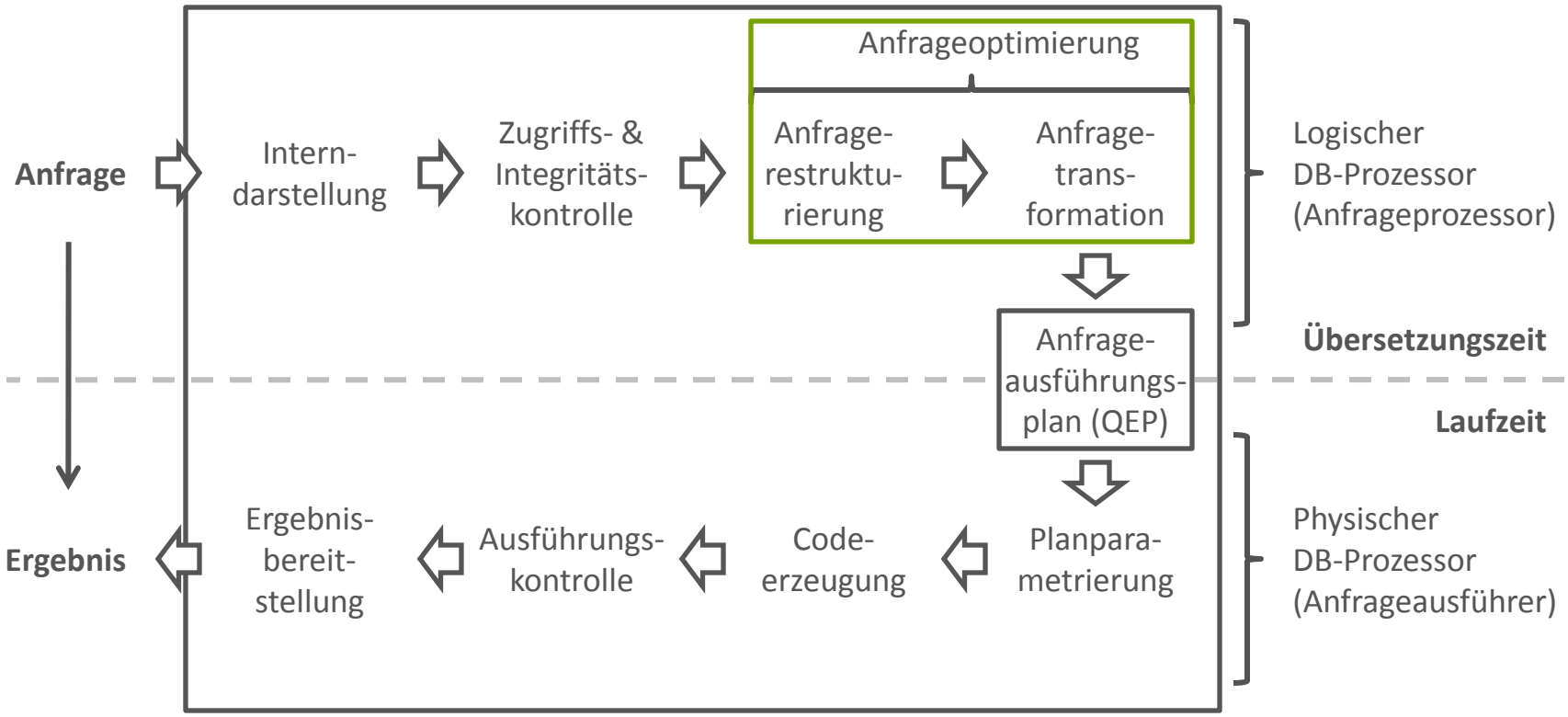
5 Query Optimization


Anfrage
Intern-darstellung
Zugriffs- & Integritäts-kontrolle
Anfrageoptimierung
Anfrage-restruktu-rierung
Anfrage-trans-formation
Logischer DB-Prozessor (Anfrageprozessor)
Anfrage-ausführungs-plan (QEP)
Übersetzungszeit
Laufzeit
Ergebnis
Ergebnis-bereit-stellung
Ausführungs-kontrolle
Code-erzeugung
Planpara-metrierung
Physischer DB-Prozessor (Anfrageausführer)

Ziel

- von der Anfrage (WAS?) zur Auswertung (WIE?)
- Ermittlung des kostengünstigsten Auswerteweges

Zentrales Problem

- globale Optimierung ist im Allgemeinen zu aufwändig
- Fehlen von exakten statistischen Informationen
- Einsatz von Heuristiken

Optimierungsziel

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

Vermeide ein und dieselbe Arbeit mehrmals zu tun

Optimiere Standardsituationen durch effiziente Realisierung (im wesentlichen Programmierung) und garantiere deren Korrektheit

Erkenne Standardsituationen und nutze deren Standardlösungen

Schau voraus, um unnötige Operationen möglichst frühzeitig zu erkennen und zu vermeiden

Wähle den billigsten Weg für das Ausführen von Elementaroperationen

Verkette Elementaroperationen optimal

Berücksichtigung unterschiedlicher Kostenarten

- Berechnungskosten
  - CPU-Kosten
  - Pfadlängen
- E/A-Kosten
  - Anzahl der physischen Referenzen
- Speicherungskosten
  - temporäre Speicherbelegung im DB-Puffer und auf Externspeicher
- Kommunikationskosten (verteilte DBS)
  - Anzahl der Nachrichten
  - Menge der zu übertragenden Daten

Merke

- Kostenarten sind nicht unabhängig voneinander
- in zentralisierten DBS oft gewichtete Funktion von Berechnungs- und E/A-Kosten

Szenario

```
Kunde(KName, KAdr, Kto);
Auftrag(KName, Ware, Menge);
```

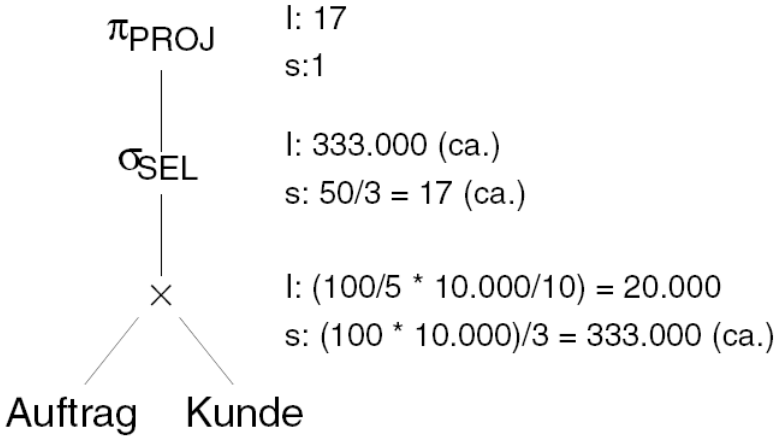
- Kunde: 100 Tupel, Blockungsfaktor 5
- Auftrag: 10.000 Tupel, Blockungsfaktor 10

Anfrage

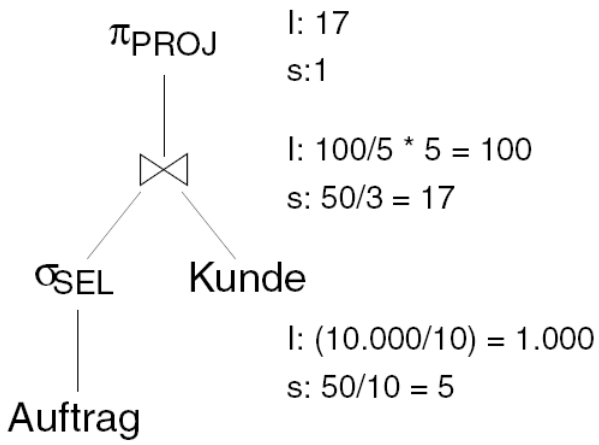
```
SELECT Kunde.KName, Kunde.Kto
  FROM Kunde, Auftrag
 WHERE Kunde.KName = Auftrag.Kname
   AND Auftrag.Ware = 'Kaffee'
```

- Selektivität von 'Ware' = 50/10.000
- Blockungsfaktor für Ergebniseinträge: 50
- Blockungsfaktor für Zwischentupel (Kunde || Auftrag): 3

Direkte Auswertung



Optimierte Auswertung



l = lesen (Eingabe) s = schreiben (Ausgabe)

Auswertung mit Indexunterstützung

1. Nutzung Index über Auftrag(Ware) + Sortierung des Ergebnisses nach KName
2. Nutzung Index über Kunde(KName) für Mischverbund

Anfragerestrukturierung

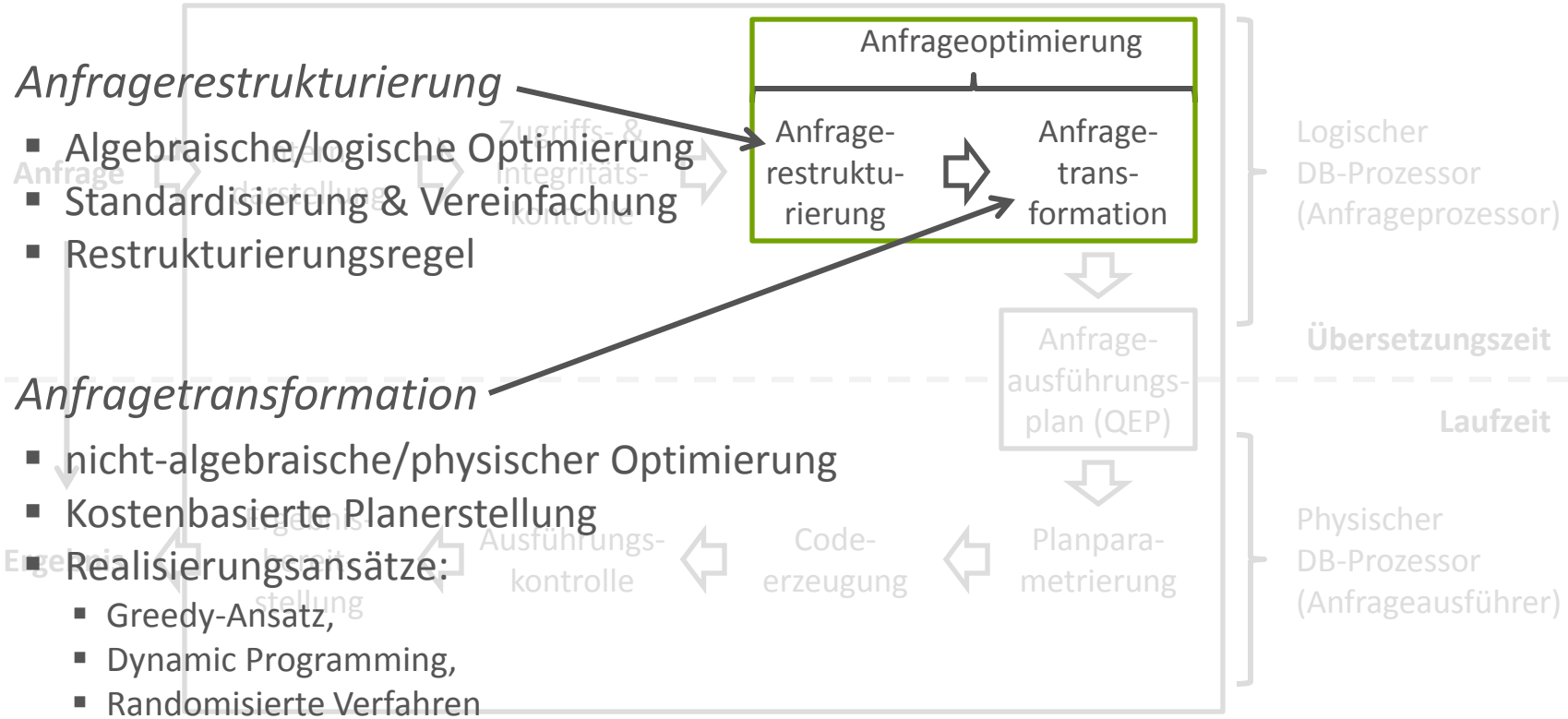
Aufgabeoptimierung erfolgt in zwei Phasen

