

# Anfragebearbeitung

- Logische Optimierung
- Physische Optimierung
- (Kostenmodelle
- „Tuning“ )

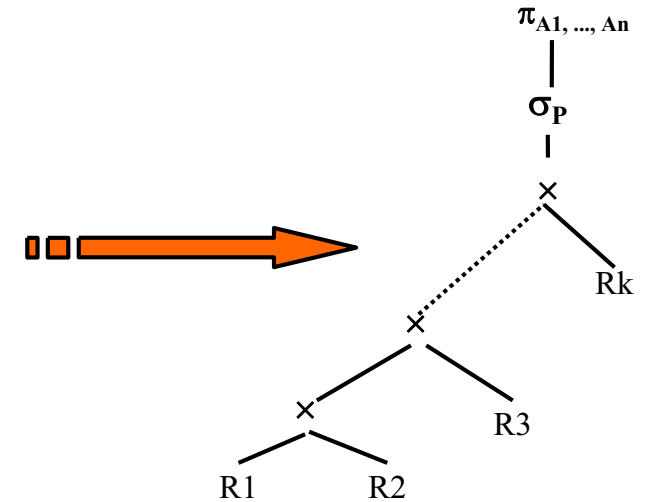


Kapitel 8

1

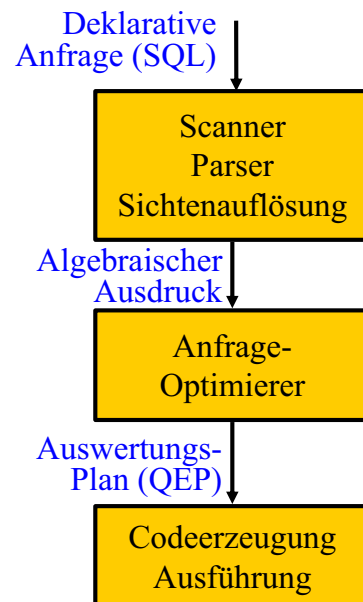
## Kanonische Übersetzung

select A1, ..., An  
from R1, ..., Rk  
where P



3

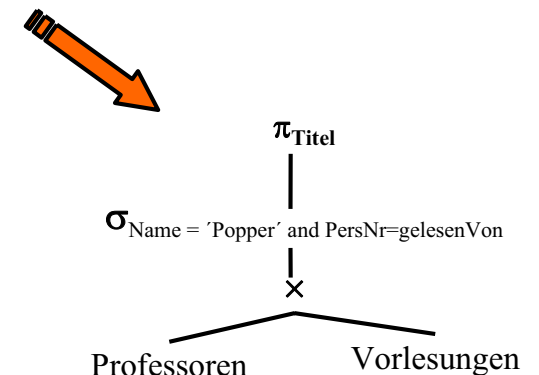
## Ablauf der Anfrageoptimierung



2

## Kanonische Übersetzung

select Titel  
from Professoren, Vorlesungen  
where Name = 'Popper' and  
PersNr = gelesenVon

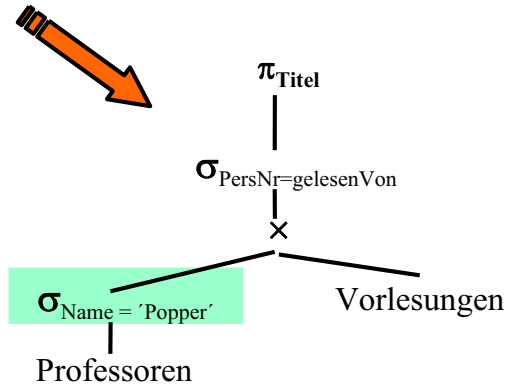


$\pi_{\text{Titel}}(\sigma_{\text{Name} = \text{'Popper' and PersNr=gelesenVon}}(\text{Professoren} \times \text{Vorlesungen}))$

4

# Erste Optimierungsidee

```
select Titel
from Professoren, Vorlesungen
where Name = 'Popper' and
      PersNr = gelesenVon
```



