Implementierung von Datenbanksystemen

Kapitel 9: Mengenorientierte Schnittstelle

9. Mengenorientierte Schnittstelle

- Übersetzungsverfahren für Datenbanksprachen
  - Vorgehensweise
  - Übersetzung vs. Interpretation
- Formen der Wirtsspracheneinbettung
  - Direkte Einbettung
  - Aufruftechnik
- Anfrageoptimierung
  - Anfragedarstellung
  - Standardisierung und Vereinfachung
  - Restrukturierung und Transformation
  - Kostenmodelle
  - Erstellung und Auswahl von Ausführungsplänen
- Code-Erzeugung
  - Aufbau von Zugriffsmoduln
  - Bindezeitpunkte für Anweisungstypen
- Behandlung von Ad-hoc-Anfragen

Logische Datenstrukturen

- Charakterisierung der Abbildung:

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

- Abbildungsfunktionen:

Sichten	↔ Basisrelationen
Relationale Ausdrücke	↔ Logische Zugriffspfade
Satzmengen	↔ Einzelne Sätze, Positionsanzeiger

- Operationen an der unteren Schnittstelle:

```
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

Beispiele prozeduraler DB-Anfragen

- Typische Beispiele (Netzwerkmodell)
  - FIND ANY PERS bezieht sich auf die (vorausgesetzte) Klausel LOCATION MODE IS CALC des Satztyps PERS
  - FIND NEXT PERS WITHIN BESCHÄFTIGT SET bezieht sich auf eine relative Position (Currency) in einer Setstruktur
  - FIND OWNER WITHIN BESCHÄFTIGT SET stellt den OWNER-Satz der 'current' Set-Ausprägung zur Verfügung.
  - Lediglich bei der allg. Suchanfrage fallen gewisse Optimierungsaufgaben an; sie ist jedoch auf einen Satztyp beschränkt

Prozedurale vs. deskriptive DB-Sprachen

- Prozedurale Sprachen
  - Erlauben leichte Abbildung der DML-Befehle auf interne Satzoperationen (~1:1)
  - Verantwortung für die Zugriffspfadauswahl liegt beim Programmierer; er bestimmt die Art und Reihenfolge der Zugriffe durch Navigation
  - Bei der Übersetzung sind lediglich Namensauflösung und Formatkonversionen erforderlich
- Deskriptive Anfragen erfordern zusätzlich
  - Überprüfung auf syntaktische Korrektheit (komplexere Syntax)
  - Überprüfung von Zugriffsberechtigung und Integritätsbedingungen
  - Anfrageoptimierung zur Erzeugung einer effizient ausführbaren Folge interner DBMS-Operationen
- Zentrales Problem
  - Umsetzung deskriptiver Anfragen in eine zeitoptimale Folge interner DBMS-Operationen
  - Anfrageübersetzer/-optimierer des DBMS ist im wesentlichen für eine effiziente Abarbeitung verantwortlich, nicht der Programmierer

Übersetzung deskriptiver DB-Sprachen

- Hohe Komplexität der Übersetzung, da die Auswahlmächtigkeit
  - an der Prädikatenlogik erster Stufe orientiert ist; durch zusätzliche Prädikate wie EXISTS, MATCHES, NULL, LIKE u. a. wird diese sogar deutlich übertroffen
  - nicht auf einen Satztyp beschränkt ist
  - unabhängige oder korrelierte Teilanfragen zur Bestimmung von Suchargumenten in beliebiger Schachtelungstiefe zulässt
  - zusätzlich den Einsatz von Built-in- und Sortier-Funktionen auf Partitionen der Satzmenge gestattet
- Zusätzliche Anforderungen
  - auch die Manipulationsoperationen sind mengenorientiert
  - referentielle Integrität ist aktiv mit Hilfe referentieller Aktionen zu wahren
  - Operationen können sich auf Sichten von Relationen beziehen
  - vielfältige Optionen der Datenkontrolle sind zu berücksichtigen
- Anfrageformulierung
  - Formulierung von 'nicht angemessenen' Anfragen (ohne Zugriffspfadunterstützung) erfordert in navigierenden Anfragesprachen einen erheblichen Programmieraufwand
  - in deskriptiven Anfragesprachen dagegen sind sie genauso leicht zu formulieren wie 'günstige' Anfragen
  - Die Ausführung ist in beiden Fällen gleich langsam

Beispiele deskriptiver SQL-Anfragen

- B1: Einfache Anfrage

```
SELECT   PNR, PNAME, GEHALT/12
FROM     PERS
WHERE    BERUF = W
AND      PROV > GEHALT
```

- Ersetzung durch

```
LET      C1 BE SELECT PNR, PNAME,
         GEHALT/12
INTO     :X, :Y, :Z
FROM     PERS
WHERE    BERUF =: W
AND      PROV > GEHALT
```

- mit Operatoren

```
OPEN     C1
FETCH    C1
INTO     :X, :Y, :Z
CLOSE    C1
```

- B2: Komplexere Anfrage mit Verbundoperation sowie unabhängige (T2) und korrelierte (T3) Teilanfragen

T1
```
SELECT   P.PNR, P.NAME, A.ANAME
FROM     PERS P, ABT A
WHERE    P.ANR, A.ANR
```

T2
```
AND      P.GEHALT < (SELECT MAX(PROV)
                     FROM PERS)
```

T3
```
AND      P.GEHALT > (SELECT AVG(PROV)
                     FROM PERS
                     WHERE ANR = P.ANR)
```


Komplexe Anfrage

```
SELECT   a6.custkey, a6.custname, a4.stddeviation, a5.stddeviation,
         a1.turnover1992, a2.turnover1993, a3.turnover1994
FROM
     (SELECT a1.custkey, SUM(a1.endprice)
      FROM   tpch4.lineitem_orders a1, tpch4.lookup_orderday a2
     WHERE   a2.orderdate = a1.orderdate
     AND     a2.orderyearkey = 1992
     GROUP BY a1.custkey
     ) AS a1 (custkey, turnover1992),

     (SELECT a1.custkey, SUM(a1.endprice)
     FROM    tpch4.lineitem_orders a1, tpch4.lookup_orderday a2
     WHERE   a2.orderdate = a1.orderdate
     AND     a2.orderyearkey = 1993
     GROUP BY a1.custkey
     ) AS a2 (custkey, turnover1993),