Anfragerestrukturierung

- Algebraische/logische Optimierung
- Standardisierung & Vereinfachung
- Restrukturierungsregel

Anfragetransformation

- nicht-algebraische/physischer Optimierung
- Kostenbasierte Planerstellung
- Realisierungsansätze:
  - Greedy-Ansatz,
  - Dynamic Programming,
  - Randomisierte Verfahren



Algebraische bzw. logische Optimierung

- Modellbasierte algebraische Umformung für effizientere Ausführung
- Zielt auf globale Verbesserung des Anfragegraphen ab
- Anwendung von heuristischen Regeln

Basis

- Äquivalenzen in der relationalen Algebra

Schritte

- Standardisierung von Prädikaten
- Vereinfachung von Prädikaten
- Auflösen von Verschachtelungen
- Anwenden Restrukturierungsregeln

Standardisierung

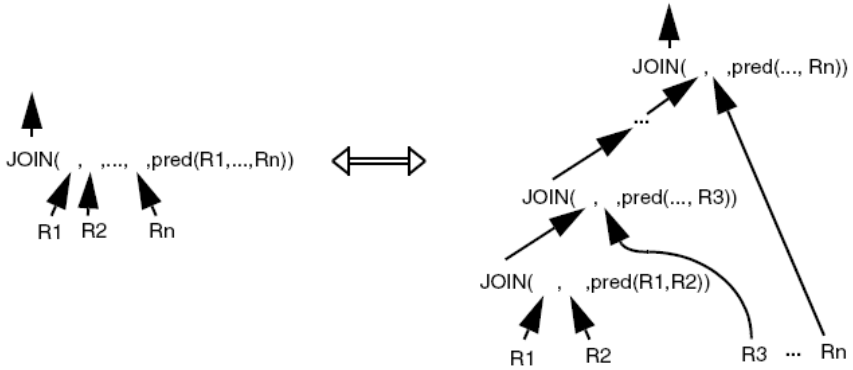
- Wahl einer Normalform
  - konjunktive Normalform (P_{11} OR ... OR P_{1n}) AND ... AND (P_{m1} OR ... OR P_{mp})
  - disjunktive Normalform (P_{11} AND ... AND P_{1q}) OR ... OR (P_{r1} AND ... AND P_{rs})
  - Pränex-Normalform (Verschiebung der Quantoren)
    z.B. zur Auflösung geschachtelter SELECT-Anweisungen

Vereinfachung

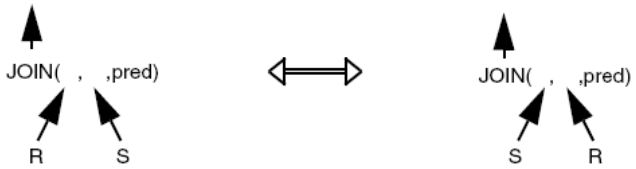
- äquivalente Ausdrücke können einen unterschiedlichen Grad an Redundanz besitzen
  - Idempotenzregeln
  - Ausdrücke mit “leeren Relationen”
- Behandlung/Eliminierung gemeinsamer Teilausdrücke
  - ($A_1 = a_{11}$ OR $A_1 = a_{12}$) AND ($A_1 = a_{12}$ OR $A_1 = a_{11}$)
- Konstanten-Propagierung (allg. Hüllenbildung der Qualifikationsprädikate)
  - A op B AND B = const. A op const. AND B = const.
- nicht-erfüllbare Ausdrücke
  - A ≥ B AND B > C AND C ≥ A A > A → false

- Nutzung von Information über semantische Integritätsbedingungen
  - A ist Primärschlüssel: π_A → keine Duplikateliminierung erforderlich
  - Auswertung hinterlegter Regeln
  - FAM-STAND = 'verh.' AND STEUERKLASSE ≥ 3
    (FAM-STAND = 'verh.' AND STEUERKLASSE = 1) → false
- Umformungs- und Idempotenzregeln 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: 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
- Umformungsregeln für quantifizierte Ausdrücke
- Auflösen geschachtelter Unteranfragen

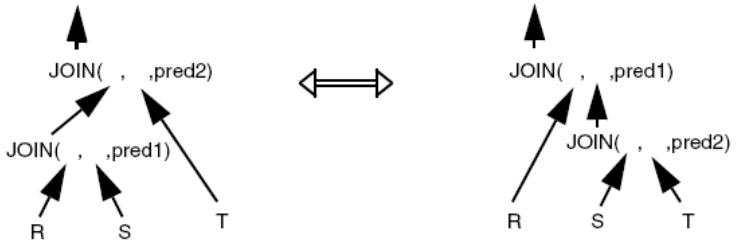
Regel 1: Bilden von binären Verbunden[1)]



Regel 2: Kommutativität von Verbunden

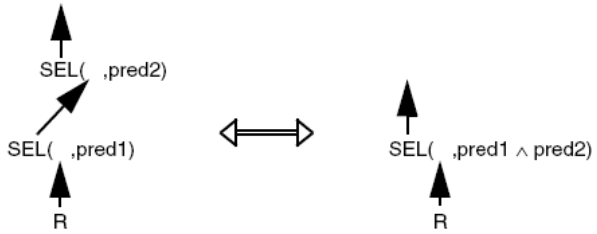


Regel 3: Assoziativität von Verbunden



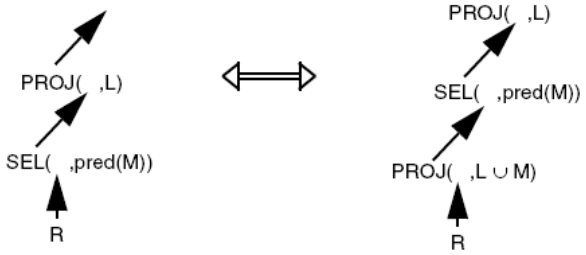
1) Hier wird davon ausgegangen, daß das Prädikat pred(R1, ,Rn) in einzelne Prädikate zerlegbar ist, die über den jeweiligen Verbundpartnern definiert sind.

Regel 4: Gruppierung von Selektionen (Restriktion)



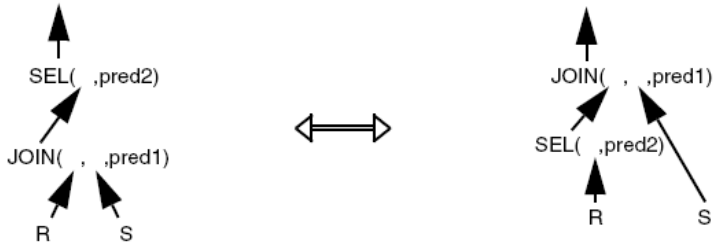
Regel 5: Gruppierung von Projektionen

Regel 6: Vertauschen von Selektion und Projektion[2)]

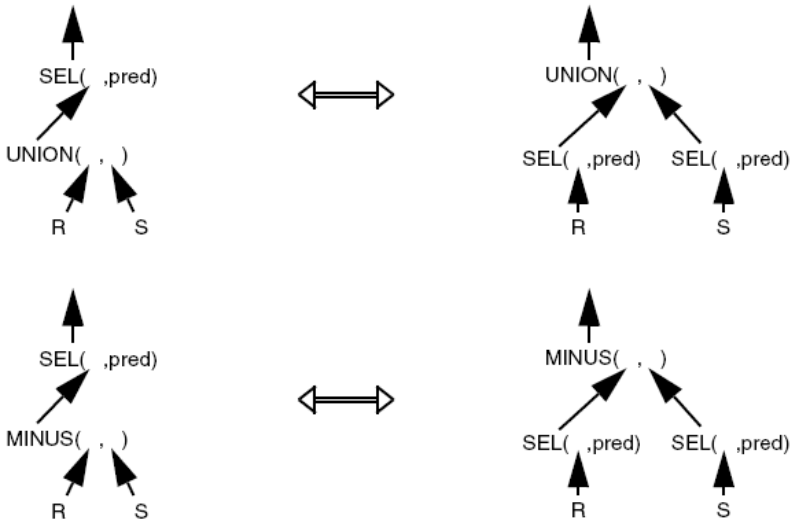


2) M bezeichnet die Attributfolge der Relation R, über der das Selektionsprädikat definiert ist.