$\pi_{\text{Titel}}(\sigma_{\text{PersNr=gelesenVon}}((\sigma_{\text{Name='Popper'}} \text{Professoren}) \times \text{Vorlesungen}))$

## Optimierung von Datenbank-Anfragen

### Grundsätze:

- Sehr hohes Abstraktionsniveau der mengenorientierten Schnittstelle (SQL).
- Sie ist **deklarativ**, **nicht-prozedural**, d.h. es wird spezifiziert, **was** man finden möchte, aber nicht **wie**.
- Das **wie** bestimmt sich aus der Abbildung der mengenorientierten Operatoren auf Schnittstellen-Operatoren der internen Ebene (Zugriff auf Datensätze in Dateien, Einfügen/Entfernen interner Datensätze, Modifizieren interner Datensätze).
- Zu einem **was** kann es zahlreiche **wie**'s geben: effiziente Anfrageauswertung durch Anfrageoptimierung.
- i.Allg. wird aber nicht die optimale Auswertungsstrategie gesucht (bzw. gefunden) sondern eine einigermaßen effiziente Variante
  - Ziel: „avoiding the worst case“

# Äquivalenzerhaltende Transformationsregeln

1. Aufbrechen von Konjunktionen im Selektionsprädikat

$$\sigma_{c_1 \wedge c_2 \wedge \dots \wedge c_n}(R) \equiv \sigma_{c_1}(\sigma_{c_2}(\dots(\sigma_{c_n}(R)) \dots))$$

2.  $\sigma$  ist kommutativ

$$\sigma_{c_1}(\sigma_{c_2}(R)) \equiv \sigma_{c_2}(\sigma_{c_1}(R))$$

3.  $\pi$ -Kaskaden: Falls  $L_1 \subseteq L_2 \subseteq \dots \subseteq L_n$  dann gilt

$$\pi_{L_1}(\pi_{L_2}(\dots(\pi_{L_n}(R)) \dots)) \equiv \pi_{L_1}(R)$$

4. Vertauschen von  $\sigma$  und  $\pi$

Falls die Selektion sich nur auf die Attribute  $A_1, \dots, A_n$  der Projektionsliste bezieht, können die beiden Operationen vertauscht werden

$$\pi_{A_1, \dots, A_n}(\sigma_c(R)) \equiv \sigma_c(\pi_{A_1, \dots, A_n}(R))$$

5.  $\times, \cup, \cap$  und  $\bowtie$  sind kommutativ

$$R \bowtie_c S \equiv S \bowtie_c R$$

# Äquivalenzerhaltende Transformationsregeln

6. Vertauschen von  $\sigma$  mit  $\bowtie$

Falls das Selektionsprädikat  $c$  nur auf Attribute der Relation  $R$  zugreift, kann man die beiden Operationen vertauschen:

$$\sigma_c(R \bowtie_j S) \equiv \sigma_c(R) \bowtie_j S$$

Falls das Selektionsprädikat  $c$  eine Konjunktion der Form „ $c_1 \wedge c_2$ “ ist und  $c_1$  sich nur auf Attribute aus  $R$  und  $c_2$  sich nur auf Attribute aus  $S$  bezieht, gilt folgende Äquivalenz:

$$\sigma_c(R \bowtie_j S) \equiv \sigma_{c_1}(R) \bowtie_j \sigma_{c_2}(S)$$

## Äquivalenzerhaltende Transformationsregeln

### 7. Vertauschung von $\pi$ mit $\bowtie$

Die Projektionsliste  $L$  sei:  $L = \{A_1, \dots, A_n, B_1, \dots, B_m\}$ , wobei  $A_i$  Attribute aus  $R$  und  $B_j$  Attribute aus  $S$  seien. Falls sich das Joinprädikat  $c$  nur auf Attribute aus  $L$  bezieht, gilt folgende Umformung:

$$\pi_L(R \bowtie_c S) \equiv (\pi_{A_1, \dots, A_n}(R)) \bowtie_c (\pi_{B_1, \dots, B_m}(S))$$

Falls das Joinprädikat sich auf weitere Attribute, sagen wir  $A_1', \dots, A_p'$  aus  $R$  und  $B_1', \dots, B_q'$  aus  $S$  bezieht, müssen diese für die Join-Operation erhalten bleiben und können erst danach herausprojiziert werden:

$$\pi_L(R \bowtie_c S) \equiv \pi_L(\pi_{A_1, \dots, A_n, A_1', \dots, A_p'}(R) \bowtie_c \pi_{B_1, \dots, B_m, B_1', \dots, B_q'}(S))$$

Für die  $\times$ -Operation gibt es kein Prädikat, so dass die Einschränkung entfällt.