     (SELECT a1.custkey, SUM(a1.endprice)
      FROM   tpch4.lineitem_orders a1, tpch4.lookup_orderday a2
     WHERE   a2.orderdate = a1.orderdate
     AND     a2.orderyearkey = 1994
     GROUP BY a1.custkey
     ) AS a3 (custkey, turnover1994),

     (SELECT a1.custkey, STDDEV(a1.endprice)
     FROM    tpch4.lineitem_orders a1, tpch4.lookup_orderday a2,
          (SELECT a1.custkey, a1.turnover1992, a2.turnover1993,
                  a3.turnover1994
          FROM
          ...
```

Erzeugt durch ein OLAP-Tool

```
...            (SELECT a1.custkey, SUM(a1.endprice)
                FROM  tpch4.lineitem_orders a1, tpch4.lookup_orderday a2
                WHERE   a2.orderdate = a1.orderdate
                AND     a2.orderyearkey = 1992
                GROUP BY a1.custkey
               ) AS a1 (custkey, turnover1992),

              (SELECT a1.custkey,
                      SUM(a1.endprice)
               FROM tpch4.lineitem_orders a1,
                    tpch4.lookup_orderday a2
               WHERE a2.orderdate = a1.orderdate
                     AND a2.orderyearkey = 1993
               GROUP BY a1.custkey
              ) AS a2 (custkey, turnover1993),

              (SELECT a1.custkey,
                      SUM(a1.endprice)
               FROM tpch4.lineitem_orders a1,
                    tpch4.lookup_orderday a2
               WHERE a2.orderdate = a1.orderdate
                     AND a2.orderyearkey = 1994
               GROUP BY a1.custkey
              ) AS a3 (custkey, turnover1994)
            WHERE a1.custkey = a2.custkey
                AND a1.custkey = a3.custkey
                AND a1.turnover1992 >= 500000
                AND a2.turnover1993 >= 500000
                AND a3.turnover1994 >= 500000
          ) AS a3 (custkey, turnover1992, turnover1993, turnover1994)
        WHERE a2.orderdate = a1.orderdate
            AND a2.orderyearkey IN (1992, 1993, 1994)
            AND a1.custkey = a3.custkey
      GROUP BY a1.custkey
     ) AS a4 (custkey, stddeviation),
```

Komplexe Anfrage

```
(SELECT a1.custkey,
 STDDEV(a1.endprice) /
  (CASE AVG(a1.endprice) WHEN 0 THEN NULL
                         ELSE AVG(a1.endprice) END)
FROM tpch4.lineitem_orders a1,
   tpch4.lookup_orderday a2,

   (SELECT a1.custkey,
        a1.turnover1992,
        a2.turnover1993,
        a3.turnover1994
    FROM
       (SELECT a1.custkey,
            SUM(a1.endprice)
        FROM tpch4.lineitem_orders a1,
           tpch4.lookup_orderday a2
        WHERE a2.orderdate = a1.orderdate
           AND a2.orderyearkey = 1992
        GROUP BY a1.custkey
       ) AS a1 (custkey, turnover1992),

       (SELECT a1.custkey,
            SUM(a1.endprice)
        FROM tpch4.lineitem_orders a1,
           tpch4.lookup_orderday a2
        WHERE a2.orderdate = a1.orderdate
           AND a2.orderyearkey = 1993
        GROUP BY a1.custkey
       ) AS a2 (custkey, turnover1993),
       ...
```

```
       ...
       (SELECT a1.custkey,
            SUM(a1.endprice)
        FROM tpch4.lineitem_orders a1,
           tpch4.lookup_orderday a2
        WHERE a2.orderdate = a1.orderdate
           AND a2.orderyearkey = 1994
        GROUP BY a1.custkey
       ) AS a3 (custkey, turnover1994)
    WHERE a1.custkey = a2.custkey
       AND a1.custkey = a3.custkey
       AND a1.turnover1992 >= 500000
       AND a2.turnover1993 >= 500000
       AND a3.turnover1994 >= 500000
    ) AS a3 (custkey, turnover1992, turnover1993, turnover1994)
  WHERE a2.orderdate = a1.orderdate
     AND a2.orderyearkey IN (1992, 1993, 1994)
     AND a1.custkey = a3.custkey