Regel 7: Vertauschen von Selektion und Verbund

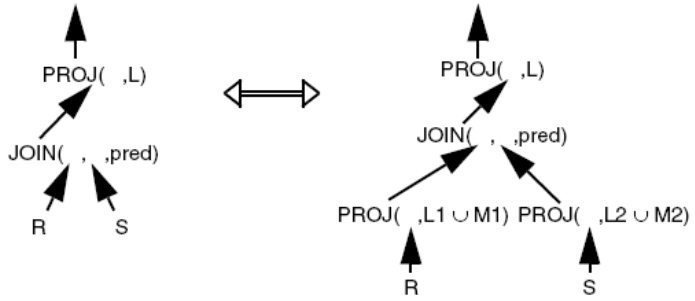


Regel 8: Vertauschen von Selektion und Vereinigung bzw. Differenz[3)]

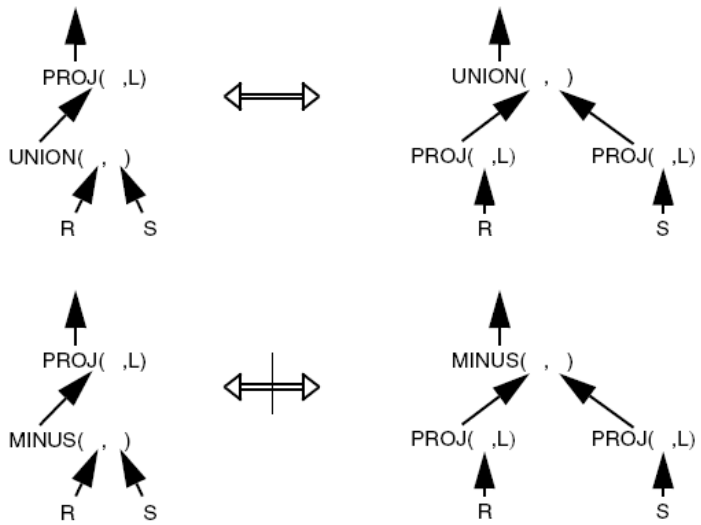


3) Aufgrund der Vereinigungsverträglichkeit der Relation R und S, kann das Selektionsprädikat auch auf R bzw. S angewendet werden

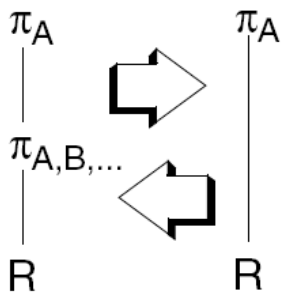
Regel 9: Vertauschen von Projektion und Verbund[4)]



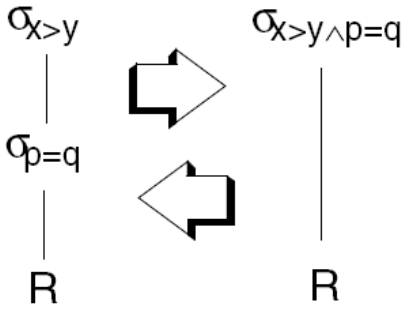
Regel 10: Vertauschen von Projektion und Vereinigung bzw. Differenz



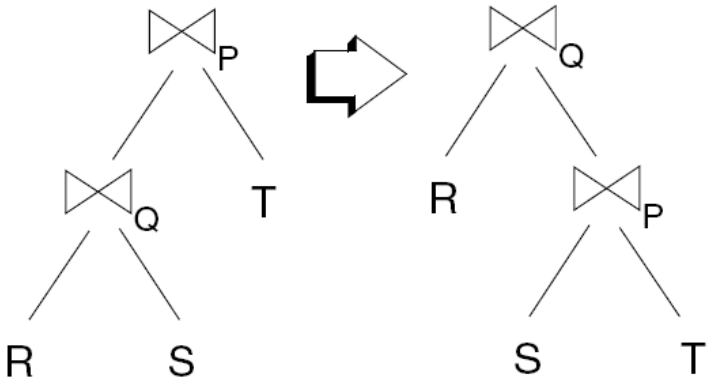
4) L sei die zu projizierende Attributfolge, die sich in die zwei Attributfolgen L1 für die Relation R und L2 für die Relation S zerlegen läßt. M sei die Attributfolge, die im Verbundprädikat pred angesprochen ist. Diese läßt sich analog zu L auch in zwei Attributfolgen M1 und M2 für die Relation R bzw. S zerlegen.



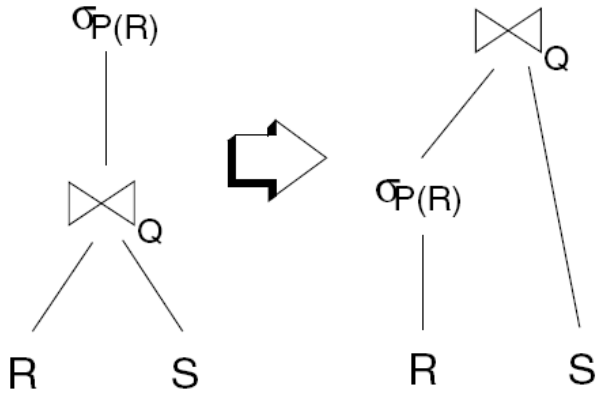
a) Schachtelung von Projektionen



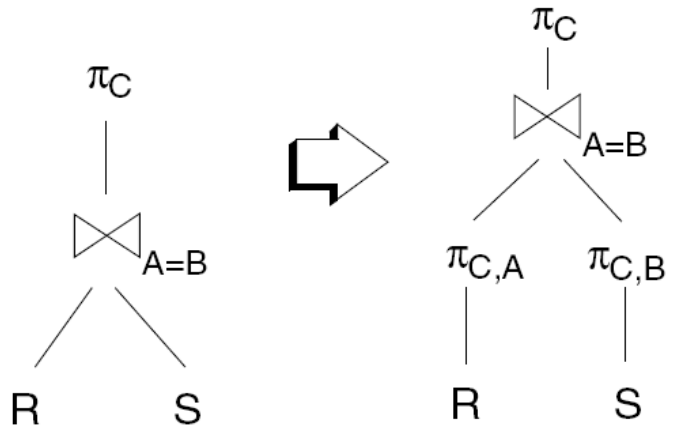
b) Zusammenfassen und Aufspalten von Selektionen



c) Assoziativität von Verbundoperationen



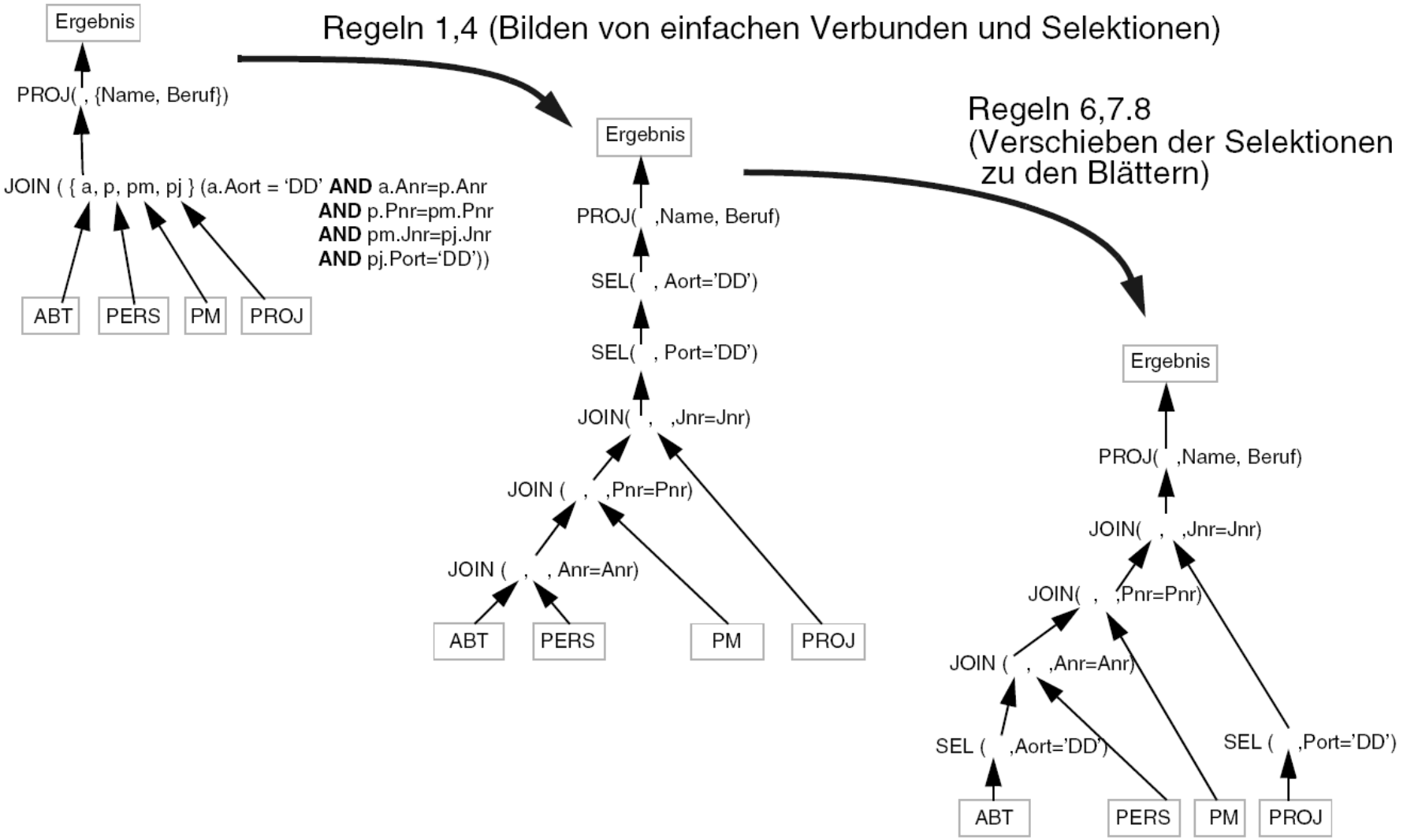
d) Vertauschen von Selektion und Verbund