9

## Äquivalenzerhaltende Transformationsregeln

8. Die Operationen  $\bowtie$ ,  $\times$ ,  $\cup$ ,  $\cap$  sind jeweils (einzeln betrachtet) assoziativ. Wenn also  $\Phi$  eine dieser Operationen bezeichnet, so gilt:

$$(R \Phi S) \Phi T \equiv R \Phi (S \Phi T)$$

9. Die Operation  $\sigma$  ist distributiv mit  $\cup$ ,  $\cap$ ,  $-$ . Falls  $\Phi$  eine dieser Operationen bezeichnet, gilt:

$$\sigma_c(R \Phi S) \equiv (\sigma_c(R) \Phi \sigma_c(S))$$

10. Die Operation  $\pi$  ist distributiv mit  $\cup$ .

$$\pi_c(R \cup S) \equiv (\pi_c(R) \cup \pi_c(S))$$

10

## Äquivalenzerhaltende Transformationsregeln

11. Die Join- und/oder Selektionsprädikate können mittels der Regeln von De Morgan umgeformt werden:

$$\neg (c_1 \wedge c_2) \equiv (\neg c_1) \vee (\neg c_2)$$

$$\neg (c_1 \vee c_2) \equiv (\neg c_1) \wedge (\neg c_2)$$

12. Ein kartesisches Produkt, das von einer Selektions-Operation gefolgt wird, deren Selektionsprädikat Attribute aus beiden Operanden des kartesischen Produktes enthält, kann in eine Joinoperation umgeformt werden.

Sei  $c$  eine Bedingung der Form  $A \theta B$ , mit  $A$  ein Attribut von  $R$  und  $B$  ein Attribut aus  $S$ .

$$\sigma_c(R \times S) \equiv R \bowtie_c S$$

11

## Heuristische Anwendung der Transformationsregeln

- Mittels Regel 1 werden konjunktive Selektionsprädikate in Kaskaden von  $\sigma$ -Operationen zerlegt.
- Mittels Regeln 2, 4, 6, und 9 werden Selektionsoperationen soweit „nach unten“ propagiert wie möglich.
- Mittels Regel 8 werden die Blattknoten so vertauscht, dass derjenige, der das kleinste Zwischenergebnis liefert, zuerst ausgewertet wird.
- Forme eine  $\times$ -Operation, die von einer  $\sigma$ -Operation gefolgt wird, wenn möglich in eine  $\bowtie$ -Operation um
- Mittels Regeln 3, 4, 7, und 10 werden Projektionen soweit wie möglich nach unten propagiert.
- Versuche Operationsfolgen zusammenzufassen, wenn sie in einem „Durchlauf“ ausführbar sind (z.B. Anwendung von Regel 1, Regel 3, aber auch Zusammenfassung aufeinanderfolgender Selektionen und Projektionen zu einer „Filter“-Operation).

