



Skript zur Vorlesung  
**Datenbanksysteme I**  
Wintersemester 2012/2013

# Kapitel 10: Relationale Anfragebearbeitung

Vorlesung: Prof. Dr. Christian Böhm  
Übungen: Sebastian Goebel, Nina Hubig

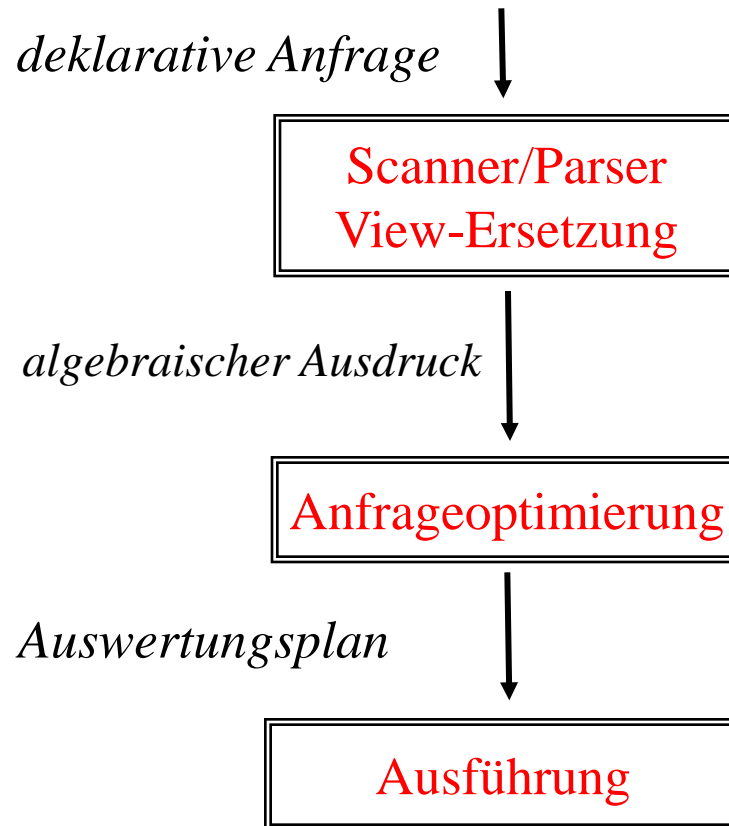
Skript © 2004 Christian Böhm

<http://www.dbs.informatik.uni-muenchen.de/Lehre/DBS>



# Relationale Anfragebearbeitung

Zentrale Aufgabe der Anfragebearbeitung ist die Übersetzung der *deklarativen* Anfrage in einen *effizienten, prozeduralen* Auswertungsplan





# Relationale Anfragebearbeitung

Kanonischer Auswertungsplan zu einer SQL-Anfrage  
(Ergebnis der ersten Übersetzungsphase)

**select**  $A_1, A_2, \dots$

**from**  $R_1, R_2, \dots$

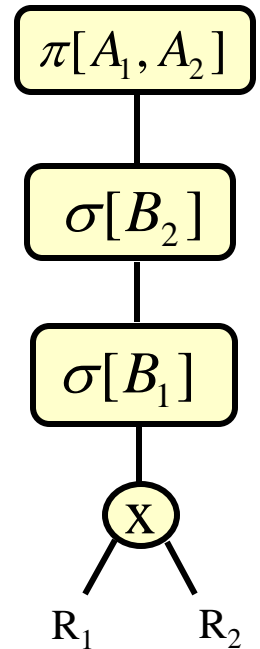
**where**  $B_1$  **and**  $B_2,$

$\dots$

1. Bilde das kartesische Produkt der Relationen  $R_1, R_2, \dots$
2. Führe Selektionen mit den einzelnen Bedingungen  $B_1, B_2, \dots$  durch.
3. Projiziere die Ergebnistupel auf die erforderlichen Attribute  $A_1, A_2,$