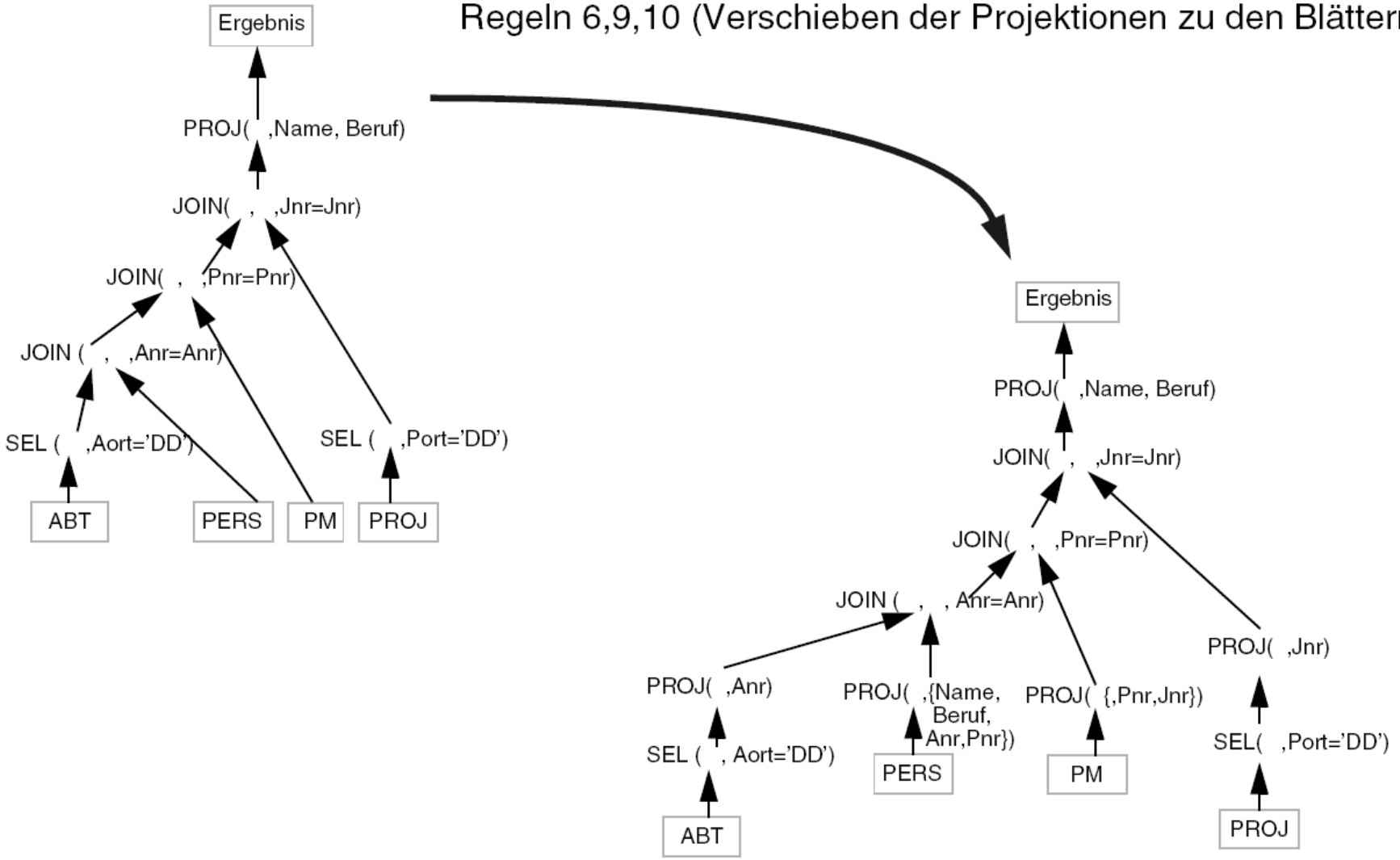


e) Vertauschen von Projektion und Verbund

Algorithmus zur Anfragerestrukturierung - vereinfachte Vorgehensweise

- Zerlegen von komplexen Verbundoperationen in binäre Verbunde
  (Bilden von binären Verbunden, Regel 1)
- Separiere Selektionen mit mehreren Prädikatstermen in separate Selektionen mit jeweils einem Prädikatsterm
  (Regel 4)
- Führe Selektionen so früh wie möglich aus, d.h., schiebe Selektionen hinunter zu den Blättern des Anfragegraphen
  (engl. selection push-down, Regel 5, 6 und 7)
- Fasse einfache Selektionen zusammen, d.h. gruppiere aufeinanderfolgende Selektionen (derselben Relation)
  (Regel 4)
- Führe Projektionen so früh wie möglich aus, d.h. schiebe Projektionen hinunter zu den Blättern des Anfragegraphen
  (engl. projection push-down, Regel 6, 9 und 10);
  vermeide dabei die teure Duplikateliminierung!


Ergebnis
PROJ(, {Name, Beruf})
JOIN ({ a, p, pm, pj } (a.Aort = 'DD' AND a.Anr=p.Anr
AND p.Pnr=pm.Pnr
AND pm.Jnr=pj.Jnr
AND pj.Port='DD'))
ABT
PERS
PM
PROJ
Regeln 1,4 (Bilden von einfachen Verbunden und Selektionen)
Ergebnis
PROJ(,Name, Beruf)
SEL(, Aort='DD')
SEL(, Port='DD')
JOIN(, ,Jnr=Jnr)
JOIN (, ,Pnr=Pnr)
JOIN (, , Anr=Anr)
ABT
PERS
PM
PROJ
Regeln 6,7.8
(Verschieben der Selektionen
zu den Blättern)
Ergebnis
PROJ(,Name, Beruf)
JOIN(, ,Jnr=Jnr)
JOIN(, ,Pnr=Pnr)
JOIN (, ,Anr=Anr)
SEL (,Aort='DD')
SEL (,Port='DD')
ABT
PERS
PM
PROJ


Regeln 6,9,10 (Verschieben der Projektionen zu den Blättern)
Ergebnis
PROJ(,Name, Beruf)
JOIN(, ,Jnr=Jnr)
JOIN(, ,Pnr=Pnr)
JOIN (, ,Anr=Anr)
SEL (,Aort='DD')
SEL (,Port='DD')
ABT
PERS
PM
PROJ
Ergebnis
PROJ(,Name, Beruf)
JOIN(, ,Jnr=Jnr)
JOIN(, ,Pnr=Pnr)
JOIN (, , Anr=Anr)
PROJ(,Jnr)
PROJ(,Anr)
PROJ(,{Name, Beruf, Anr,Pnr})
PROJ({,Pnr,Jnr})
SEL (, Aort='DD')
SEL(,Port='DD')
PERS
PM
ABT
PROJ

Fall 1: Typ-A-Schachtelung

- Innerer SWF-Block ist nicht korreliert und berechnet einzelnes Aggregatwert
- Auflösung: Berechnung des Aggregatwertes und Einsetzen in äußere Anfrage

```
SELECT BestNr
  FROM Bestellung
 WHERE ProdNr = (SELECT MAX(ProdNr)
                   FROM Produkt
                  WHERE Preis < 100)
```

Fall 2: Typ-N-Schachtelung

- Innerer SWF-Block ist nicht korreliert und liefert Menge von Tupeln
- Auflösung: Umschreiben in symmetrische Form

```
SELECT BestNr
  FROM Bestellung
 WHERE ProdNr in (SELECT ProdNr
  FROM Produkt WHERE Preis < 100)
```

⇨

```
SELECT BestNr
  FROM Bestellung B, Produkt P
 WHERE B.ProdNr = P.ProdNr
   AND P.Preis < 100
```

Fall 3: Typ-J-Schachtelung

- Entfaltung korrelierter Unteranfragen

```
SELECT BestNr
  FROM Bestellung B
 WHERE ProdNr in
       (SELECT ProdNr FROM Projekt P
         WHERE P.ProjNr = B.BestNr
           AND P.Budget > 100.000)
```

```
SELECT BestNr
  FROM Bestellung B, Produkt P
 WHERE B.ProdNr = P.ProdNr
   AND P.ProjNr = B.BestNr
   AND P.Budget > 100.000
```

Fall 4: Typ-JA-Schachtelung

- Entfaltung korrelierter Unteranfragen mit Aggregation

```
SELECT BestNr
  FROM Bestellung B
 WHERE ProdNr in
       (SELECT MAX(ProdNr)
          FROM Projekt P
         WHERE P.ProjNr = B.BestNr
           AND P.Budget > 100.000)
```

```
SELECT BestNr
  FROM Bestellung B
 WHERE ProdNr IN
       (SELECT ProdNr FROM
           (SELECT ProjNr, MAX(ProdNr)
            FROM Projekt
            GROUP BY ProjNr) P
          WHERE P.ProjNr = B.BestNr
            AND P.Budget > 100.000)
```

- Weiteres Entschachteln analog zu Fall 3

Ansatz

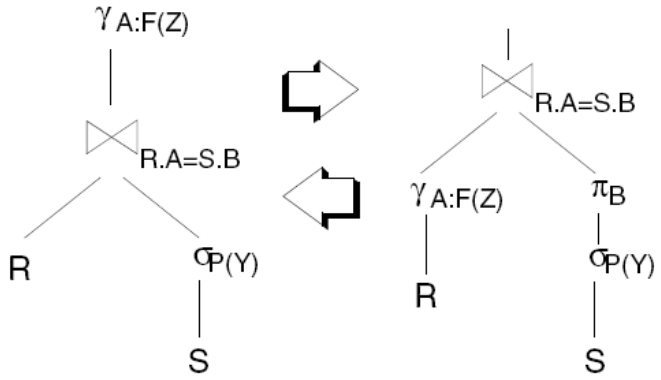
- Integration der Gruppierung in die logische Optimierung
- Reduktion der Kardinalität eines Datenstroms vor einer Verbundoperation

Invariantes Gruppieren (invariant grouping/group-by push-down)

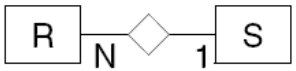
- Verschiebung eines Gruppierungsoperators über eine Verbundbedingung, falls
  - Verbundpartner trägt nicht direkt zum Ergebnis bei
  - Verbundpartner realisiert eine implizite Selektion
  - Gruppierungsattribute haben die rolle eines Fremdschlüssels zu den Verbundpartnern in der nachfolgenden Verbundoperation
- Gruppierung ist invariant zu seiner Position innerhalb des Operatorengraphens

Beispiel

```
SELECT L_ORDERKEY, L_PARTKEY, SUM(L_QUANTITY) AS SUM_QUAN
  FROM TPCD.LINEITEM, TPCD.ORDERS, TPCD.PART
 WHERE L_ORDERKEY = O_ORDERKEY
   AND L_PARTKEY = P_PARTKEY
   AND O_ORDERSTATUS <> 'F'
   AND P_CONTAINER <> 'LG_BAG'
 GROUP BY L_ORDERKEY, L_PARTKEY;
```

