

Anfrageoptimierung in verteilten Datenbanken

Seminar „Informationsverwaltung
in Sensornetzen“

Markus Drescher, WS 2006 / 2007

Szenario: verteilte Datenbank

Teilnehmer

<u>MatrNr</u>	<u>Thema</u>	<u>Betreuer</u>
1112222	Nutzung von Indizes auf Sensornetzen	Markus Bestehorn
1212121	Sensorknoten-Hardware und -Software	Markus Bestehorn
1223344	Anfrageprozessoren für Sensornetze	Mirco Stern

Mitarbeiter

<u>Name</u>	<u>Geb.</u>	<u>Institut</u>
Mirco Stern	01.04.76	IPD
Markus Bestehorn	02.04.76	IPD
Erik Buchmann	03.04.76	IPD

Ort: IPD



Student

<u>MatrNr</u>	<u>Sname</u>	<u>Fach</u>
1112222	Marc Philipp	Informatik
1212121	Niko Huber	Informatik
1223344	Mathias Reisch	Informatik
1234567	Wiwi Wichtig	Wi-Ing

Ort: Studienbüro

Anfrage



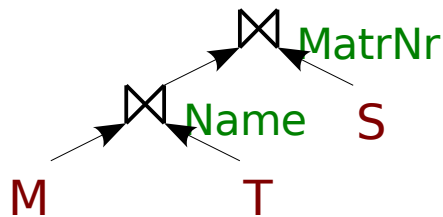
Ort: zu Hause

Welche Informatiker nehmen an einem Seminar an welchem Institut teil?

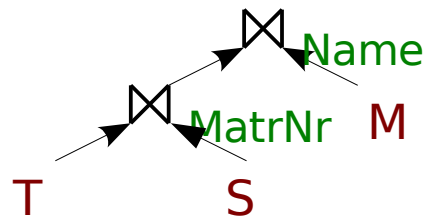
Anfrageformulierung

- **SELECT** Sname, Institut
FROM Mitarbeiter M, Teilnehmer T, Student S
WHERE M.Name = T.Betreuer
AND T.MatrNr = S.MatrNr

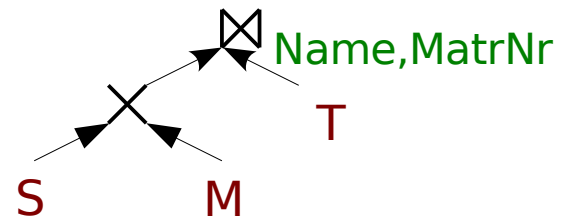
- Äquivalente Joinbäume:



(a)



(b)



(c)