$\dots$

$$\pi_{A_1, A_2} (\sigma_{B_2} (\sigma_{B_1} (R_1 \times R_2)))$$





# Relationale Anfragebearbeitung

## Beispiel Autodatenbank

Kunden(KNr, Name, Adresse, Region, Saldo)

| KNr | Name   | Adresse    | Region       | Saldo   |
|-----|--------|------------|--------------|---------|
| 201 | Klein  | Lilienthal | Bremen       | 200.000 |
| 337 | Horn   | Dieburg    | Rhein-Main   | 100.000 |
| 444 | Berger | München    | München      | 300.000 |
| 108 | Weiss  | Würzburg   | Unterfranken | 500.000 |

View GuteKunden(KNr, Name, Adresse, Region, Saldo) =

**select \* from Kunden where Saldo  $\geq$  300.000**

Bestellt(BNr, Datum, KNr, Region, Saldo) Produkt(PNr, Bezeichnung, Anzahl, Preis)

| BNr | Datum    | KNr | PNr |
|-----|----------|-----|-----|
| 221 | 10.05.04 | 201 | 12  |
| 312 | 11.05.04 | 201 | 4   |
| 401 | 20.05.04 | 337 | 330 |
| 456 | 13.05.04 | 444 | 330 |
| 458 | 14.05.04 | 444 | 98  |

| PNr | Bezeichnung | Anzahl | Preis   |
|-----|-------------|--------|---------|
| 12  | BMW 318i    | 10     | 40.000  |
| 4   | Golf 5      | 40     | 25.000  |
| 330 | Fiat Uno    | 5      | 18.000  |
| 98  | Ferrari 380 | 1      | 180.000 |
| 14  | Opel Corsa  | 14     | 17.000  |



# Relationale Anfragebearbeitung

Einfache SQL-Anfrage:

Welche guten Kunden (Name) haben einen Fiat Uno bestellt  
(und Saldo  $\geq$  300.000) ?

```
select  Name
from    GuteKunden k, Bestellt b, Produkt p
where   b.KNr = k.KNr
and     b.PNr = p.PNr
and     Bezeichnung = ,Fiat Uno‘
```

**Expansion der View:**

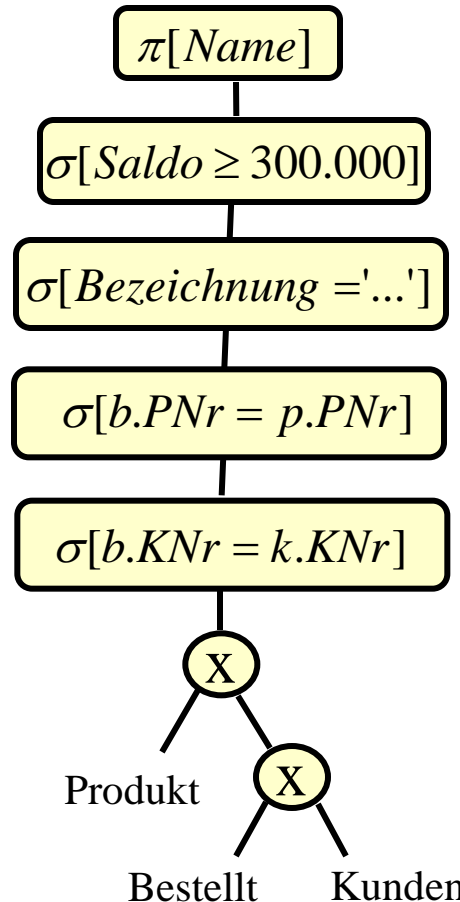
```
select  Name
from    Kunden k, Bestellt b, Produkt p
where   b.KNr = k.KNr
and     b.PNr = p.PNr
and     Bezeichnung = ,Fiat Uno‘
and     Saldo  $\geq$  300.000
```



# Relationale Anfragebearbeitung

Übersetzung in relationale Algebra (kanonisch):

$$\pi_{Name} (\sigma_{Saldo \geq 30000} (\sigma_{Bezeichnung='FiatUno'} (\sigma_{b.PNr=p.PNr} (\sigma_{b.KNr=k.KNr} (Produkt \times (Bestellt \times Kunden))))))$$