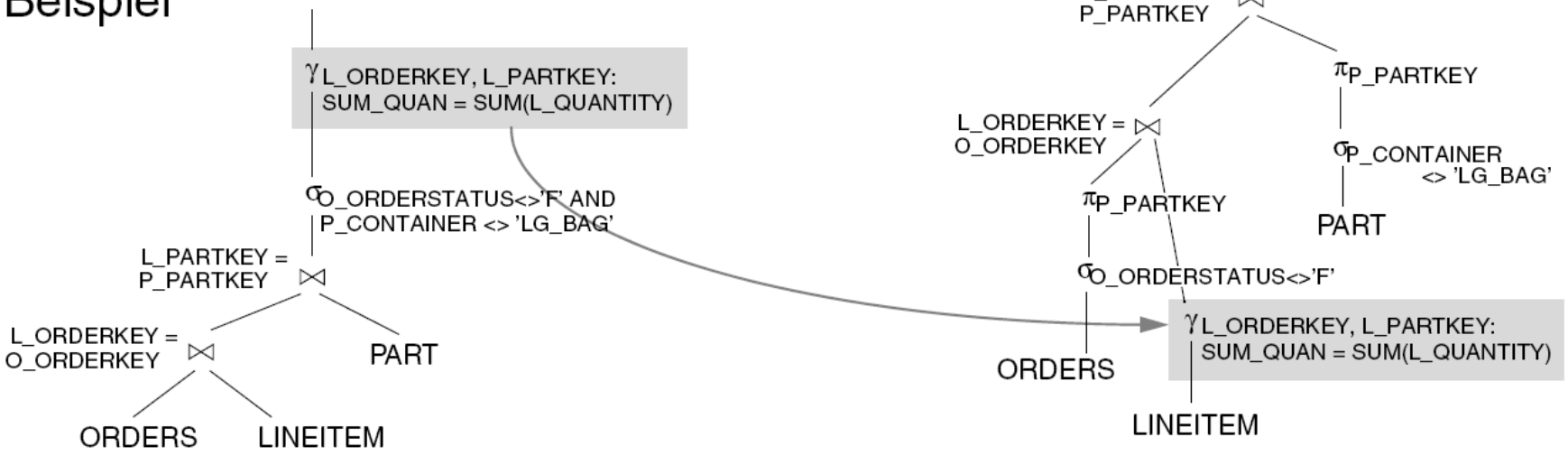


wobei zusätzlich gilt:



- A und Z sind Attribute aus R
- B und Y sind Attribute aus S
- F() ist Aggregationsfunktion

Beispiel



a) Operatorengraph *vor* Restrukturierung

b) Operatorengraph *nach* Restrukturierung

Ansatz

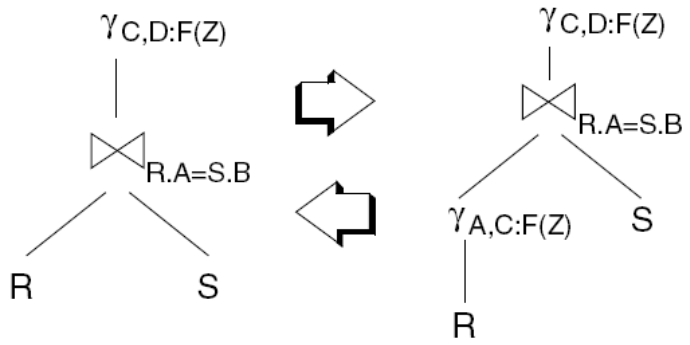
- Invariantes Gruppierung zu restriktiv
- Erweiterung zum punktuellen Einbringen von Projektionsoperatoren

Frühzeitiges Vorgruppieren (simple coalescing grouping/eager group-by)

- Einschub eines Gruppierungsoperators vor einer Verbundoperation, falls
  - Gruppierungsbedingung enthält die Verbundattribute zu den Verbundpartnern
  - Attribute der Aggregationsoperation beziehen sich nicht auf Attribute der Partnerrelation

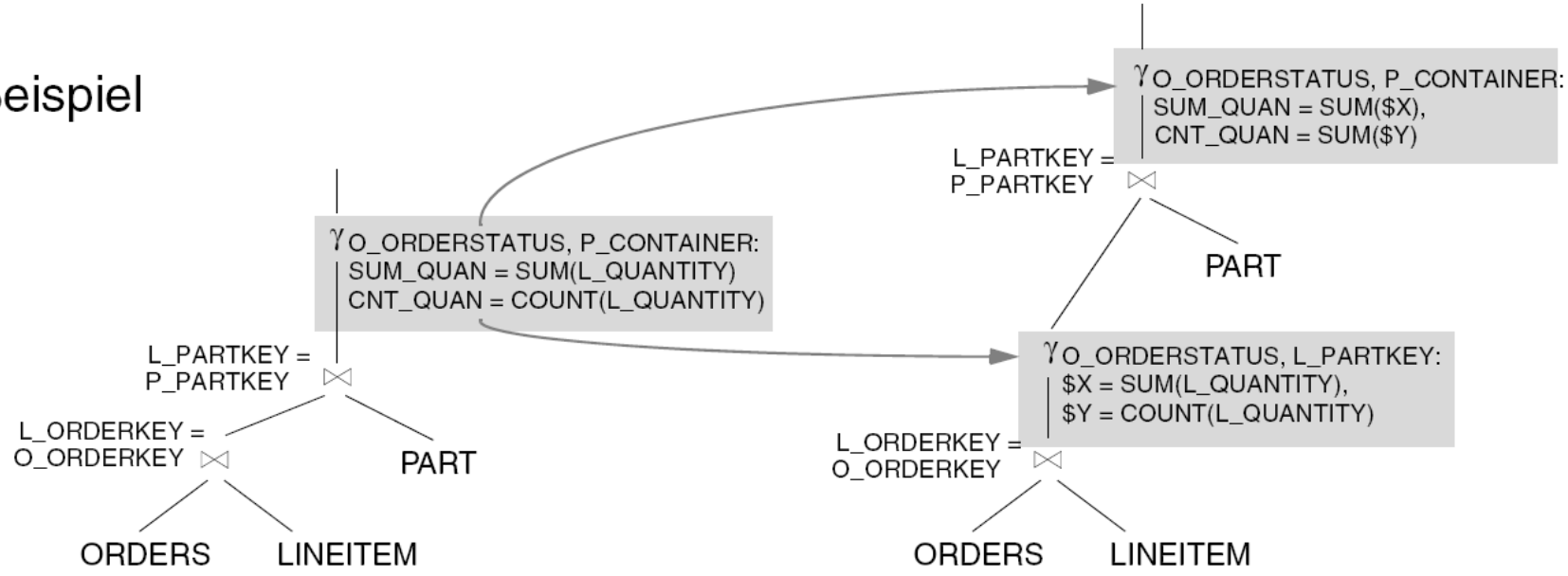
Beispiel

```
SELECT O_ORDERSTATUS, P_CONTAINER,
       SUM(L_QUANTITY) AS SUM_QUAN,
       COUNT(L_QUANTITY) AS CNT_QUAN
  FROM TPCD.LINEITEM,
       TPCD.ORDERS,
       TPCD.PART
 WHERE L_ORDERKEY = O_ORDERKEY
   AND L_PARTKEY = P_PARTKEY
 GROUP BY O_ORDERSTATUS, P_CONTAINER;
```



wobei zusätzlich gilt:
- A, C und Z sind Attribute aus R
- B und D sind Attribute aus S
- F() ist Aggregationsfunktion