12

## Anwendung der Transformationsregeln

select distinct s.Semester

from Studenten s, hören h

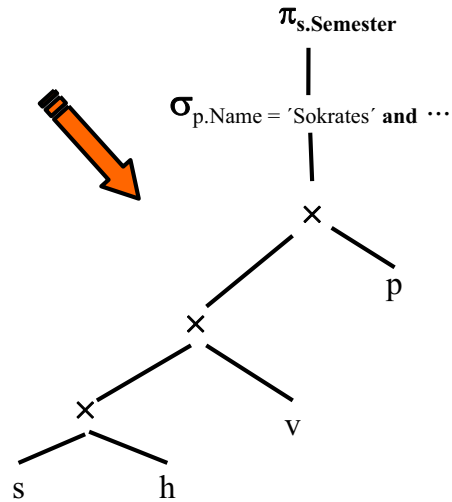
Vorlesungen v, Professoren p

where p.Name = 'Sokrates' and

v.gelesenVon = p.PersNr and

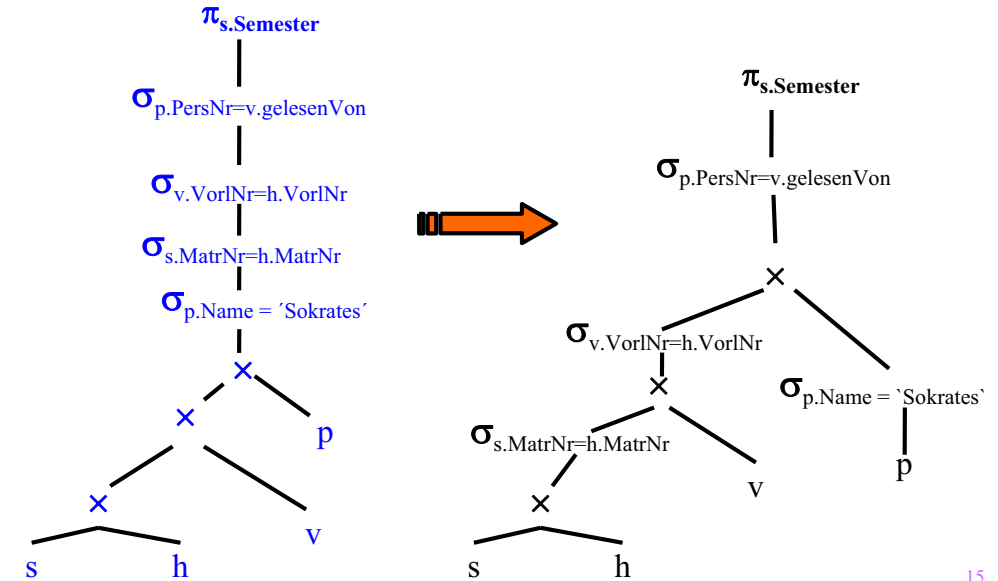
v.VorlNr = h.VorlNr and

h.MatrNr = s.MatrNr