# Relationale Anfragebearbeitung

Kunden:

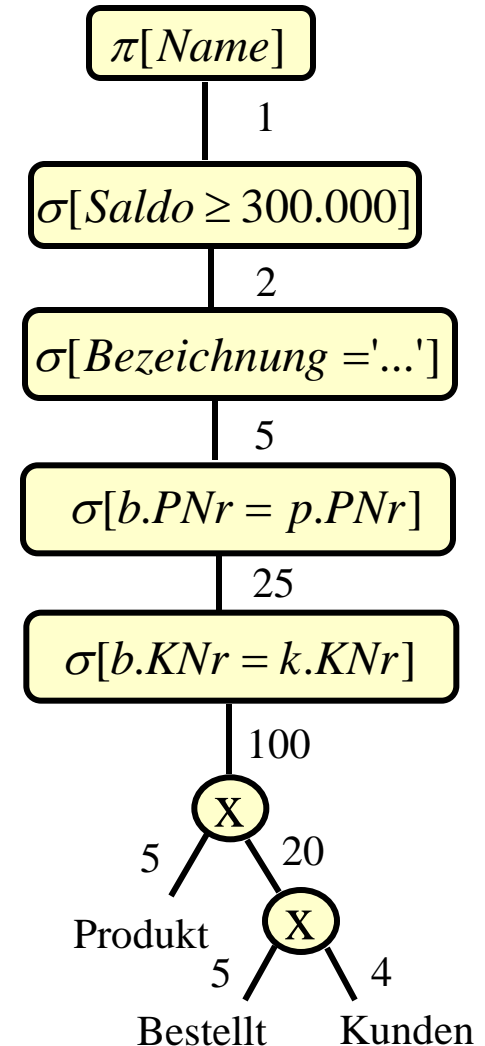
| KNr | Name   | Adresse    | Region       | Saldo   |
|-----|--------|------------|--------------|---------|
| 201 | Klein  | Lilienthal | Bremen       | 200.000 |
| 337 | Horn   | Dieburg    | Rhein-Main   | 100.000 |
| 444 | Berger | München    | München      | 300.000 |
| 108 | Weiss  | Würzburg   | Unterfranken | 500.000 |

Bestellt:

| BNr | Datum    | KNr | PNr |
|-----|----------|-----|-----|
| 221 | 10.05.04 | 201 | 12  |
| 312 | 11.05.04 | 201 | 4   |
| 401 | 20.05.04 | 337 | 330 |
| 456 | 13.05.04 | 444 | 330 |
| 458 | 14.05.04 | 444 | 98  |

Produkt:

| PNr | Bezeichnung | Anzahl | Preis   |
|-----|-------------|--------|---------|
| 12  | BMW 318i    | 10     | 40.000  |
| 4   | Golf 5      | 40     | 25.000  |
| 330 | Fiat Uno    | 5      | 18.000  |
| 98  | Ferrari 380 | 1      | 180.000 |
| 14  | Opel Corsa  | 14     | 17.000  |