  GROUP BY a1.custkey
 ) AS a5 (custkey, stddeviation),
 tpch4.lookup_customer a6
WHERE a4.custkey = a1.custkey
  AND a4.custkey = a2.custkey
  AND a4.custkey = a3.custkey
  AND a4.custkey = a5.custkey
  AND a4.custkey = a6.custkey
  AND a5.stddeviation <= 0.66794004454646;
```


Übersetzung von DB-Anweisungen

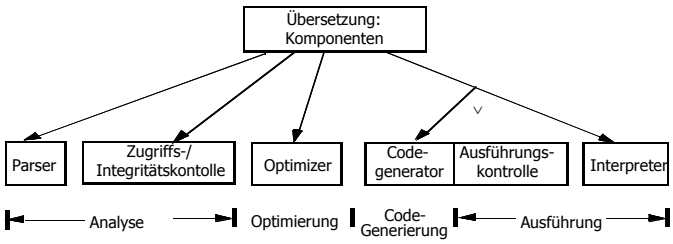
1. Lexikalische und syntaktische Analyse
   - Erstellung eines Anfragegraphs (AG) als Bezugsstruktur für die nachfolgenden Übersetzungsschritte
   - Überprüfung auf korrekte Syntax (Parsing)
2. Semantische Analyse
   - Feststellung der Existenz und Gültigkeit der referenzierten Relationen, Sichten und Attribute
   - Einsetzen der Sichtdefinitionen in den AG
   - Ersetzen der externen durch interne Namen (Namensauflösung)
   - Konversion vom externen Format in interne Darstellung
3. Zugriffs- und Integritätskontrolle
   - sollen aus Leistungsgründen, soweit möglich, schon zur Übersetzungs-zeit erfolgen
   - Zugriffskontrolle erfordert bei Wertabhängigkeit Generierung von Laufzeitaktionen
   - Durchführung einfacher Integritätskontrollen (Kontrolle von Formaten und Konversion von Datentypen)
   - Generierung von Laufzeitaktionen für komplexere Kontrollen

Übersetzung von DB-Anweisungen

4. Standardisierung und Vereinfachung
   - Dienen der effektiveren Übersetzung und frühzeitigen Fehlererkennung
   - Überführung des AG in eine Normalform
   - Elimination von Redundanzen
5. Restrukturierung und Transformation
   - Restrukturierung zielt auf globale Verbesserung des AG ab; bei der Transformation werden ausführbare Operationen eingesetzt
   - Anwendung von heuristischen Regeln (algebraische Optimierung) zur Restrukturierung des AG
   - Transformation führt Ersetzung und ggf. Zusammenfassen der logischen Operatoren durch Planoperatoren durch (nicht-algebraische Optimierung): Meist sind mehrere Planoperatoren als Implementierung eines logischen Operators verfügbar
   - Bestimmung alternativer Ausführungspläne (nicht-algebraische Optimierung): Meist sind viele Ausführungsreihenfolgen oder Zugriffspfade auswählbar
   - Bewertung der Kosten und Auswahl des günstigsten Ausführungsplanes
   - Schritte 4 + 5 werden als Anfrageoptimierung zusammengefasst

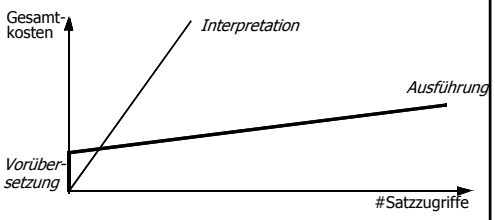
Übersetzung von DB-Anweisungen

5. Code-Generierung
   - Generierung eines zugeschnittenen Programms für die vorgegebene (SQL-) Anfrage
   - Erzeugung eines ausführbaren Zugriffsmoduls
   - Verwaltung der Zugriffsmodule in einer DBMS-Bibliothek



Übersetzung vs. Interpretation

- Anfrageanalyse und -optimierung können zur Übersetzungszeit des AP oder zur Laufzeit (Interpretation) erfolgen
- Übersetzung:
  - Spracherweiterungsansatz erfordert erweiterten Compiler C' , Vorübersetzeransatz einen Pre-Compiler (PC)
  - C' bzw. PC führen Abbildungsfunktionen aus
  - Aufwändige Optimierung und effiziente Ausführung möglich
  - Änderungen des DB-Zustandes nach der Übersetzung werden nicht berücksichtigt (neue Zugriffspfade, geänderte Statistiken etc.)
  - Invalidierung des Zugriffsmoduls und Neuübersetzung
- Interpretation:
  - Allgemeiner Interpreter wertet Anfrage direkt zur Laufzeit aus
  - Aktueller DB-Zustand lässt sich für Auswertungsstrategie berücksichtigen
  - Sehr hohe Ausführungskosten bei Programmschleifen sowie durch häufige Katalogzugriffe
  - Interessant vor allem für Ad-hoc-Anfragen bzw. dynamische SQL-Anweisungen (PREPARE / EXECUTE)

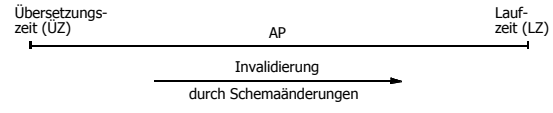


Übersetzung vs. Interpretation

- Was heißt „Binden"?
  - AP:

```
Select    Pnr, Name, Gehalt
From      Pers
Where     Beruf = 'Programmierer'
```

- DB-Katalog:
  - SYSREL: Tabellenbeschreibungen: Pers, . . .
  - SYSATTR: Attributbeschreibungen: Pnr, Name, Gehalt, . . .
  - SYSINDEX: I_{Pers}(Beruf), . . .
  - SYSAUTH: Nutzungsrechte
  - SYSINT/RULES: Integritätsbedingungen, Zusicherungen, . . .
- Zeitpunkt des Bindens



Übersetzungskosten:
unerheblich für Antwortzeit (AZ)

Zugriffe (zur LZ):
effizient
datenabhängig!

{ Ausgleich gesucht! }

Interpretation:
erheblich für AZ

Zugriffe (zur LZ):
teuer
datenunabhängig!

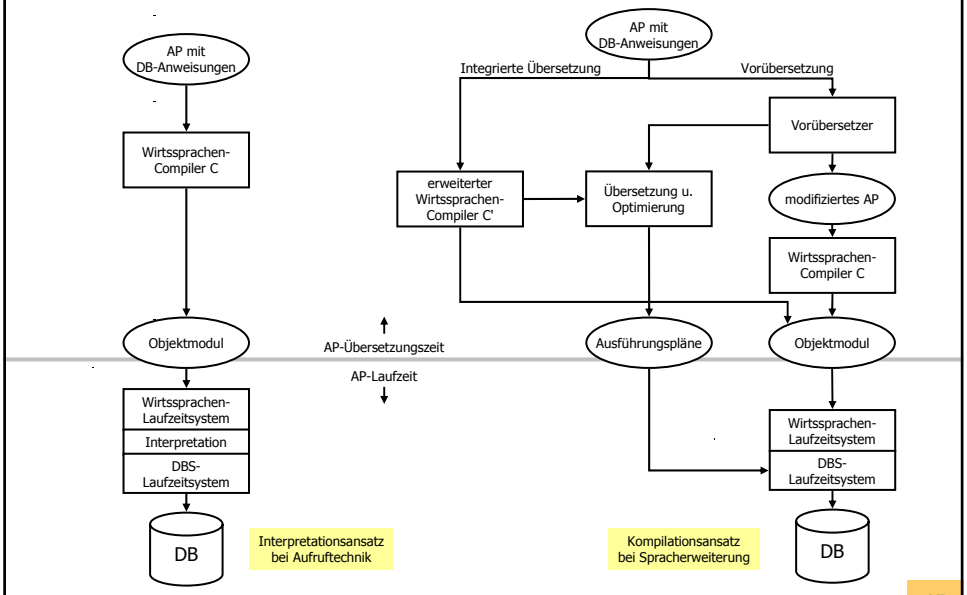
9. Mengenorientierte Schnittstelle

- Übersetzungsverfahren für Datenbanksprachen
  - Vorgehensweise
  - Übersetzung vs. Interpretation
- Formen der Wirtsspracheneinbettung
  - Direkte Einbettung
  - Aufruftechnik
- Anfrageoptimierung
  - Anfragedarstellung
  - Standardisierung und Vereinfachung
  - Restrukturierung und Transformation
  - Kostenmodelle
  - Erstellung und Auswahl von Ausführungsplänen
- Code-Erzeugung
  - Aufbau von Zugriffsmoduln
  - Bindezeitpunkte für Anweisungstypen
- Behandlung von Ad-hoc-Anfragen

Formen der Wirtssprachen-Einbettung

- Selbständige DB-Sprachen:
  - Deskriptive Anweisungen und mengenorientierte Zugriffe sind die natürliche Form des Einsatzes
  - Weiterverarbeitung der Ergebnisse verlangt Einbettung der DB-Anweisungen in eine Wirtssprache
  - Bei deskriptiven Sprachen mit mengenorientiertem Datenzugriff ergibt sich eine Fehlanpassung bei der Kopplung mit satzorientierter Verarbeitung ("impedance mismatch")
- Prinzipielle Möglichkeiten
  - Direkte Einbettung (Spracherweiterung)
    Dabei wird syntaktisch keine Unterscheidung zwischen Programm- und DB-Anweisungen gemacht. Eine DB-Anweisung wird als Zeichenkette A ins AP integriert.
  - Aufruftechnik
    Eine DB-Anweisung wird durch einen expliziten Funktionsaufruf an das Laufzeitsystem des DBS übergeben (z. B. CALL DBS ('A'))

Übersetzungstechniken für AP mit DB-Anweisungen



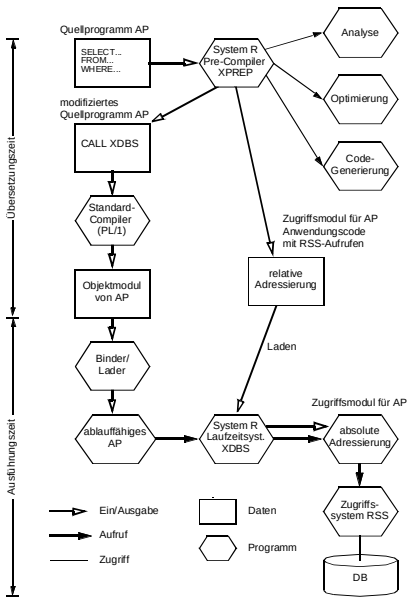
Direkte Einbettung / Spracherweiterung

- Übersetzung von AP mit
  - modifiziertem Wirtssprachen-Compiler C‘: AP → MC
  - Pre-Compiler
    PC: AP → AP‘
    +
    C: AP‘ → MC
- Bindung von A
  - zur ÜZ (Optimierung)
  - zur LZ (interner CALL-Aufruf)
- Mögliche Vorgehensweisen bei C’ bzw. PC:
  - A wird als Ergebnis von Übersetzung und Optimierung in eine interne Form überführt, die eine direkte Ausführung zur Laufzeit gestattet, d. h., die Bindung von A an DB-interne Namen und Zugriffspfade erfolgt zum Übersetzungszeitpunkt.
  - A wird als aktueller Parameter eines (internen) CALL-Aufrufs abgelegt, seine weitere Behandlung erfolgt erst zur Laufzeit. A kann entweder übersetzt oder interpretiert werden.

Aufruftechnik

- Übersetzung von AP mit
  - C: AP → MC
  - keine Modifikation des Wirtssprachen-Compilers notwendig
- Bindung von A:
  - zur LZ
- Interpretation/Übersetzung + Bindung zur Laufzeit
- Höherer konzeptioneller Abstand als bei direkter Einbettung

Übersetzung und Ausführung von DB-Anweisungen Beispiel: System R



Formen der Wirtssprachen-Einbettung

- Art der DB-seitigen Datenstrukturen im AP
  - Variante 1:
    Die Datenstrukturen, auf die das AP zugreift, sind als Subschema (oder Sicht) explizit im AP deklariert, d. h., das AP besitzt einen statisch zugeordneten Bereich für die Datenstrukturen der DB (UWA = User Working Area)
  - Variante 2:
    Im AP ist kein Speicherplatz für DB-seitige Datenstrukturen reserviert. DB-Werte werden explizit an normale Programmvariablen übergeben

Klassifikation der DB-Schnittstelle

		Operatoren in Wirtssprache integriert: ja	Operatoren in Wirtssprache integriert: nein
Datenstrukturen in AP deklariert	Ja	Direkte Einbettung Übergabe in statisch zugeordneten Datenstrukturen B-Typ = C/C	CALL-Technik Übergabe in statisch zugeordneten Datenstrukturen B-Typ = C/L
	Nein	Direkte Einbettung Übergabe an Programm-Variable B-Typ = L/C	CALL-Technik Übergabe an Programmvariable/ Pufferbereiche B-Typ = L/L

Bindungstyp (B-Type): Bindezeitpunkte für Datenstrukturen/Operatoren
C = Compilerzeitbindung L = Laufzeitbindung

Beispiele der Wirtssprachen-Einbettung

- Datenstrukturen und Operatoren in Wirtssprache integriert
  - Subschema in UWA (CODASYL)
  - DDL/DML