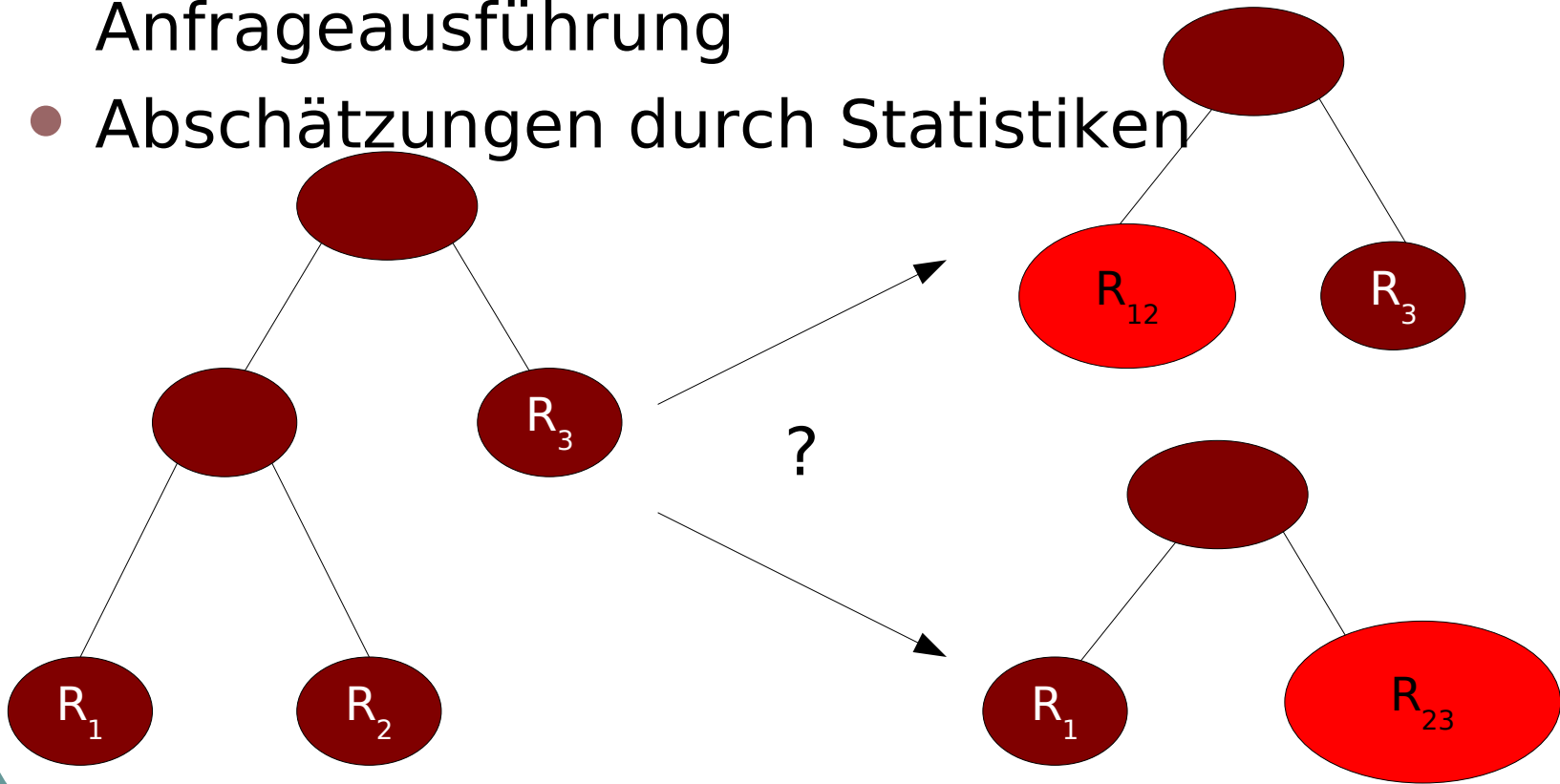
- für n Relationen: #Bäume $\in O(n!)$

Überblick

- 1) Motivation
- 2) Anfrageoptimierung in INGRES**
- 3) Anfrageoptimierung in System R
- 4) Kritik
- 5) Vergleich der Algorithmen

Optimierungsziel

- Minimierung der Zwischenergebnisse bei Anfrageausführung
- Abschätzungen durch Statistiken