Beispiel

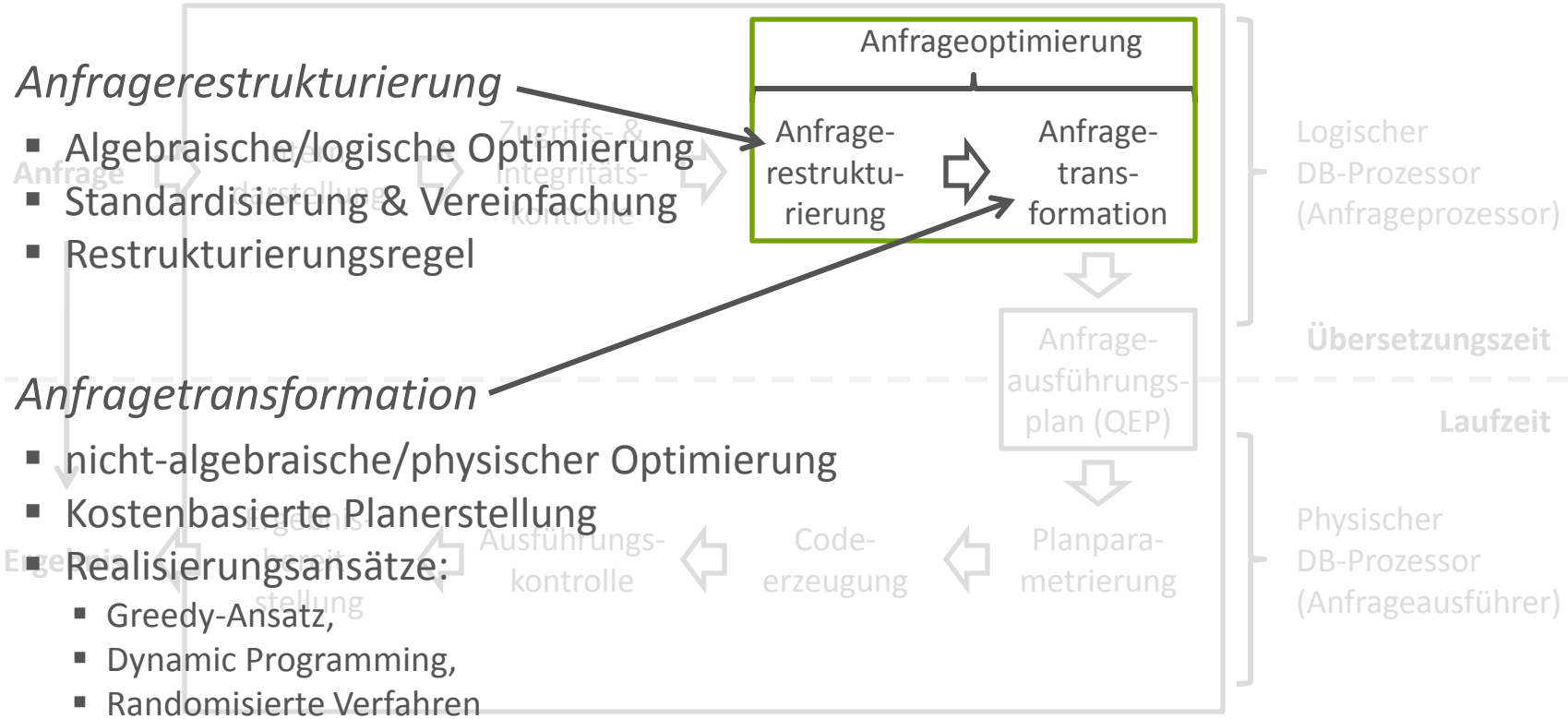


a) Operatorengraph *vor* Restrukturierung

b) Operatorengraph *nach* Restrukturierung

Anfragetransformation

Aufgabeoptimierung erfolgt in zwei Phasen



Anfragerestrukturierung

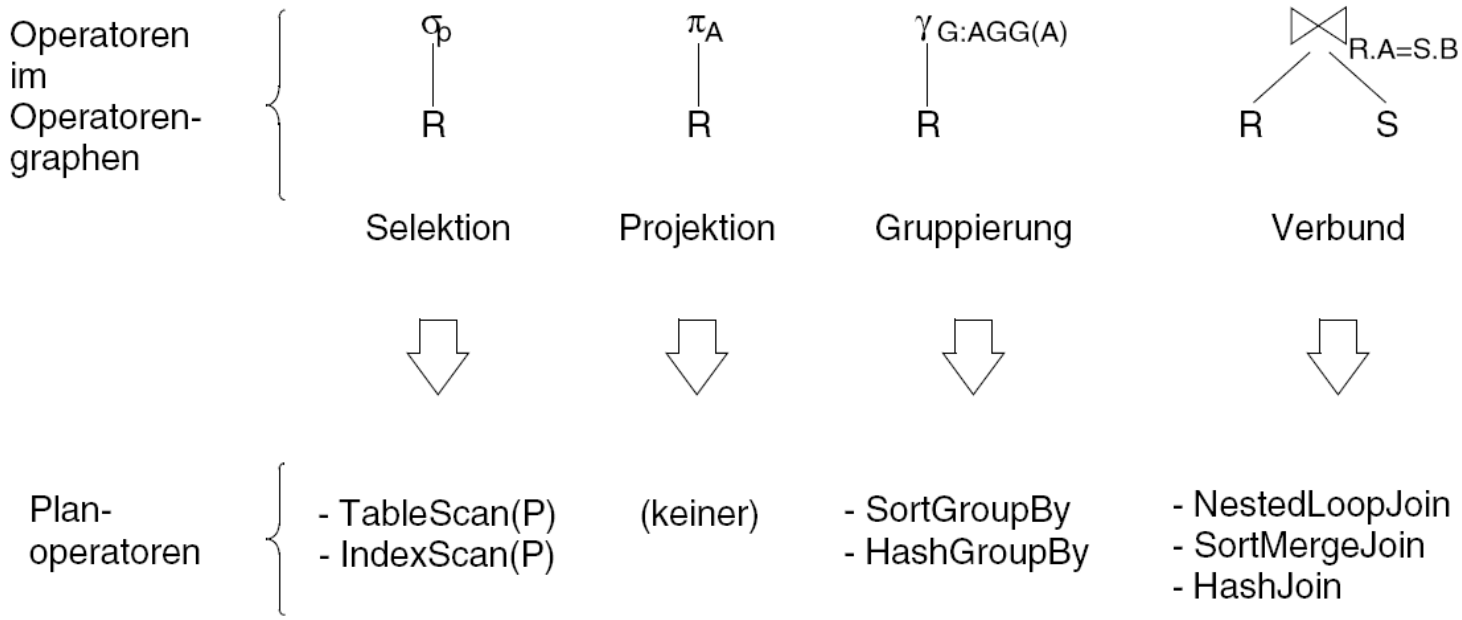
- Algebraische/logische Optimierung
- Standardisierung & Vereinfachung
- Restrukturierungsregel

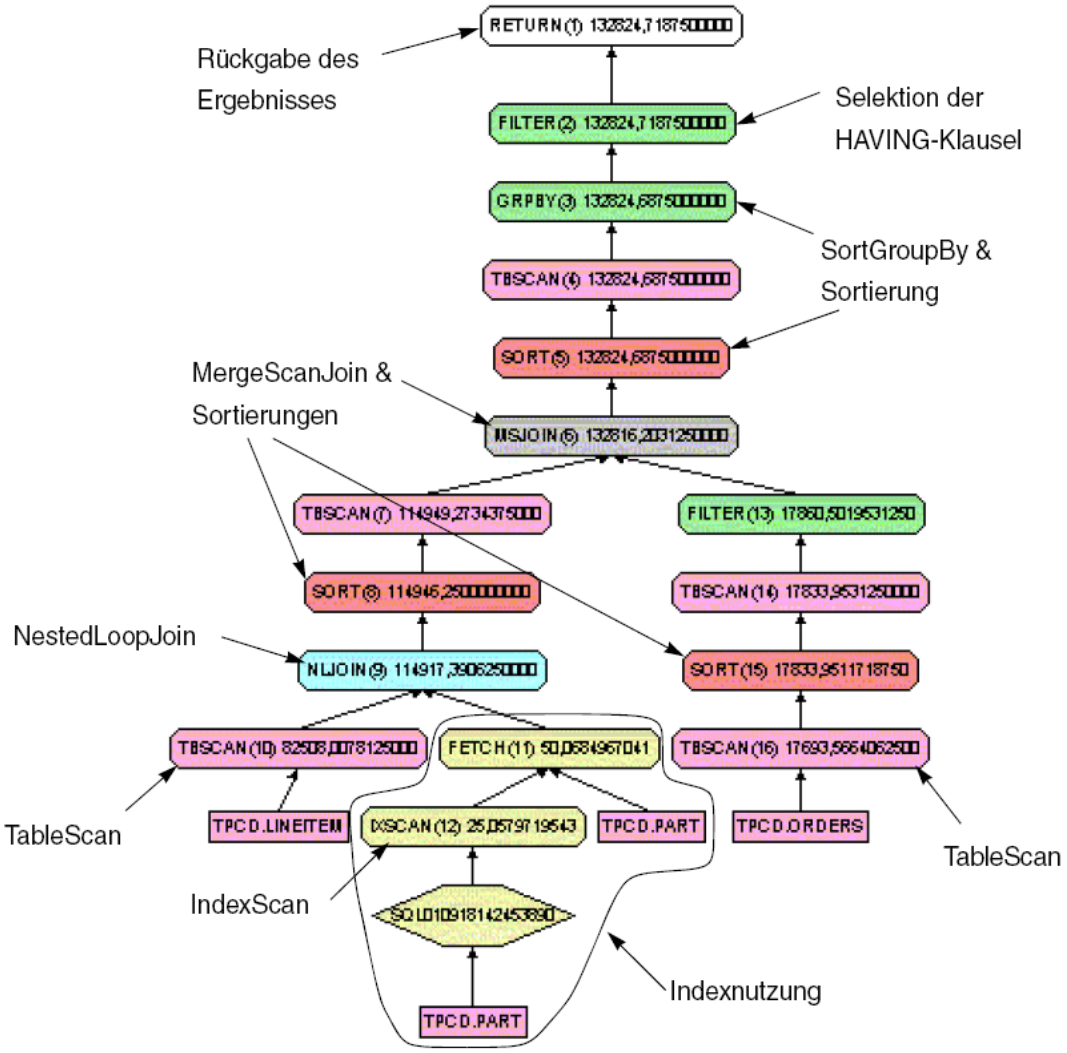
Anfragetransformation

- nicht-algebraische/physischer Optimierung
- Kostenbasierte Planerstellung
- Realisierungsansätze:
  - Greedy-Ansatz,
  - Dynamic Programming,
  - Randomisierte Verfahren

Nicht-algebraische/physischer Optimierung

- Abbildung eines relationalen Operatorengraphen auf physisch ausführbare Operatoren und deren Ausführungsreihenfolge
  -> also Erstellung eines Ausführungsplans




Rückgabe des Ergebnisses
RETURN(1) 132824,71875
Selektion der HAVING-Klausel
FILTER(2) 132824,71875
GRPBY(3) 132824,6875
SortGroupBy & Sortierung
TBSCAN(4) 132824,6875
SORT(5) 132824,6875
MergeScanJoin & Sortierungen
MSJOIN(6) 132816,2
TBSCAN(7) 114949,2734375
FILTER(13) 17860,5019531250
SORT(8) 114945,25
TBSCAN(14) 17833,953125
NestedLoopJoin
NLJOIN(9) 114917,390625
SORT(15) 17833,95117187
TBSCAN(10) 82508,0078125
FETCH(11) 50,0584967041
TBSCAN(16) 17693,5664062
TableScan
TPCD.LINEITEM
IXSCAN(12) 25,0579719543
TPCD.PART
TPCD.ORDERS
TableScan
IndexScan
SQL0109181424538900
Indexnutzung
TPCD.PART

Problem

- Unterschiedliche Implementierungen (z.B. Join) bzw. Abbildungsvarianten (z.B. Indexnutzung)

Teilprobleme

- Gruppierung von direkt benachbarten Operatoren zur Auswertung durch einen Planoperator
  - Beispiel: Verbund mit Selektionen und/oder Projektionen auf den beteiligten Relationen lässt sich durch einen speziellen Planoperator ersetzen
- Verknüpfungsreihenfolge bei Verbundoperationen
  - Ziel: minimale Kosten für die Operationsfolge
  - Heuristik: Minimierung der Größe der Zwischenergebnisse, d. h. die kleinsten (Zwischen-)Relationen immer zuerst zu verknüpfen
- Erkennung gemeinsamer Teilbäume
  - Einmalige Berechnung
  - dafür nötig: Zwischenspeichung der Ergebnisrelation

Eingabe

- Algebraisch optimierter Anfragegraph
- Existierende Speicherungsstrukturen und Zugriffspfade
- Kostenmodell

Ausgabe

- optimaler (zumindest: guter) Ausführungsplan

Rahmenbedingungen

- Fatale Annahmen
  - Alle Datenelemente und alle Attributwerte sind gleichverteilt
  - Suchprädikate in Anfragen sind unabhängig
  - beide Annahmen sind (im allgemeinen Fall) falsch!
  - Beispiel: (GEHALT ≥ '100K') AND (ALTER BETWEEN 20 AND 30)
    mit Gehalt: [10K - 1M] und Alter: [20 - 65]
    -> lineare Interpolation, Multiplikation von Wahrscheinlichkeiten
- Riesiger Suchraum
  - Beispiel: Für n-ären Join gibt es n! mögliche Joinreihenfolgen
- Begrenzte Ressourcen
  - Kosten der Anfrageoptimierung sollen nicht die erzielte Reduktion von Ausführungskosten überschreiten