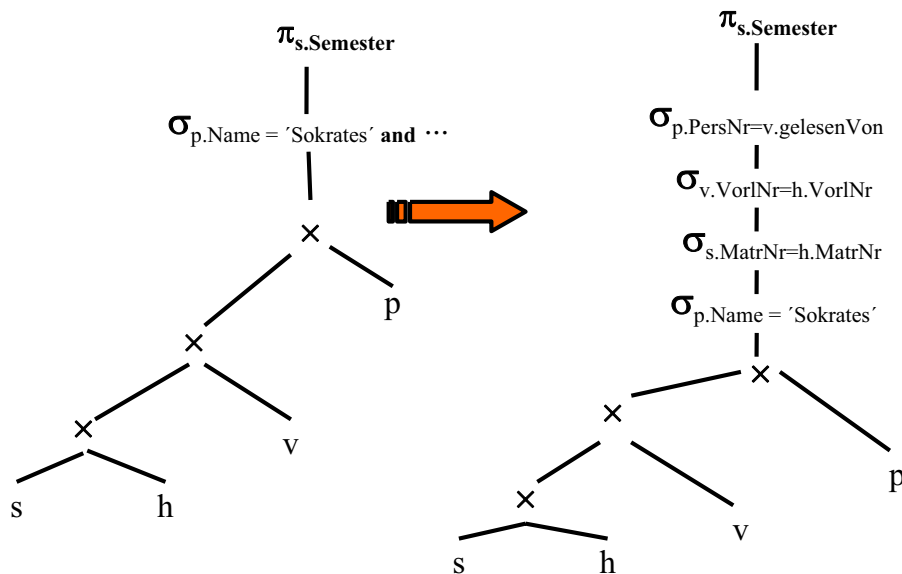
13

## Verschieben der Selektionsprädikate „Pushing Selections“



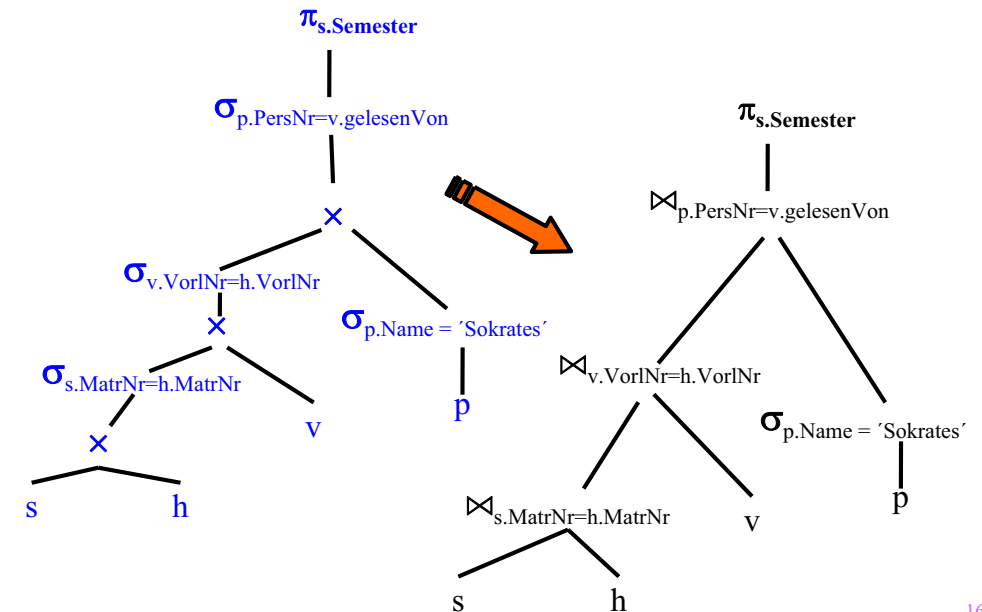
15

## Aufspalten der Selektionsprädikate



14

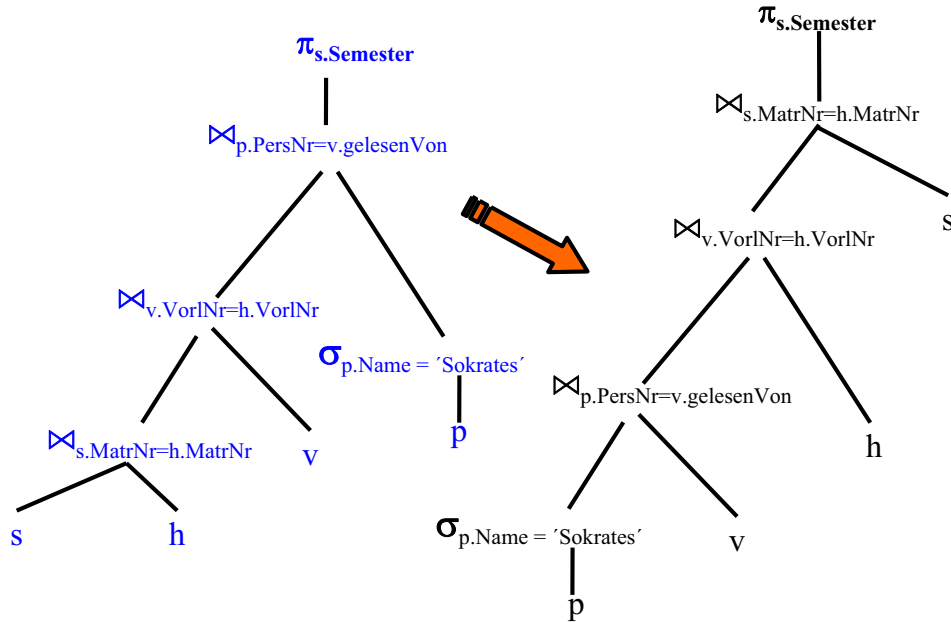
## Zusammenfassung von Selektionen und Kreuzprodukten zu Joins