```
FIND X-RECORD USING Y, Z
```

- Datenstrukturen in Wirtssprache integriert
  - Subschema in UWA

```
CALL "DML"  USING FUNKTIONSNAME, FUNKTIONSWAHL,
                  ZUSATZWAHL, BENUTZERINFO, ...
```

- Operatoren in Wirtssprache integriert
  - Bsp.: SQL

```
DECLARE C1 CURSOR FOR
SELECT   PNR, PNAME
FROM     PERS
WHERE    BERUF= :Z AND ANR = 'K55'
. . .
FETCH C1 INTO :X, :Y
```

- Keine Integration

```
CALL ADABAS
(CONTROL_BLOCK, FOBU, REBU, SEBU, VABU, ISNBU)
```


Einbettung einer mengenorientierten Schnittstelle

- SQL-Anweisung:

```
SELECT   PNR, PNAME, GEHALT/12
FROM     PERS
WHERE    BERUF='Operateur'
AND      PROV>GEHALT
```

- DBS stellt Anweisungen bereit zur:
  - … Spezifikation der gesuchten Tupelmenge (Qualifikations-operator)
  - … sukzessiven Bereitstellung der qualifizierten Tupel (Abholoperator)

```
DECLARE C1 CURSOR FOR
SELECT   PNR, PNAME, GEHALT/12
FROM     PERS
WHERE    BERUF = :W
AND      PROV > GEHALT

OPEN C1;                    →   Bindung von W, z.B. 'Operateur',
                                Aktivierung des Cursors
FETCH C1 INTO :X, :Y, :Z;   →   Abholen eines Tupels
CLOSE C1;                   →   Deaktivierung des Cursors
```

Einbettung einer mengenorientierten Schnittstelle

- Mögliche Ersetzung durch Pre-Compiler:

DECLARE C1 ...	→	Kommentar
OPEN C1	→	DCL T(3) POINTER; T(1) = ADDR(W) CALL XDBS (ZM1, 2, OPEN, ADDR(T))
FETCH C1 INTO ...	→	T(1) = ADDR(X); T(2) = ADDR(Y); T(3) = ADDR(Z); CALL XDBS (ZM1, 2, FETCH, ADDR(T))

9. Mengenorientierte Schnittstelle

- Übersetzungsverfahren für Datenbanksprachen
  - Vorgehensweise
  - Übersetzung vs. Interpretation
- Formen der Wirtsspracheneinbettung
  - Direkte Einbettung
  - Aufruftechnik
- Anfrageoptimierung
  - Anfragedarstellung
  - Standardisierung und Vereinfachung
  - Restrukturierung und Transformation
  - Kostenmodelle
  - Erstellung und Auswahl von Ausführungsplänen
- Code-Erzeugung
  - Aufbau von Zugriffsmoduln
  - Bindezeitpunkte für Anweisungstypen
- Behandlung von Ad-hoc-Anfragen

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:

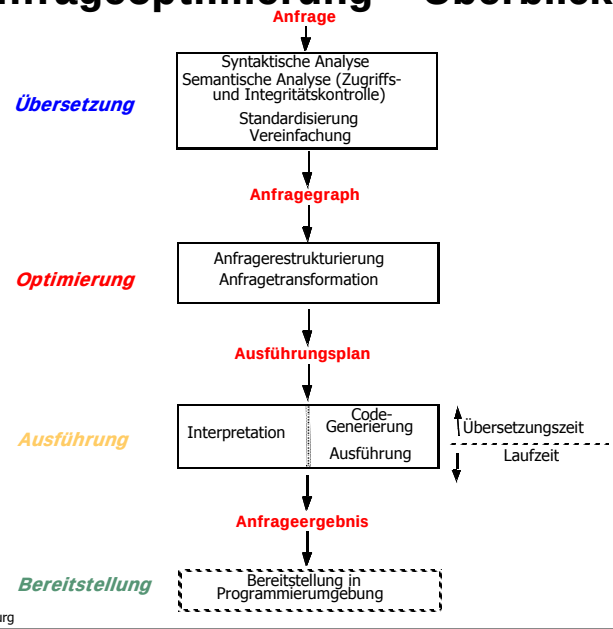
- Minimierung der Ressourcennutzung für gegebenen Output
  - Antwortzeitminimierung für eine gegebene Anfragesprache, einem Mix von Anfragen verschiedenen Typs und einer gegebenen Systemumgebung!
- oder Maximierung des Outputs bei gegebenen Ressourcen
  - Durchsatzmaximierung ?

Anfrageoptimierung

- Welche Kosten sind zu berücksichtigen?
  - E/A-Kosten (# der physischen Referenzen)
  - Berechnungskosten (CPU-Kosten, Pfadlängen)
  - Speicherungskosten (temporäre Speicherbelegung im DB-Puffer und auf Externspeichern)
  - Kommunikationskosten in verteilten DBS!
    (# der Nachrichten, Menge der zu übertragenden Daten)
- 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
    $(A_{11}$ OR ... OR $A_{1n})$
    AND ...
    AND $(A_{m1}$ OR ... OR $A_{mn})$
  - 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)

Distributivregeln

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 Teilausdrücke
  - (A1 = a11 OR A1 = a12) AND (A1 = a12 OR A1 = a11)
- Vereinfachung von Ausdrücken, die an "leere Relationen" gebunden sind
- Konstanten-Propagierung
  - A op B AND B = const. $\Rightarrow$ A op const.
- nicht-erfüllbare Ausdrücke
  - A $\geq$ B AND B > C AND C $\geq$ A $\Rightarrow$ A > A $\rightarrow$ false
- Nutzung von Integritätsbedingungen (IB)
  - IB sind für alle Tupel der betreffenden Relation wahr
  - A ist Primärschlüssel: $\pi_A \rightarrow$ keine Duplikateliminierung erforderlich
  - Regel: FAM-STAND = 'verh.' AND STEUERKLASSE $\geq$ 3
    Ausdruck: (FAM-STAND = 'verh.' AND STEUERKLASSE = 1) $\rightarrow$ false

Vereinfachung einer Anfrage

- 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: deskriptive 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

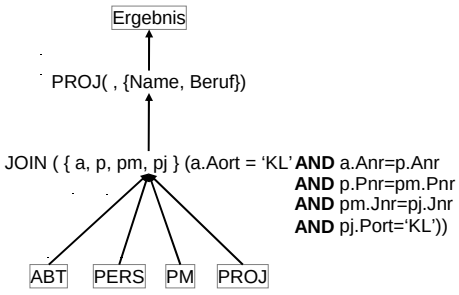
Interndarstellung einer Anfrage

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

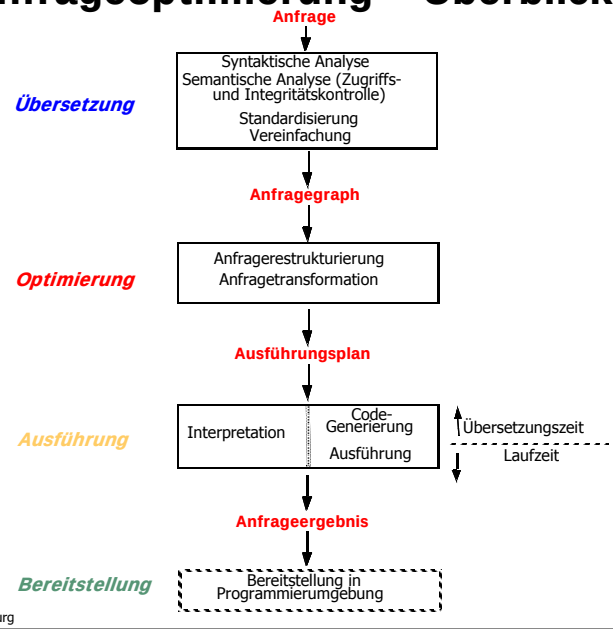
  Finde Name und Beruf von Angestellten, die Projekte in 'KL' durchführen und deren zugehörige Abteilung sich ebenfalls in 'KL' befindet"