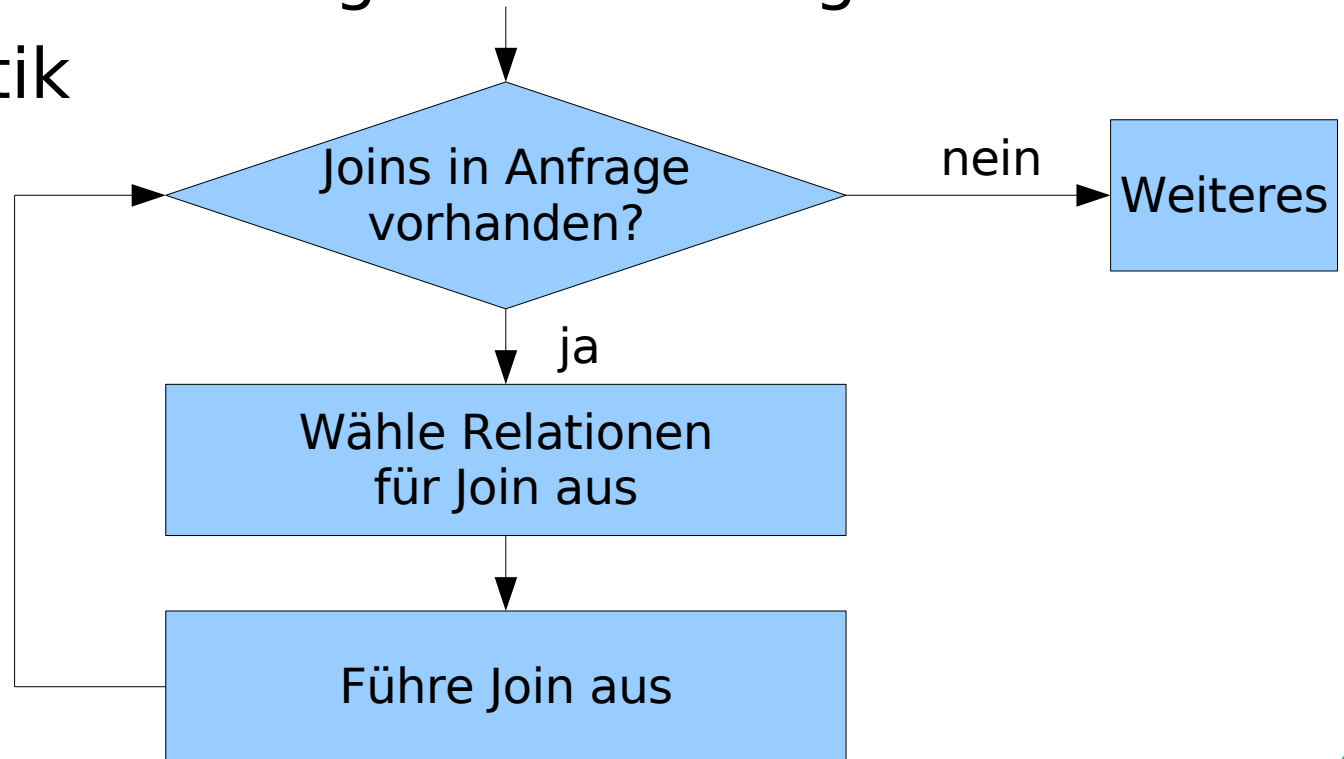


Kosten

- Zur Abschätzung werden Kostenfunktionen verwendet
- Komponenten einer Kostenfunktion
 - CPU-Zeit
 - I/O
 - Kosten für Kommunikation (Kardinalitäten der Relationen, Selektivitätsfaktoren)

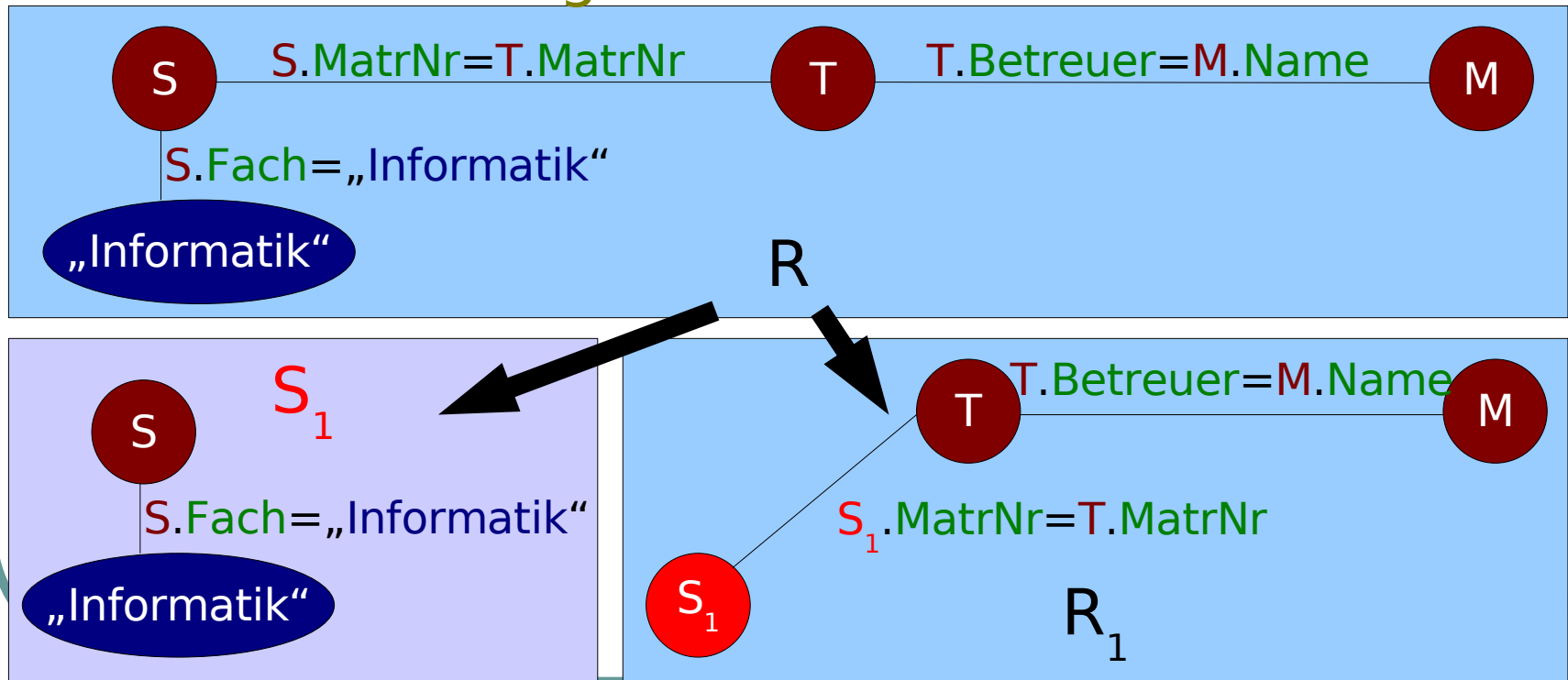
INGRES-Alg.: Zentraler Fall

- Dynamischer Algorithmus: Optimierung zur Laufzeit – verlängert die Anfragezeit
- Heuristik