16

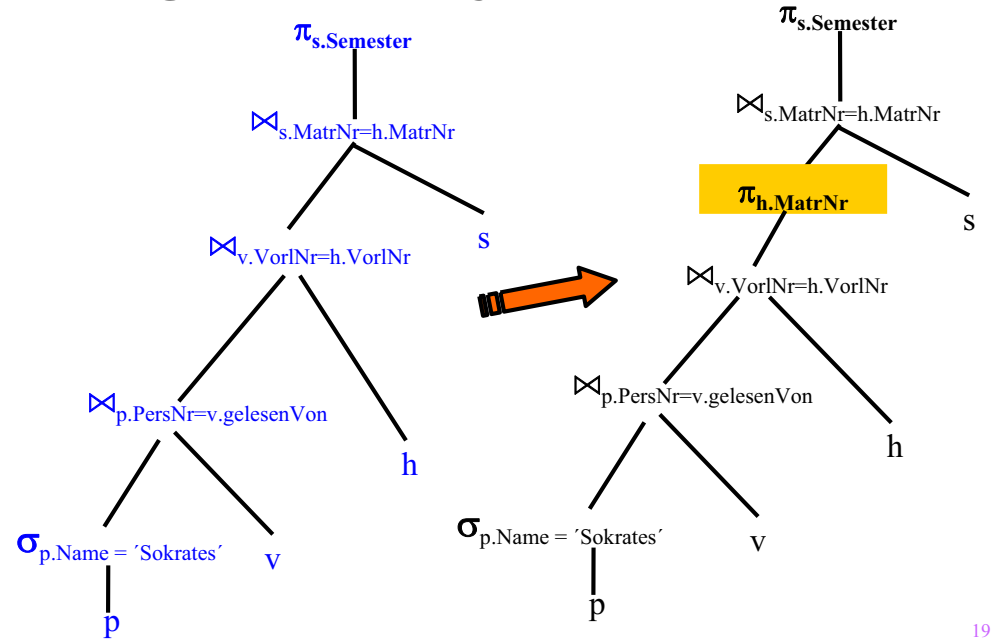
# Optimierung der Joinreihenfolge

## Kommutativität und Assoziativität ausnutzen



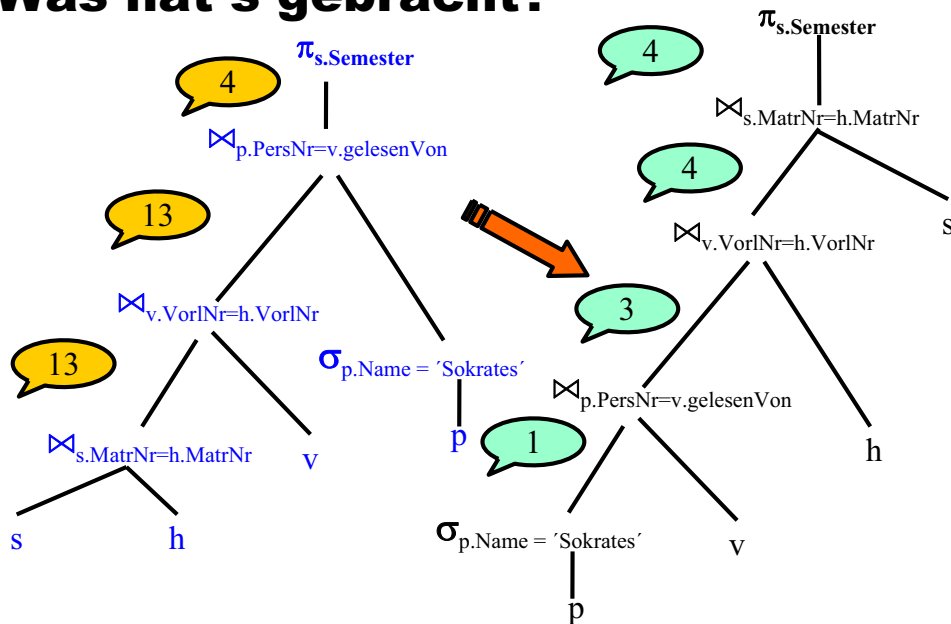
17

# Einfügen von Projektionen



19

# Was hat's gebracht?



18

# Der natürliche Verbund zweier Relationen $R$ und $S$

$R$			$S$			$R \bowtie S$				
$A$	$B$	$C$	$C$	$D$	$E$	$A$	$B$	$C$	$D$	$E$
$a_1$	$b_1$	$c_1$	$c_1$	$d_1$	$e_1$	$a_1$	$b_1$	$c_1$	$d_1$	$e_1$
$a_2$	$b_2$	$c_2$	$c_3$	$d_2$	$e_2$	$a_3$	$b_3$	$c_1$	$d_1$	$e_1$
$a_3$	$b_3$	$c_1$	$c_4$	$d_3$	$e_3$	$a_5$	$b_5$	$c_3$	$d_2$	$e_2$
$a_4$	$b_4$	$c_2$	$c_5$	$d_4$	$e_4$					
$a_5$	$b_5$	$c_3$	$c_7$	$d_5$	$e_5$					
$a_6$	$b_6$	$c_2$	$c_8$	$d_6$	$e_6$					
$a_7$	$b_7$	$c_6$	$c_5$	$d_7$	$e_7$					

20

# Implementierung der Verbindung: Strategien

## J1 nested (inner-outer) loop

- „brute force“-Algorithmus

```
foreach r ∈ R
  foreach s ∈ S