Vereinfachte Vorgehensweise

- Generiere alle “vernünftigen” logischen Ausführungspläne zur Auswertung der vorliegenden Anfrage
- Vervollständige die Ausführungspläne durch Einzelheiten der physischen Datenrepräsentation (Sortierreihenfolge, Zugriffspfadmerkmale, statistische Information)
- Bewertung der generierten Alternativen und Auswahl des billigsten Ausführungsplanes gemäß dem vorgegebenen Kostenmodell

Problem: Entstehung sehr großer Suchräume bei komplexen Anfragen

- für jeden Planoperator liegen verschiedene Methoden (Implementierungen) vor
- Operationsreihenfolgen (z. B. bei Mehrfachverbunden) können variiert werden
- Beispiel: 10^{70} mögliche Ausführungspläne bei einer Anfrage mit 15 Verbunden
  -> Übersetzen einer Anfrage dauert oftmals länger als die eigentliche Ausführung !!!
- Optimierereinstellungen (z.B. Oracle): OPTIMIZER_MAX_PERMUTATIONS
  - „This parameter controls the maximum number of permutations that the CBO considers when generating execution plans for SQL statements with joins. The range of values is 4 to 262140. However, reducing its value can result in the optimizer missing an optimal join permutation. Default: 80,000“

Kostenmodell

- Dient zur Erstellung eines Kostenvoranschlags für einen Ausführungsplan
- Hat entscheidenden Einfluss auf die Güte der Anfrageoptimierung
- Berücksichtig verschiedene Kostenfaktoren
- Fasst Kostenfaktoren in gewichtete Kostenformel zusammen

Beispiele

- gewichtetes Maß für E/A- und CPU-Auslastung
  C = #physischer Seitenzugriffe + W * (#Aufrufe des Zugriffssystems)
  - W ist das Verhältnis des Aufwandes für einen Aufruf des Zugriffssystems zu einem Seitenzugriff (Kalibrierung in Abhängigkeit von der jeweiligen Plattform)
  - Ziel der Gewichtung: Minimierung der Kosten in Abhängigkeit des Systemzustandes
- CPU-bound System: höherer E/A, geringerer CPU-Aufwand
  - Beispiel für W: WCPU = (#Aufruf des Zugriffsystems) / (#Instr. pro E/A-Vorgang)
- I/O-bound System: möglichst geringer E/A, höherer CPU-Aufwand
  - Beispiel für W: WIO = (#Aufruf des Zugriffsystems) / (#Instr. pro E/A-Vorgang + Zugriffszeit * Mips-Rate)

Anforderungen an die Ausführungsplansuche

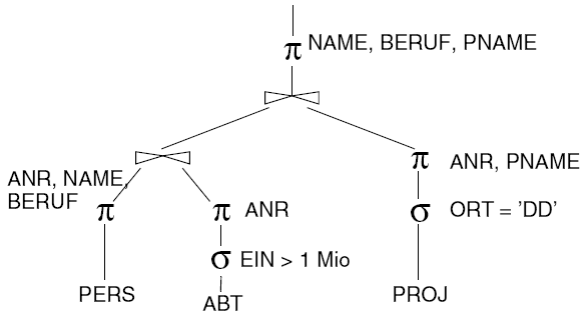
- (möglichst) optimaler Plan
- möglichst schnell
- mit einer möglichst kleiner Anzahl generierter Pläne

Unterschiedliche Strategieklassen

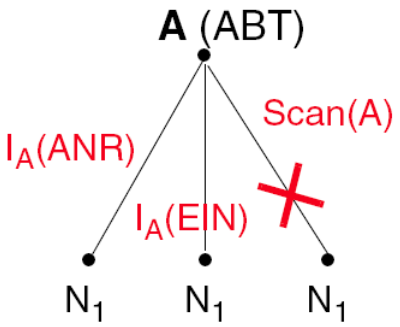
- voll-enumerativ
  - Findet global optimalen Plan
  - Oft zu aufwändig
- beschränkt-enumerativ
  - Frühzeitiger Ausschluss von Bereichen des Suchraums
  - Findet oft nur lokal optimalen Plan
- Zufallsgesteuert
  - Vermeidung des Hängenbleibens an lokal optimalen Pläne durch Wahl zufälliger Nachbarn
  - Genetische Algorithmen und die Strategien des ‘simulated annealing’

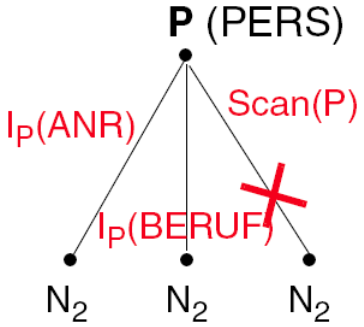
Beispiel

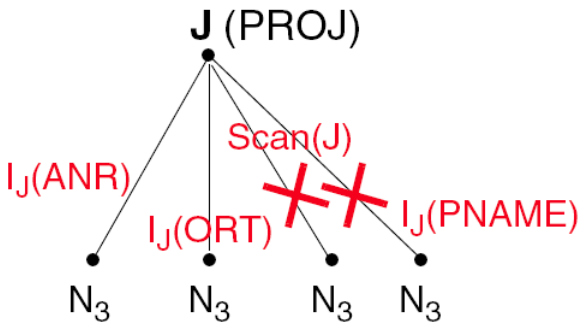
```
SELECT P.NAME, P.BERUF, J.PNAME
  FROM PERS P, ABT A, PROJ J
 WHERE A.EIN > 1000000
   AND J.ORT = 'DD'
   AND A.ANR = P.ANR
   AND A.ANR = J.ANR
```



Mögliche Zugriffspfade für die einzelnen Relationen

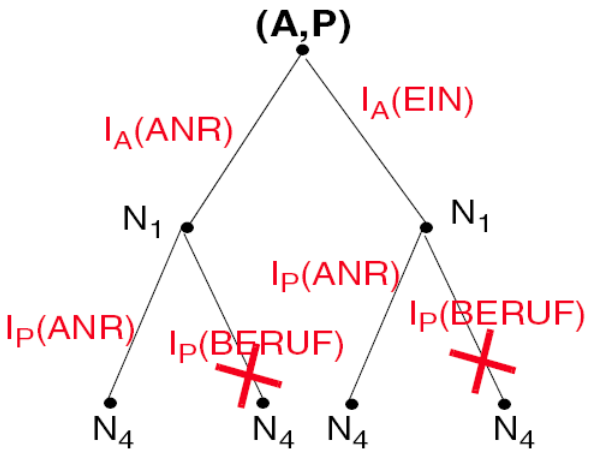


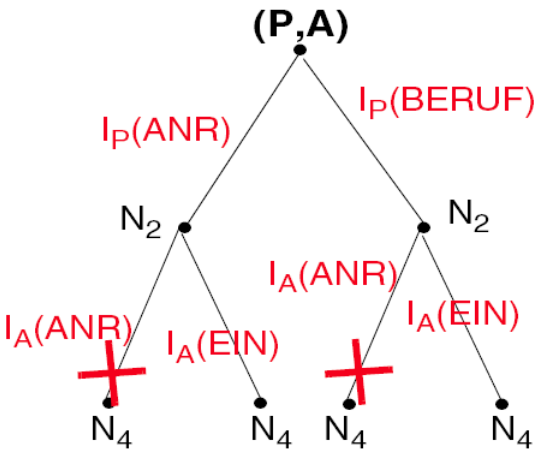


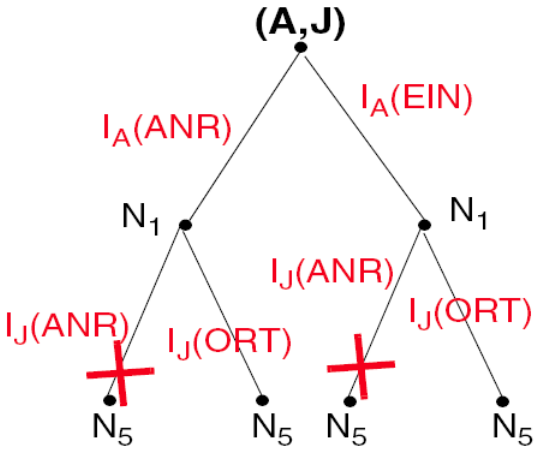


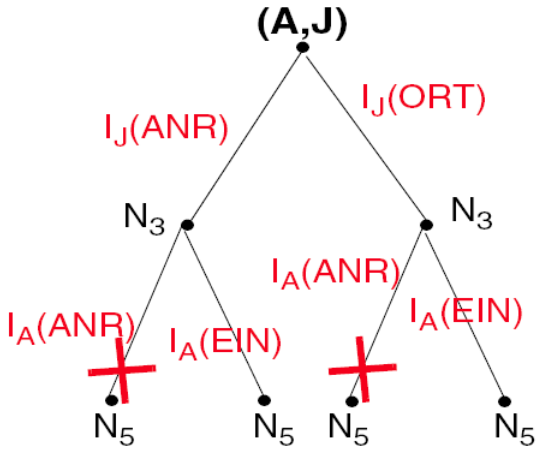
- Relation ABT: DB-Scan und Index über ANR und EIN
- Relation PERS: DB-Scan und Index über ANR und BERUF
- Relation PROJ: DB-Scan und Index über ANR, ORT und PNAME

Erweiterter Lösungsbaum für „Nested-Loop“-Verbund









Ansatz

- Vollständigen Plan schrittweise aus partiellen Lösungen (Teilplänen) konstruieren
- Hinzunahme von Verbund mit minimalen Kosten zu optimalen Teilplan

Algorithmus

```
Input: Verbundanfrage Q mit n Relationen r_1, ..., r_n
Output: Anfrageplan optPlan
foreach r_i, r_j ∈ Q mit i ≠ j do
begin
   bestimme |r_i ⋈ r_j|
end
optPlan := (r_i ⋈ r_j) mit |r_i ⋈ r_j| ist minimal
for k := 3 to n do
begin
   foreach r_k ∈ {Q - optPlan} do
   begin
      bestimme |optPlan ⋈ r_k|
   end
   optPlan := (optPlan ⋈ r_k) mit |optPlan ⋈ r_k| ist minimal
end
return optPlan
```

Beispiel

- Anfrage: r(Bestellung) ⋈ r(Produkt) ⋈ r(Kunde) ⋈ r(Lieferant)