Detachment I

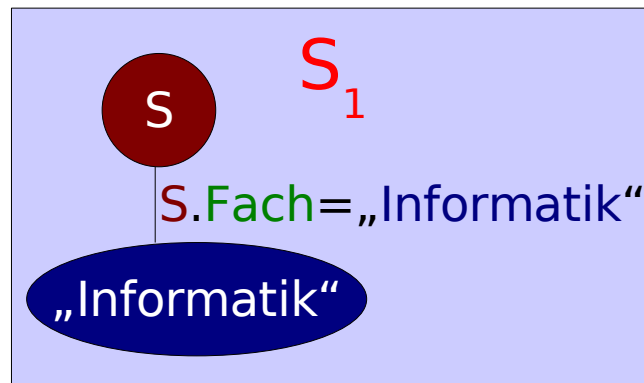
1) Zerlege Anfrage mit vielen Relationen in m Ein-Relation-Anfragen und eine Multi-Relation-Anfrage.



Monorelationale Anfragen

2) Führe jede **Ein-Relation-Anfrage** aus.

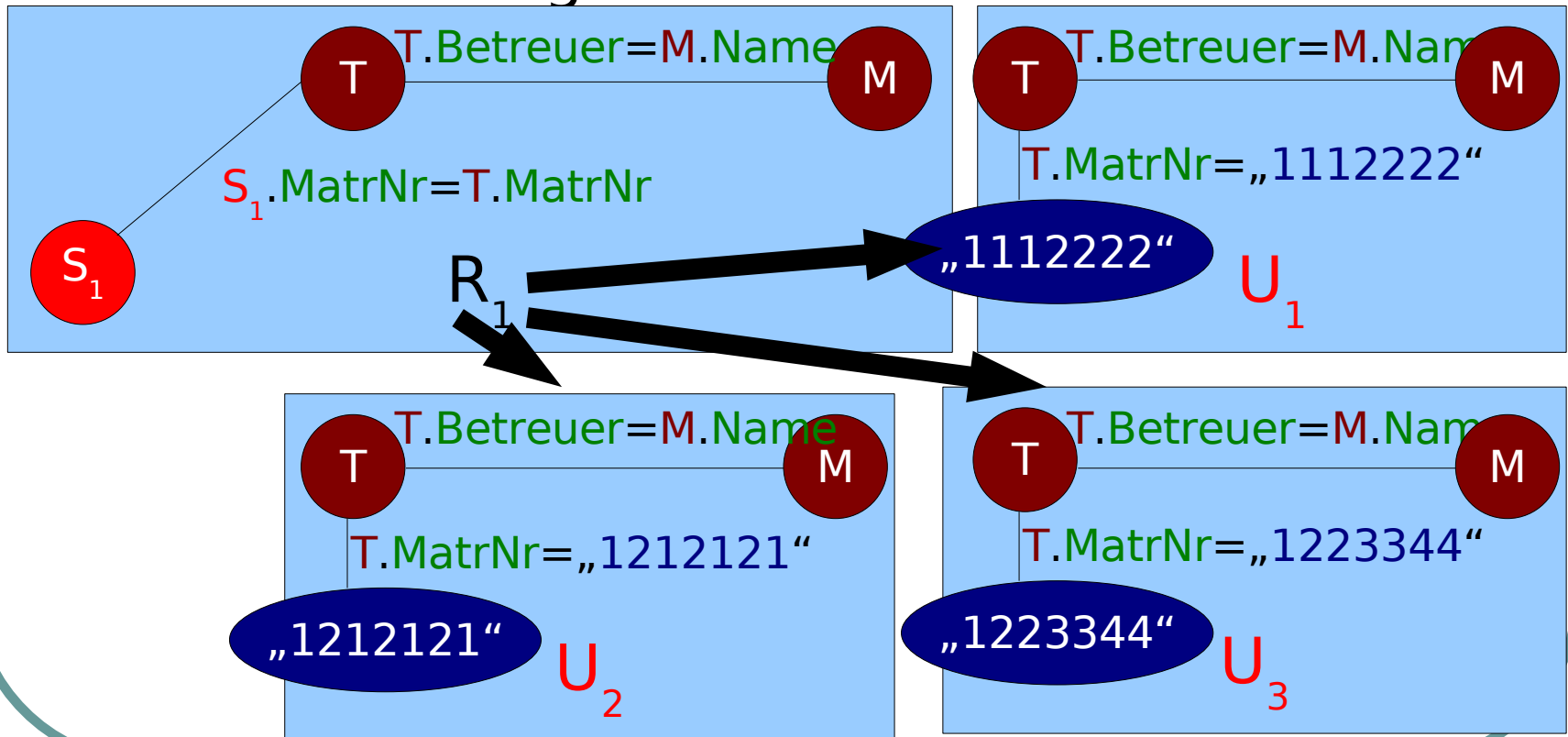
- Ergebnis:



S_1	
MatrNr	Sname
1112222	Marc Philipp
1212121	Niko Huber
1223344	Mathias Reisch

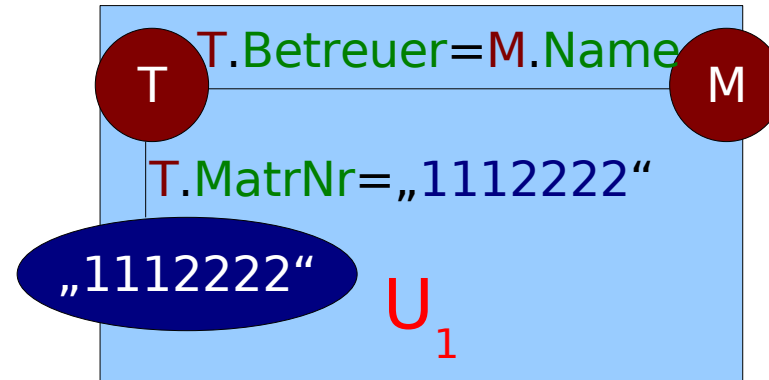
Tupelsubstitution

3) Verwende Tupelsubstitution, um die Multi-Relation-Anfrage um eine Relation zu



Weiterverarbeitung

- 4) Führe den Algorithmus für jede dieser neu entstandenen Multi-Relationen-Anfragen aus
- Beispiel für U_1 im Folgenden (U_2, U_3 analog)



Weiterverarbeitung

- 4) Führe den Algorithmus für jede dieser neu entstandenen Multi-Relationen-Anfragen aus
- Rekursiver Aufruf der Relationen U_1 , U_2 , U_3 liefert folgendes Gesamtergebnis:

U_1	
Sname	Institut
Marc Philipp	IPD

U_2	
Sname	Institut
Niko Huber	IPD

U_3	
Sname	Institut
Mathias Reisch	IPD



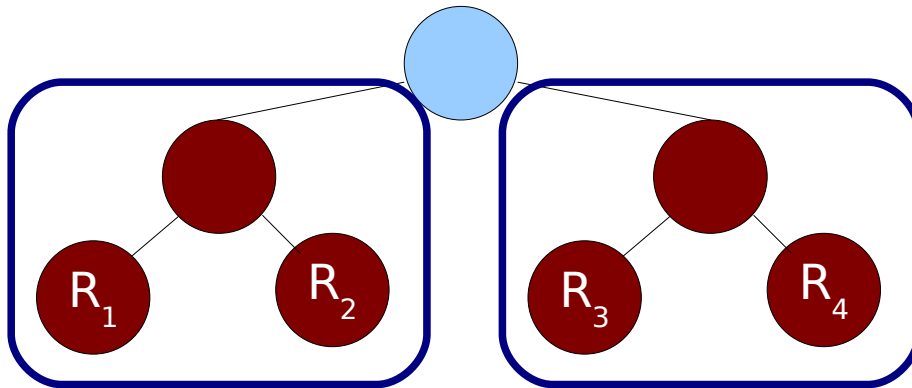
Gesamtergebnis	
Sname	Institut
Marc Philipp	IPD
Niko Huber	IPD
Mathias Reisch	IPD

INGRES - Anfrageoptimierung

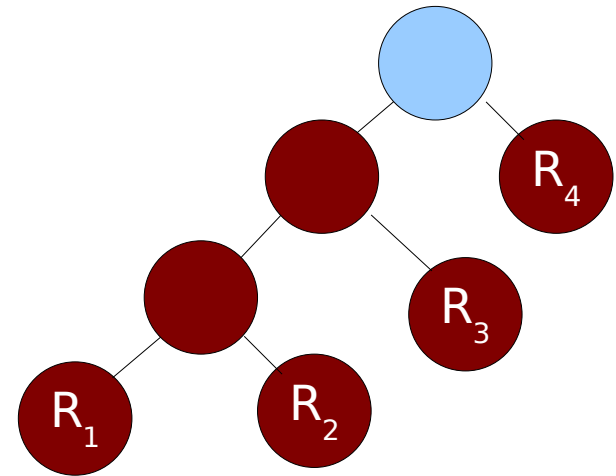
- Anfrageoptimierung im Zentralen Fall
- Vorgehen: (Detachment/Substitution)
 - 1) Zerlege Anfrage mit vielen Relationen in m Ein-Relation-Anfragen und eine Multi-Relation-Anfrage
 - 2) Führe jede Ein-Relation-Anfrage aus
 - 3) Verringere Multi-Relation-Anfrage um eine Relation durch Tupelsubstitution
 - 4) Führe den Algorithmus für jede dieser neu entstandenen Multi-Relationen-Anfragen aus
- Im Folgenden: Verteilter Fall (Erweiterung dieses Algorithmus)