```
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



Anfrageoptimierung – Überblick



Anfrageoptimierung

- Aufgabe: Abbildung
  von den logischen Operationen der Interndarstellung
  über zugeordnete ausführbare Operationen
  auf effizient durchführbare Ausführungspläne
- Problematik:
  - ➢ Komplexität dieser Aufgabe
  - Finden des besten unter den vielen möglichen Ausführungsplänen
  - leistungsbestimmende Merkmale:
    - Wahl der physischen Operatoren
    - Wahl der besten Realisierungsalternative für einen physischen Operator
    - Reihenfolge der Operatorausführung
    - Größe der Zwischenergebnisse
- Schritte:
  - Anfragerestrukturierung
  - Anfragetransformation

Anfragerestrukturierung

- Wichtigste Regeln für Restrukturierung und Transformation
  - Selektionen (σ) und Projektionen (π) ohne Duplikateliminierung sollen möglichst frühzeitig ausgeführt werden.
  - Folgen von unären Operatoren (wie σ und π) auf einer Relation sind zu einer Operation mit komplexerem Prädikat zusammenzufassen.
  - Selektionen und Projektionen, die eine Relation betreffen, sollen so zusammengefasst werden, dass jedes Tupel nur einmal verarbeitet werden muss.
  - Bei Folgen von binären Operatoren (wie $\cap$, $\cup$, $-$, $\bowtie$) ist eine Minimierung der Größe der Zwischenergebnisse anzustreben.
  - Gleiche Teile im AG sind nur einmal auszuwerten.
- Zusammenfassung von Operationsfolgen

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

  R2: $\sigma_{pn}(... \sigma_{p2}(\sigma_{p1}(Rel))...)...) \Leftrightarrow \sigma_{p1\ AND\ p2\ ...\ AND\ pn}(Rel)$
- Minimierung der Größe von Zwischenergebnissen
  - selektive Operationen (σ, π) vor konstruktiven Operationen ($\bowtie$)

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 aufeinander folgende 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 (derselben 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, Änderungsoperationen (Einfügen, Löschen, Modifizieren), ACCESS zum Zugriff auf Basisrelationen
    + Erweiterungen: Rekursion, Gruppierung, . . .
  - auf zwei Relationen:
    Verbund, Mengen-Operationen, Kartesisches Produkt.
- Anpassungen im AG zum effektiven Einsatz von Planoperatoren
  - Gruppierung von direkt benachbarten Operatoren zur Auswertung durch einen Planoperator;
    z. B. lassen sich durch einen speziellen Planoperator ersetzen:
    Verbund (oder Kartesisches Produkt) mit Selektionen und/oder Projektionen auf den beteiligten Relationen.
  - Bestimmung der Verknüpfungsreihenfolge bei Mengen- und Verbundoperationen;
    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.
  - Erkennung gemeinsamer Teilbäume, die dann nur jeweils einmal zu berechnen sind. Allerdings steht dieser Einsparung die Zwischenspeichung der Ergebnisrelation gegenüber.

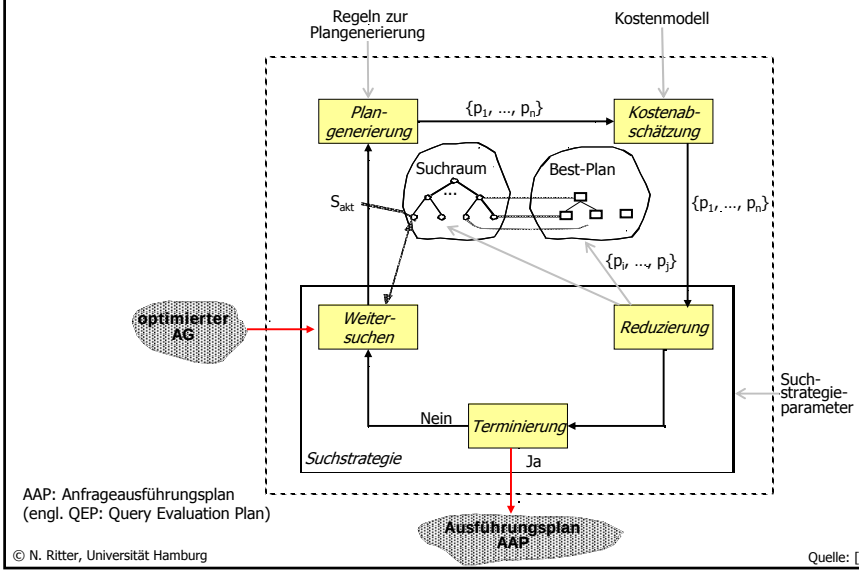
Bewertung von Ausführungsplänen – Grundsätzliche Probleme

- Anfrageoptimierung beruht i.A. auf zwei „fatalen" Annahmen
  - Alle Datenelemente und alle Attributwerte sind gleichverteilt
  - Suchprädikate in Anfragen sind unabhängig
  - ➢ beide Annahmen sind falsch (im allgemeinen Fall)
- Beispiel
  - (GEHALT ≥ '100K') AND (ALTER BETWEEN 20 AND 30)
  - lineare Interpolation, Multiplikation von Wahrscheinlichkeiten
- Lösungen ?
  - 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:
  - Generiere alle „vernünftigen" logischen Ausführungspläne zur Auswertung der Anfrage
  - Vervollständige die Ausführungspläne durch Einzelheiten der physischen Datenrepräsentation (Sortierreihenfolge, Zugriffspfadmerkmale, statistische Information)
  - Wähle den billigsten Ausführungsplan gemäß dem vorgegebenen Kostenmodell aus

Erstellung und Auswahl von Ausführungsplänen



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

Quelle: [Mit95]

Erstellung und Auswahl von Ausführungsplänen

- Wie entstehen alternative Ausführungspläne für einen AG?
  - für jeden Planoperator liegen verschiedene Methoden (Implementierungen) vor
  - Operationsreihenfolgen (z. B. bei Mehrfachverbunden) können variiert werden
  - 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

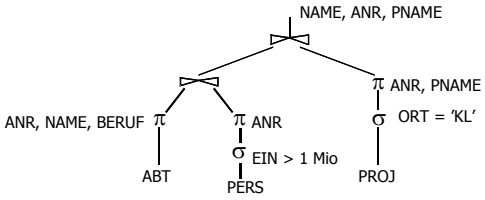
- Plangenerierung soll
  - immer und möglichst schnell den „optimalen" Plan finden
  - mit einer möglichst kleinen Anzahl generierter Pläne auskommen
- Suchstrategien
  - voll-enumerativ
  - beschränkt-enumerativ
  - zufallsgesteuert
  - Reduzierung: Bestimmte Suchpfade zur Erstellung von AAPs werden nicht weiter verfolgt
- Kostenabschätzung
  - verlangt hinreichend genaues Kostenmodell
  - wird bei allen Suchverfahren inkrementell durchgeführt