Schritt	Plan	Ergebnisgröße	
1	r(Kunde) ⋈ r(Produkt)	5.000.000	
	r(Kunde) ⋈ r(Lieferant)	1.000.000	
	r(Kunde) ⋈ r(Bestellung)	20.000	
	r(Produkt) ⋈ r(Lieferant)	5.000	
	r(Produkt) ⋈ r(Bestellung)	20.000	
	r(Lieferant) ⋈ r(Bestellung)	2.000.000	
2	(r(Produkt) ⋈ r(Lieferant)) ⋈ r(Bestellung)	20.000	
	(r(Produkt) ⋈ r(Lieferant)) ⋈ r(Kunde)	5.000.000	
Ergebnis	((r(Produkt) ⋈ r(Lieferant)) ⋈ r(Bestellung)) ⋈ r(Kunde)		Bester Plan

Eigenschaften

- Vorteile
  - Nur wenige Pläne zu berücksichtigen (hier 8 anstelle von 5!=120)
- Nachteile
  - Nur links-orientierte Bäume
    a-priori Ausschluss von Lösungsmöglichkeiten
    z.B. : (r(Bestellung) ⋈ r(Produkt)) ⋈ (r(Kunde) ⋈ r(Lieferant))
  - Keine Garantie, dass optimaler Plan gefunden wird

Variante

- Berücksichtigung des Selektivitätsfaktors der Relation anstelle der Größe der Zwischenergebnisse

Ansatz

- Beobachtung: optimale Lösung kann nur optimale Teillösungen enthalten
  - Problem in abhängige Teilprobleme zerlegen und lokale optimale Lösung erstellen
  - mehrfach auftretende Teilprobleme nur einmal lösen
- Anpassung an Anfrageoptimierung
  - Verbundreihenfolge zwischen n Relationen durch Optimierung der Teilverbunde zwischen 2, 3, ..., n-1 Relationen optimieren
- Vorgehen: Kostentabelle mit
  - k-elementiger Teilmenge $R \subseteq \{r1, r2, ... rn\}$
  - optimale Lösung (Verbundreihenfolge)
- Kosten
  - zur Kostenbestimmung benötigte Informationen (z.B. Größe der Zwischenergebnisse)

Algorithmus

```
Input: Verbundanfrage Q mit n Relationen r_1, ..., r_n
Output: Anfrageplan in R[{r_1, ..., r_n}]
for i := 1 to n do
begin
   R[r_i] := r_i
end
for i := 2 to n do
begin
   foreach s ⊆ r_1, ..., r_n mit |s| = i do
   begin
      R[s] := {}
      foreach r_k ∈ s do
      begin
         R[s] := R[s] ∪ (R[s-{r_k}] ⋈  r_k)
      end
   end
end
return R[{r_1, ..., r_n}]
```

Beispiel

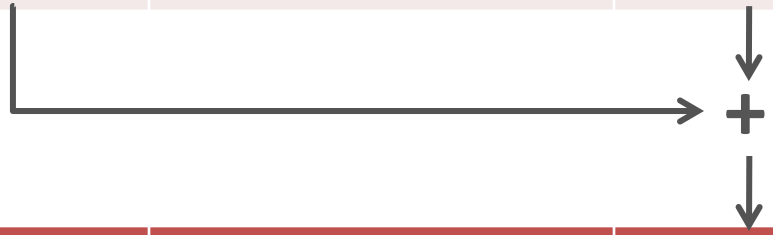
- Anfrage: r(Bestellung) ⋈ r(Produkt) ⋈ r(Kunde) ⋈ r(Lieferant)

	R	Ergebnisgröße	Plan
Schritt 1	{K}	1.000	r(Kunde)
	{P}	5.000	r(Produkt)
	{L}	100	r(Lieferant)
	{B}	20.000	r(Bestellung)
Schritt 2	{K,P}	5.000.000	r(Kunde) ⋈ r(Produkt)
	{K,L}	1.000.000	r(Kunde) ⋈ r(Lieferant)
	{K,B}	20.000	r(Kunde) ⋈ r(Bestellung)
	{P,L}	5.000	r(Produkt) ⋈ r(Lieferant)
	{P,B}	20.000	r(Produkt) ⋈ r(Bestellung)
	{L,B}	2.000.000	r(Lieferant) ⋈ r(Bestellung)

Beispiel – Fortsetzung

Ergebnisgröße aus Schritt 2

	R	Ergebnisgröße	Plan	Kosten
Schritt 3	{K,P,L}	5.000.000	(r(P) ⋈ r(L)) ⋈ r(K)	5.000
	{K,P,B}	20.000	(r(K) ⋈ r(B)) ⋈ r(P)	20.000
	{K,L,B}	2.000.000	(r(K) ⋈ r(B)) ⋈ r(L)	20.000
	{P,L,B}	20.000	(r(P) ⋈ r(L)) ⋈ r(B)	5.000



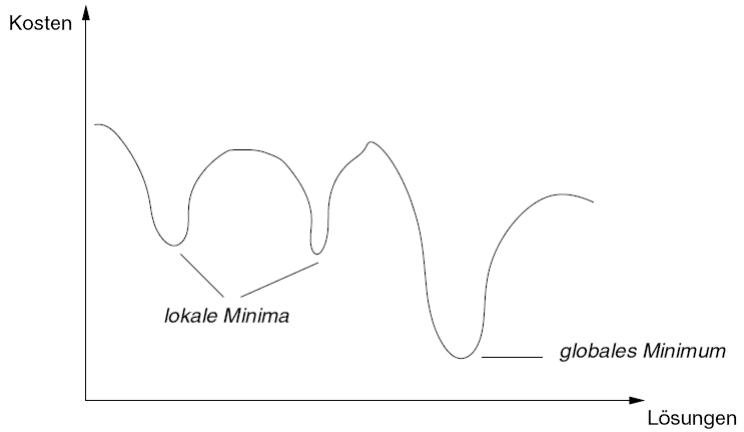
	R	Ergebnisgröße	Plan	Kosten	
Schritt 4	{K,P,L,B}		((r(P) ⋈ r(L)) ⋈ r(K)) ⋈ r(B)	5.005.000	
	{K,P,L,B}		((r(K) ⋈ r(B)) ⋈ r(P)) ⋈ r(L)	40.000	
	{K,P,L,B}		((r(K) ⋈ r(B)) ⋈ r(L)) ⋈ r(P)	2.020.000	
	{K,P,L,B}		((r(P) ⋈ r(L)) ⋈ r(B)) ⋈ r(K)	25.000	Bester Plan

Erweiterungen

- Berücksichtigung buschiger Bäume
- Berücksichtigung mehrerer alternativer Teilpläne
  - spezielle Verbundalgorithmen (z.B. Merge Join), Sortier- oder Gruppierungsoperationen
  - Teilpläne betrachten, die Tupel in potenziell geeigneter Reihenfolge produzieren
- Nutzung von Heuristiken zum Pruning (Abschneiden offensichtlich ungeeigneter Zweige) nach jeder Iteration
  - möglichst späte Berechnung von kartesischen Produkten
  - Beschränkung auf eine Baumform, z.B. links-orientierte Bäume

Ansatz

- Vermeiden des vollständigen Durchsuchens durch Betrachtung von ‚Nachbarplänen'
- Vermeiden des Hängenbleibens in lokalen Minimum durch Wahl zufälliger Nachbarn
- Transformation: Plan --> Nachbarplan
  - Wechsel des Verbundalgorithmus: r1 ⋈ r2 → r1 ⋈ r2
  - Kommutativität: r1 ⋈ r2 → r2 ⋈ r1
  - Assoziativität: (r1 ⋈ r2) ⋈ r3 → r1 ⋈ (r2 ⋈ r3)
  - Linkstausch: (r1 ⋈ r2) ⋈ r3 → (r1 ⋈ r3) ⋈ r2
  - Rechtstausch: r1 ⋈ (r2 ⋈ r3) → r2 ⋈ (r1 ⋈ r3)



- erster Ansatz: Hill Climbing
  - Initialisierung mit mehreren zufällig gewählten Startplänen
  - Abbruchbedingung: Zeitlimit oder festgelegte Anzahl von Startplänen
  - ohne Begrenzung: Wahrscheinlichkeit für Finden des globalen Minimums -> 1
- Verbesserung: Simulated Annealing
  - Simulation des Abkühlens von Metallen
  - mit gewisser Wahrscheinlichkeit auch zwischenzeitlich schlechte Lösungen akzeptieren
  - mit zunehmender Dauer (sinkender Temperatur) Annahmewahrscheinlichkeit für schlechte Lösungen reduzieren

Algorithmus

```
Input: Verbundanfrage Q
Output: Anfrageplan für Q
minCost := ∞
while not Abbruchbedingung do
begin
   newPlan := randomSolution(Q)
   while not lokalMinimum(newPlan) do
   begin
       neighbor := randomNeighbor(newPlan)
       if cost(neighbor) < cost(newPlan) then
          newPlan := neighbor
       endif
   end
   if cost(newPlan) < minCost then
      optPlan := newPlan; minCost := cost(optPlan)
   endif
end
return optPlan
```

Anfragerestrukturierung

- Standardisierung und Vereinfachung
- Auflösen geschachtelter Unteranfragen
- Restrukturierungsregeln
- Invariantes Gruppieren
- Frühzeitiges Vorgruppieren

Anfragetransformation

- Kostenbasierter Optimierer
- Kostenmodell
- Suchstrategien

Literatur

- Härder, T. & Rahm, E. Datenbanksysteme: Konzepte und Techniken der Implementierung. Springer-Verlag, 1999
- Saake, G.; Heuer, A. & Sattler, K.-U. Datenbanken: Implementierungstechniken. MITP-Verlag, 2005
- Hellerstein, J. M.; Stonebraker, M. & Hamilton, J. R. Architecture of a Database System. Foundations and Trends in Databases, 2007, 1, 141-259
- Stillger, M. et al. LEO - DB2's LEarning Optimizer. VLDB 2001, Morgan Kaufmann, 2001, 19-28
- Graefe, G. Query Evaluation Techniques for Large Databases. ACM Computing Surveys, 1993, 25, 73-170
- Jarke, M. & Koch, J. Query Optimization in Database Systems. ACM Computing Surveys, 1984, 16, 111-152
- Selinger, P. G. et al. Access Path Selection in a Relational Database Management System. SIGMOD 1979, ACM, 1979, 23-34
- Astrahan, M. M. et al. System R: Relational Approach to Database Management. ACM Transactions on Database Systems, 1976, 1, 97-137