Verteilter Algorithmus

- *Ziel:* Minimierung der *Kommunikationszeit* und der *Antwortzeit* (Kompromiss erforderlich)



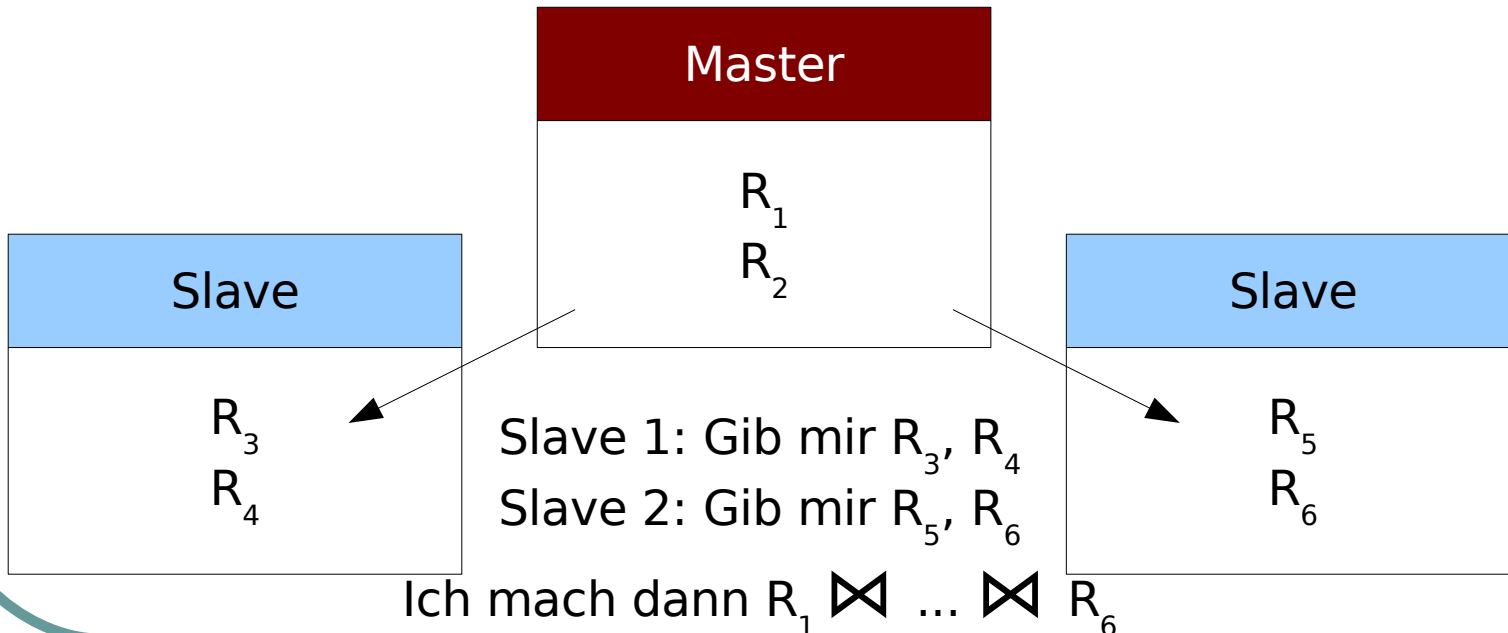
Antwortzeiten
geringer
(durch Parallelität)



Kommunikationszeit
geringer (kleinere
Zwischenergebnisse)

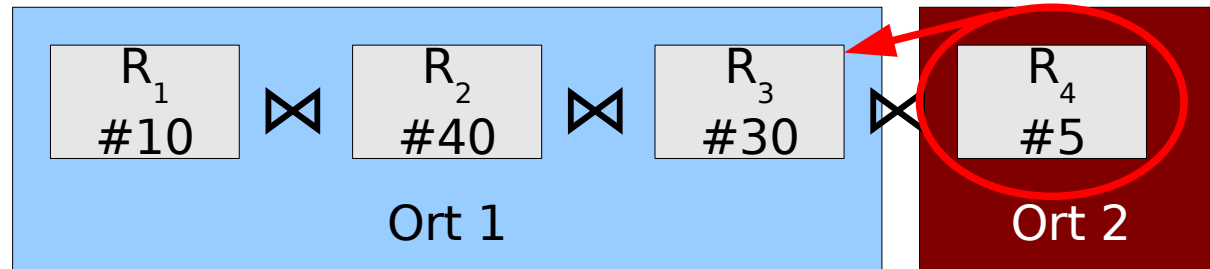
Eigenschaften des vert. Alg.