    if s.B = r.A then Res := Res ∪ (r ∘ s)
```

21

## iterator NestedLoop<sub>p</sub>

### open

- Öffne die linke Eingabe

### next

- Rechte Eingabe geschlossen?
  - Öffne sie
- Fordere rechts solange Tupel an, bis Bedingung  $p$  erfüllt ist
- Sollte zwischendurch rechte Eingabe erschöpft sein
  - Schließe rechte Eingabe
  - Fordere nächstes Tupel der linken Eingabe an
  - Starte **next** neu
- Gib den Verbund von aktuellem linken und aktuellem rechte Tupel zurück

### close

- Schließe beide Eingabequellen

# Implementierung der Verbindung: Strategien

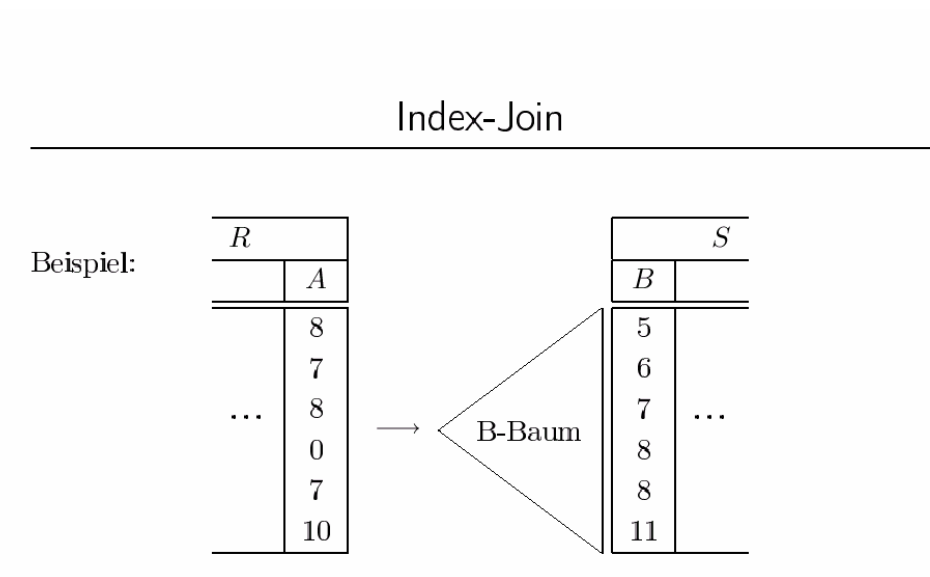
## J2 Zugriffsstruktur auf $S$

### Index Nested Loop Join

- in jedem Durchlauf von  $R$  werden nur die in  $S$  qualifizierenden Tupel gelesen
- dazu ist ein Index auf  $B$  erforderlich

```
foreach r ∈ R
  foreach s ∈ S[B=r.A]
    Res := Res ∪ (r ∘ s)
```