Erstellung und Auswahl von Ausführungsplänen

- Problemdarstellung - Beispiel

```
SELECT   P.NAME, P.BERUF, J.NAME
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

A

a I_A(ANR) N_1 — b I_A(EIN) N_1 — c Scan(A) N_1

Kostenab-schätzung: C (A.ANR) . . .

P

d I_P(ANR) N_2 — e I_P(BERUF) N_2 — f Scan(P) N_2

Kostenab-schätzung: C (P.ANR) . . .

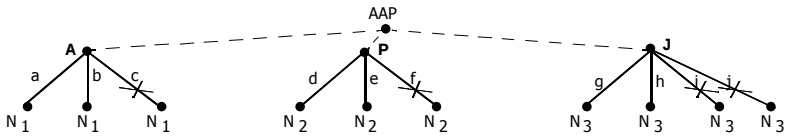
J

g I_J(ANR) N_3 — h I_J(ORT) N_3 — i Scan(J) N_3 — j I_J(PNAME) N_3

Kostenab-schätzung: C (J.ANR) . . .

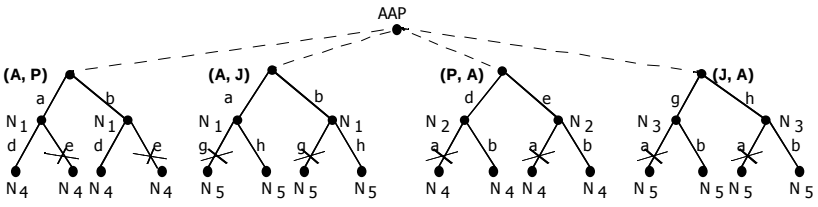
a) mögliche Zugriffspfade für die einzelnen Relationen

Erstellung und Auswahl von Ausführungsplänen



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

Erstellung und Auswahl von Ausführungsplänen



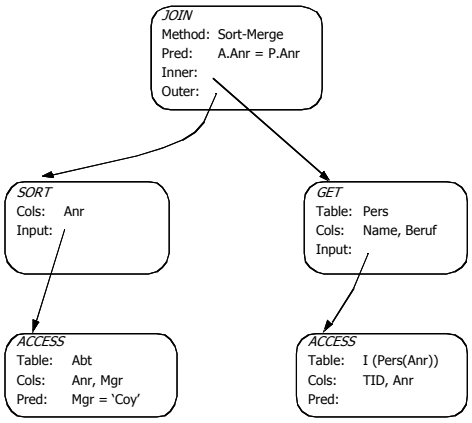
c) Erweiterter Lösungsbaum für den Nested-Loop-Verbund mit der zweiten Relation
Kostenabschätzung pro Pfad: z. B. durch C(C(A.ANR) + C(P.ANR) + Verbundkosten)

Ausführungsplan – Beispiel

- Anfrage-Beispiel:

```
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": kleines W

$$W_{I/O} = \frac{\#Instr.\ pro\ ZS-Aufruf}{\#Instr.pro\ E/A + Zugriffszeit \cdot MIPS\text{-}Rate} \qquad W_{I/O} = \frac{1000\ I.}{2500\ I. + 12\ msec \cdot 10^7\ I./sec} = 0,008$$

  - System "CPU-bound": relativ großes W

$$W_{CPU} = \frac{\#Instr.pro\ ZS-Aufruf}{\#Instr.\ pro\ E/A} \qquad W_{CPU} = \frac{1000}{2500} = 0,4$$

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 Tupel 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 Generierungs-zeitpunkt von Relationen und Indexstrukturen
  - periodische Neubestimmung der Statistiken durch eigenes Kommando/ Dienstprogramm (DB2: RUNSTATS)

Kostenmodell – Berechnungsgrundlagen

- Selektivitätsfaktor SF
  - 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 (σ_p(R)) = SF(p) x Card (R)
- Selektivitätsfaktor SF bei:

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

Kostenmodell – Berechnungsgrundlagen

- Berechnung von Ausdrücken
  - SF (p(A) $\wedge$ p(B)) = SF (p(A)) . SF (p(B))
  - SF (p(A) $\vee$ p(B)) = SF (p(A)) + SF (p(B)) - SF (p(A)) . 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))

Beispiel: Einfache 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 = durchschnittliche # 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'

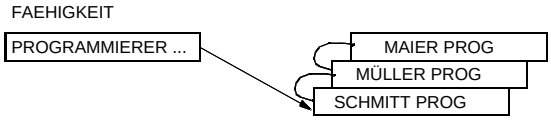
- Kosten:
  - Clusterbildung auf I_{PERS}(BERUF) $K \approx 3 + \frac{N}{C \cdot j_{BERUF}} + w \frac{N}{j_{BERUF} \cdot 10}$
  - keine Clusterbildung $K \approx 3 + \frac{N}{j_{BERUF}} + w \cdot \frac{N}{j_{BERUF} \cdot 10}$

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



FETCH Vater-Tupel mit BERUF = 'PROGRAMMIERER'
OPEN LINK-SCAN

FETCH NEXT ON LINK WHERE GEHALT ≥ 100.000

CLOSE SCAN wenn Ende des LINK

- Annahme:
  - Schneller Zugriff auf Relation 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 $K \approx 3 + \frac{N}{j_{BERUF}} + w \cdot \frac{N}{j_{BERUF} \cdot 10}$