- Ausführung des Algorithmus von einem „Masterort“
 - Optimierung findet also beim Master statt
 - Entscheidet: **WAS** wird **WO** ausgeführt



Essenzen des vert. Alg.

- Monorelationale Anfragen am Ort, wo sich die Daten befinden, ausführen
- Auswahl des Ortes der nächsten Unterabfrage: Relation mit geringerer Kardinalität wird zum Ort der anderen Relation geschickt



- Fragmentierte Relationen müssen an den Orten, an denen sie verwendet werden werden, komplett rekonstruiert werden

Vert. Algorithmus: Beispiel

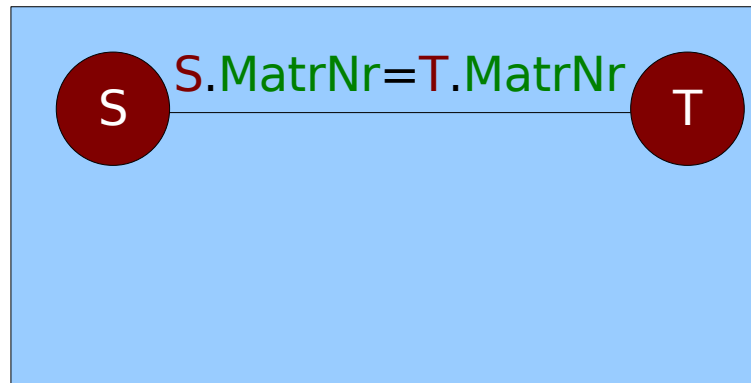
- Szenario nun wie zu Beginn:
S an Ort 1 (Studienbüro)
M, T an Ort 2 (IPD)

Ort 1	Ort 2
S #25000	M #100 T #100

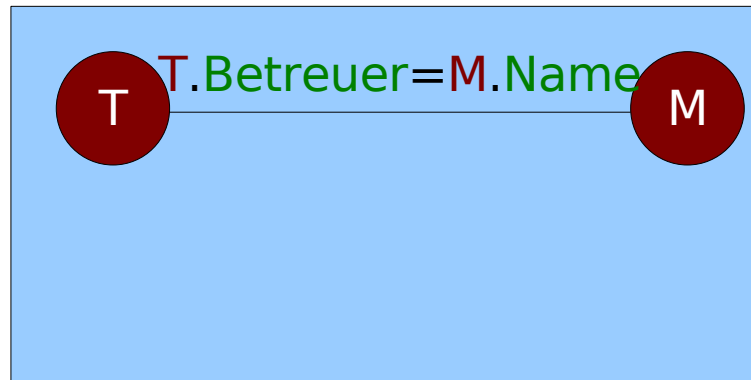
Vert. Alg.: mögl. Strategien

- Erster Schritt: Wahl des ersten Verbunds:

1)



2)



Strategiewahl

- Entscheidung anhand **erwarteter Ergebnisgröße**, lokale Sicht
- *Kommunikationskostenminimierung*:
Wähle 2), T und M am gleichen Ort, Kosten 0
- *Abwicklungszeitminimierung*:
Suche $\min\{PT(1), PT(2), PT(3)\}$,
PT(i): Abschätzung der Abwicklungszeit von Strategie i
- Kompromiss aus beiden extremen Sichtweisen sinnvoll

Überblick

- 1) Motivation
- 2) Anfrageoptimierung in INGRES
- 3) Anfrageoptimierung in System R**
- 4) Kritik
- 5) Vergleich der Algorithmen

System-R-Alg.: Eigenschaften

- statisches Verfahren: Abschätzung der Zwischenergebnisse vor Ausführung
- dynamische Programmierung
 - fast erschöpfende Suche
 - Äquivalente, teurere Strategien werden verworfen

Eigenschaften des Alg.

- nur links-tiefe Join-Bäume ($n!$ Stück)
- Bottom-up Generierung von Anfrageplänen
- CPU-Last, Seitenzugriffe werden berücksichtigt
- Konstruktion der Bäume: zunächst $R_i \bowtie R_j$,
danach $(R_i \bowtie R_j) \bowtie R_k$
 - teurere Permutationen verwerfen
 - Obere Grenze: 2^n Permutationen

Vorgehen anhand Beispiel I

- „Welche Informatiker nehmen an einem Seminar an welchem Institut teil?“
- **SELECT** Sname, Institut
FROM Mitarbeiter M, Teilnehmer T, Student S
WHERE M.Name = T.Betreuer
AND T.MatrNr = S.MatrNr
- Indizes:
M (Mitarbeiter) auf Name
T (Teilnehmer) auf MatrNr
S (Student) auf MatrNr

Vorgehen anhand Beispiel II

1) Bestimme billigsten Zugriffspfad für jede Relation, basierend auf einem SELECT-Prädikat