23



24

# Implementierung der Verbindung: Strategien

## J3 Sort-Merge Join

- erfordert zwei Sortierungen
  - $R$  muss nach  $A$  und
  - $S$  nach  $B$  sortiert sein
- sehr effizient
- falls  $A$  oder  $B$  Schlüsselattribut ist, wird jedes Tupel in  $R$  und  $S$  nur genau einmal gelesen

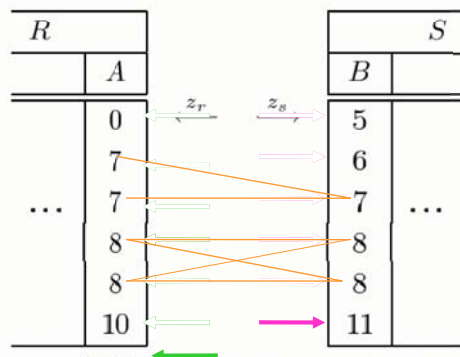
$R$		$S$		Ergebnis:	
...	A	B	...	...A	B...
...	5	4	...	...5	5...
...	5	4	...	...5	5...
...	5	4	...	...5	5...
...	5	5	...	...5	5...
...	6	5	...	...5	5...
...	6	6	...	...5	5...
...	6	7	...	...6	6...
...	7	7	...	...6	6...
...	7	7	...	...6	6...
...	7	8	...	...7	7...

25

Der Merge-Join

- Voraussetzung:  $R$  und  $S$  sind sortiert (notfalls vorher sortieren)

Beispiel:



26

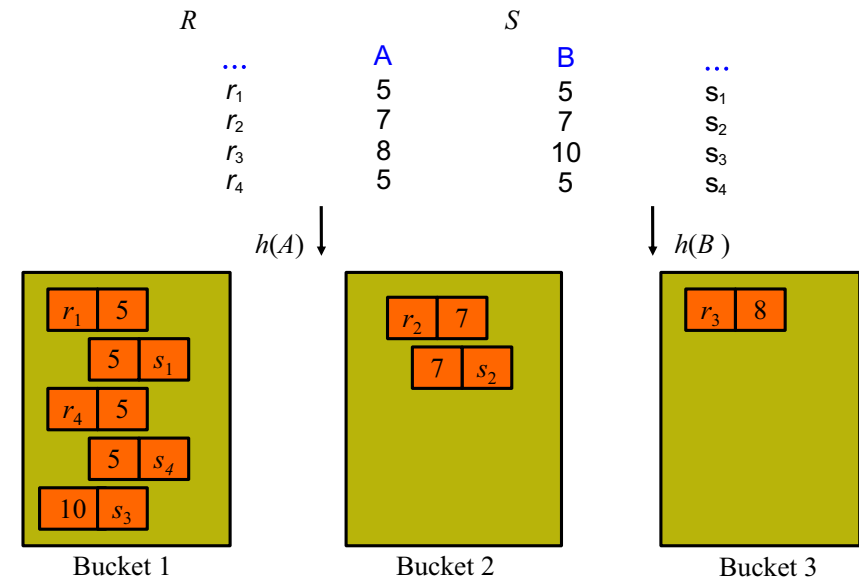
# Implementierung der Verbindung: Strategien

## J4 Hash-Join

- $R$  und  $S$  werden mittels der gleichen Hashfunktion  $h$  – angewendet auf  $R.A$  und  $S.B$  – auf (dieselben) Hash-Buckets abgebildet
- Hash-Buckets sind i.Allg. auf Hintergrundspeicher (abhängig von der Größe der Relationen)
- Zu verbindende Tupel befinden sich dann im selben Bucket
- Wird (nach praktischen Tests) nur von J3 „geschlagen“, wenn die Relationen schon vorsortiert sind

27

# Implementierung der Verbindung: Strategien



28

# Anfragebearbeitung

---

- Logische Optimierung
- Physische Optimierung
- (Kostenmodelle
- „Tuning“ )



**Kapitel 8**