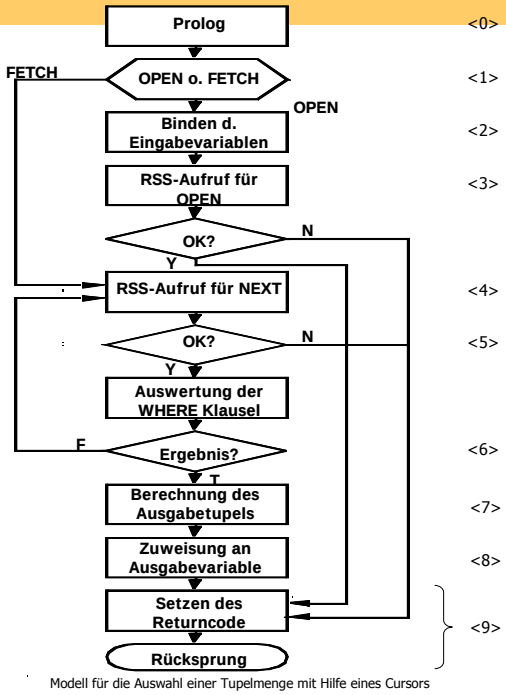
9. Mengenorientierte Schnittstelle

- Übersetzungsverfahren für Datenbanksprachen
  - Vorgehensweise
  - Übersetzung vs. Interpretation
- Formen der Wirtsspracheneinbettung
  - Direkte Einbettung
  - Aufruftechnik
- Anfrageoptimierung
  - Anfragedarstellung
  - Standardisierung und Vereinfachung
  - Restrukturierung und Transformation
  - Kostenmodelle
  - Erstellung und Auswahl von Ausführungsplänen
- Code-Erzeugung
  - Aufbau von Zugriffsmoduln
  - Bindezeitpunkte für Anweisungstypen
- Behandlung von Ad-hoc-Anfragen

Code-Erzeugung

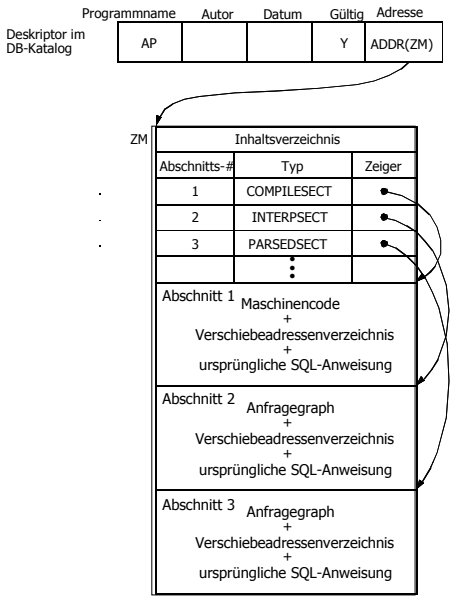
- Optimierter Anfragegraph
  - Ergebnis der Optimierungsphase
  - Eingabe-Datenstruktur für Code-Generator
- Nutzung der Operationen des Zugriffssystems
  - direkte Operationen (z.B. INSERT <satz>)
  - Scan-Operationen (Beispiel SYSTEM R)
    - CALL RSS (OPEN, SCAN_STRUCTURE, RETURN_CODE)
    - CALL RSS (NEXT, SCAN_STRUCTURE, RETURN_CODE)
    - SCAN_STRUCTURE ist komplexe Datenstruktur zur Übergabe von Ein-/Ausgabewerten, Suchargumenten usw.
  - Diese Operationen werden bei der Code-Generierung als Primitive eingesetzt
- Klassifikation der SQL-Anweisungen
  - jede Klasse wird durch Basisprozess beschrieben (z.B. Auswahl einer Tupelmenge mit Hilfe eines Cursors)
  - Skelett eines Basisprozesses heißt Modell
  - Verarbeitungsschritt im Modell heißt Fragment (als Codefolge in einer Bibliothek abgelegt)
  - Klassifikation erfolgt nach Art der Zugriffsaktionen
- Bereitstellung von Modellen und Fragmenten
  - 4 Modelle für einfache Fragen (Frageblöcke)
  - insgesamt 30 Modelle mit jeweils 5-10 Fragmenten (<100 Fragmente)

Flußdiagram für ein Zugriffsmodul



Modell für die Auswahl einer Tupelmenge mit Hilfe eines Cursors

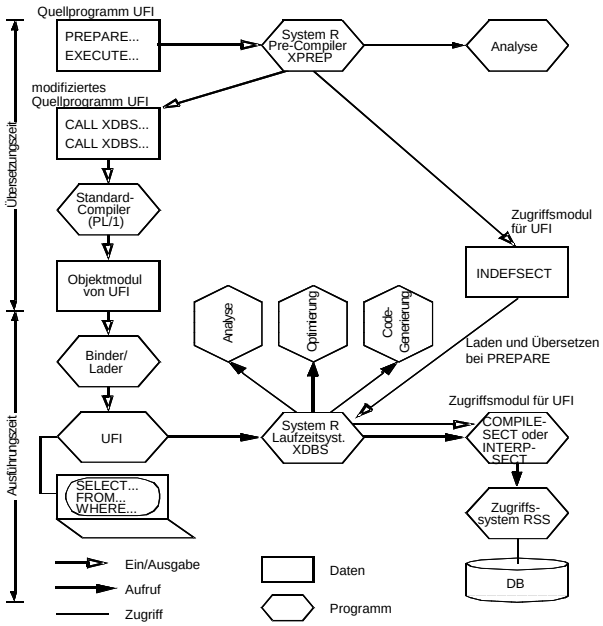
Aufbau eines Zugriffsmoduls



9. Mengenorientierte Schnittstelle

- Übersetzungsverfahren für Datenbanksprachen
  - Vorgehensweise
  - Übersetzung vs. Interpretation
- Formen der Wirtsspracheneinbettung
  - Direkte Einbettung
  - Aufruftechnik
- Anfrageoptimierung
  - Anfragedarstellung
  - Standardisierung und Vereinfachung
  - Restrukturierung und Transformation
  - Kostenmodelle
  - Erstellung und Auswahl von Ausführungsplänen
- Code-Erzeugung
  - Aufbau von Zugriffsmoduln
  - Bindezeitpunkte für Anweisungstypen
- Behandlung von Ad-hoc-Anfragen

Vorbereitung, Übersetzung und Ausführung von Ad-hoc-Anfragen



Spektrum der Bindezeiten in System R

Anweisungstyp	Abschnittstyp	Analyse	Optimierung	Code-Generierung	Ausführung
Normale Operationen (Query, Insert, Delete, Update)	COMPILESECT	Übersetzungszeit			Laufzeit
Nicht-optimierbare Operationen (Create/Drop Table, etc.)	INTERPSECT	Übersetzungszeit			Laufzeit
Operationen auf temporären Objekten	PARSEDECT	Übersetzungszeit	Laufzeit		
Dynamisch definierte Anweisungen (Prepare, Execute)	INDEFSECT	Laufzeit			

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 Ausführung einer Anweisung)
- 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 Erstellung und Auswahl von Ausführungsplänen sehr wichtig