# Relationale Anfragebearbeitung

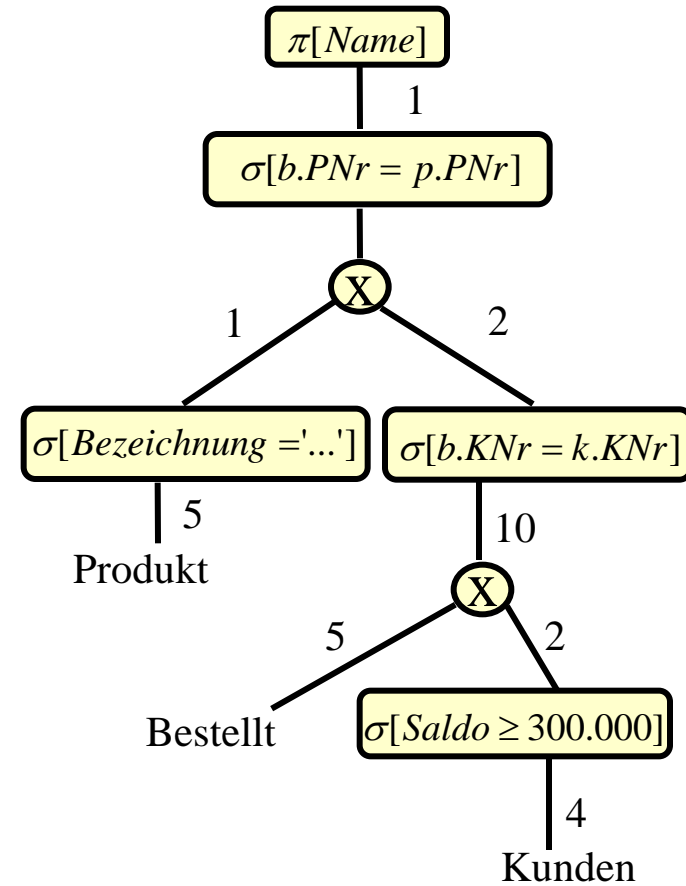
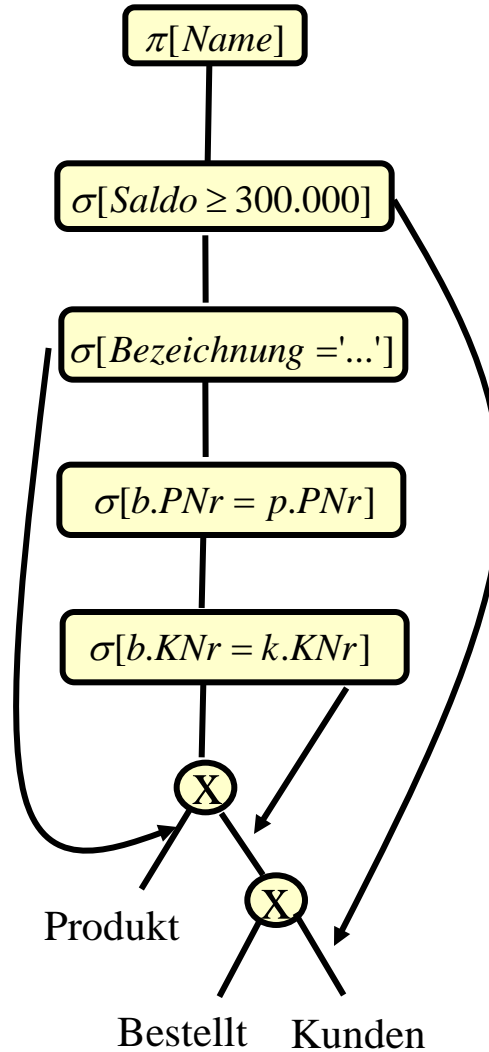
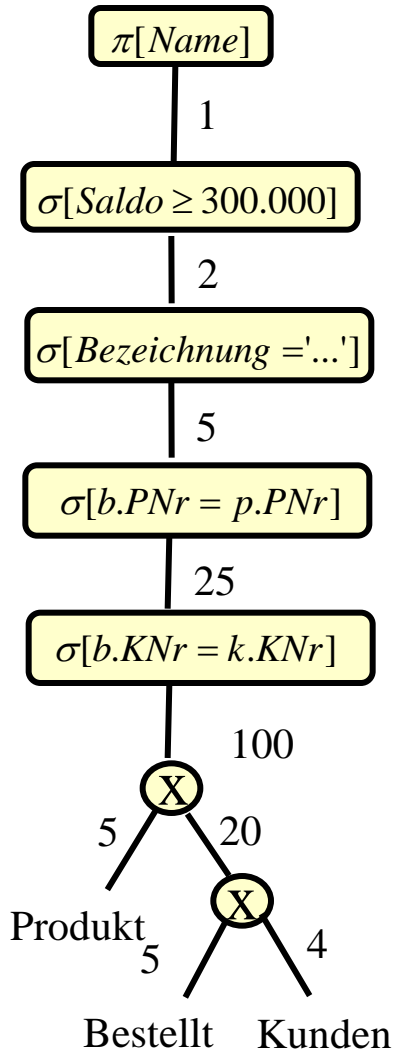
Beobachtungen:

- Der kanonische Auswertungsplan erzeugt das kartesische Produkt der 3 Relationen
- Die Kardinalität des kartesischen Produkts ist  $|\text{Kunden}| * |\text{Bestellt}| * |\text{Produkt}| = 100$  Tupel
- Für jedes der 100 Tupel muss z.B. die Bedingung  $b.KNr=k.KNr$  ausgewertet werden
- Günstiger wäre es z.B., wenn man sich gleich von Anfang an auf das Produkt ‚Fiat Uno‘ und die Kunden mit hohem Saldo beschränken würde:





# Relationale Anfragebearbeitung





# Relationale Anfragebearbeitung

- i.A. gibt es viele verschiedene, *gleichwertige* Auswertungspläne für dieselbe Anfrage
- Die Performanz gleichwertiger Auswertungspläne variiert häufig zwischen wenigen Sekunden (schnellster Plan) und vielen Stunden (Standardplan)
- Die Aufgabe der Anfrageoptimierung ist es, den günstigsten Auswertungsplan zu ermitteln (bzw. zumindest einen sehr günstigen Plan zu ermitteln)
- Wegen des großen Unterschiedes zwischen günstigstem und ungünstigstem Plan ist die Optimierung bei der relationalen Anfragebearbeitung wesentlich wichtiger als z.B. bei der Übersetzung von (imperativen) Programmiersprachen



# Relationale Anfragebearbeitung

Logische und physische Anfrageoptimierung:

- Optimierungstechniken, die den Auswertungsplan betrachten und „umbauen“ werden als logische Anfrageoptimierung bezeichnet
- Physische Anfrageoptimierung: Auswahl einer geeigneten Auswertungsstrategie für Join-Operationen oder Entscheidung, ob für eine Selektionsoperation ein Index verwendet wird.

Beispiel: Auswertungsstrategien für Joins

- Erzeuge alle Tupel des kartesischen Produkts und prüfe Join-Bedingung (Nested Loop)
- Sortiere beide Relationen nach dem Joinattribut und filtere passende Paare (Sort Merge)
- Betrachte alle Tupel der einen Relation und greife auf die Joinpartner über einen passenden Index der anderen Relation zu (Indexed Loop)



# Relationale Anfragebearbeitung

## Regel- und kostenbasierte Optimierung

- Es gibt zahlreiche Regeln (Heuristiken), um die Reihenfolge der Operatoren im Auswertungsplan zu modifizieren und so eine Performanz-Verbesserung zu erreichen, z.B. Push Selection: Führe Selektionen möglichst frühzeitig (vor Joins) aus
- Optimierer, die sich ausschließlich nach solchen starren Regeln richten, nennt man regelbasierte oder auch algebraische Optimierer



# Relationale Anfragebearbeitung

- Optimierer, die die voraussichtliche Performanz von Auswertungsplänen ermitteln, werden als **kostenbasierte Optimierer** bezeichnet.  
Die Vorgehensweise ist meist folgende:
  1. Generiere einen initialen Plan (z.B. Standardauswertungsplan)
  2. Schätze bei der Auswertung entstehende Kosten
  3. Modifiziere den aktuellen Plan gemäß vorgegebener Heuristiken
  4. Wiederhole die Schritte 2 und 3 bis ein Stop-Kriterium erreicht ist
  5. Gib den besten erhaltenen Plan aus

Als Kostenmaß eignen sich der Erwartungswert der Antwortzeit (Einbenutzerbetrieb) oder die Belegung von Ressourcen wie z.B. Anzahl zugegriffener Blöcke oder CPU-Nutzung (Durchsatz-Optimierung v.a. im Mehrbenutzerbetrieb)