findet folgende Parameter im DB-Katalog:
  - N = # der Tupel in Relation PERS
  - C = durchschn. # von PERS-Tupeln 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)

OPEN SCAN auf I_{PERS}(BERUF)

bei BERUF = ‘PROGRAMMIERER’

FETCH NEXT WHERE GEHALT ≥ 100.000;

CLOSE SCAN wenn BERUF ≠ ‘PROGRAMMIERER’

- **Kosten:**
  - Clusterbildung auf I_{PERS}(BERUF)

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

  - keine Clusterbildung

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

Methode 2: Scan über I_{PERS}(GEHALT)

OPEN SCAN auf I_{PERS}(GEHALT)

bei GEHALT = 100.000

FETCH NEXT WHERE BERUF = ‘PROGRAMMIERER’;

CLOSE SCAN wenn EOT

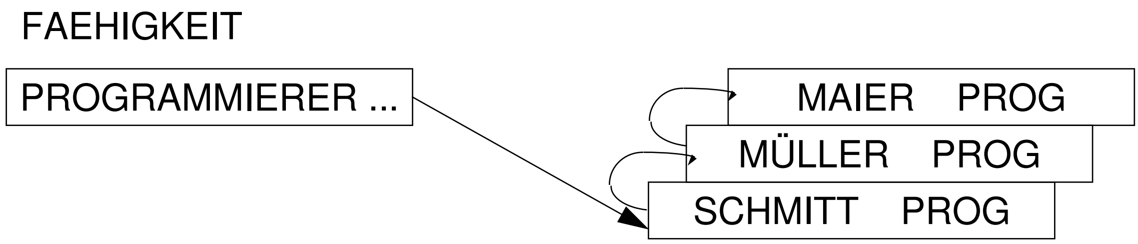
- **Kosten:**
  - Clusterbildung auf I_{PERS}(GEHALT)

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

  - 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 Relation



- **Annahme:**

  Schneller Zugriff auf Relation FAEHIGKEIT als Einstieg in LINK möglich,

  z. B. über $I_{FAEHIGKEIT}$(BERUF)

  ↓

  FETCH Vater-Tupel mit BERUF = ‘PROGRAMMIERER’

  ↓

  OPEN Link-SCAN

  FETCH NEXT ON LINK WHERE GEHALT ≥ 100.000

  CLOSE SCAN wenn Ende des Link

- **Kosten:**

  - Clusterbildung auf Link

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

  - keine Clusterbildung

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