- M: sequent. Scan (da keine Selektion auf M)
oder: Index auf Join-Attribut, falls innere Relation
- T: sequent. Scan (da keine Selektion auf T)
oder: Index auf Join-Attribut, falls innere Relation
- S: Index auf Sname (da Selection auf S, basierend auf Sname)

Vorgehen anhand Beispiel III

2) Erzeuge Baum der Alternativstrategien

M

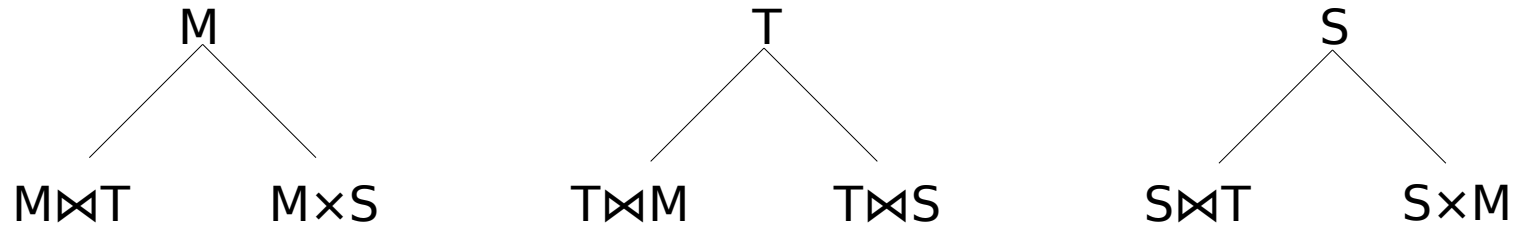
T

S

- Erste Ebene des Baums.
- Beste Zugriffsmethoden für Einzelrelationen

Vorgehen anhand Beispiel IV

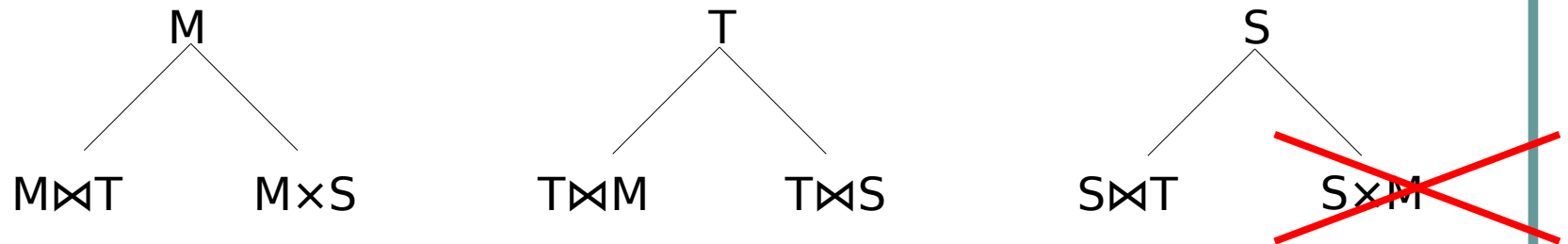
2) Erzeuge Baum der Alternativstrategien



- Zweite Ebene des Baumes

Vorgehen anhand Beispiel V

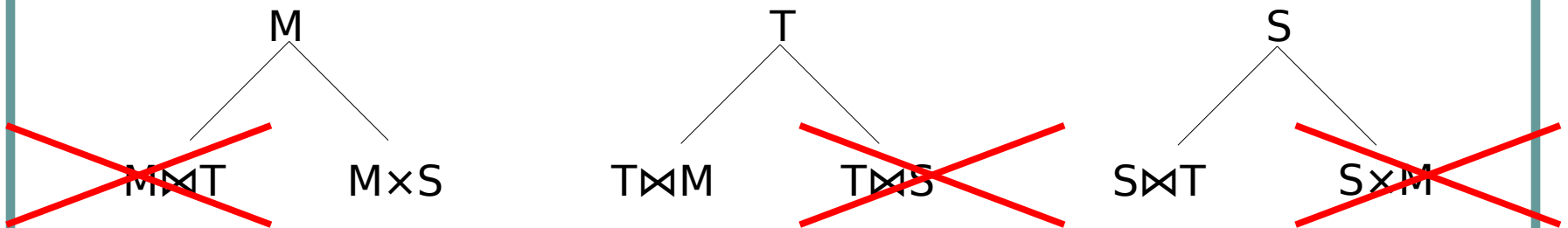
2) Erzeuge Baum der Alternativstrategien



- Zweite Ebene des Baumes
- Entferne eines der kartesischen Produkte (sehr teuer, implizit)

Vorgehen anhand Beispiel VI

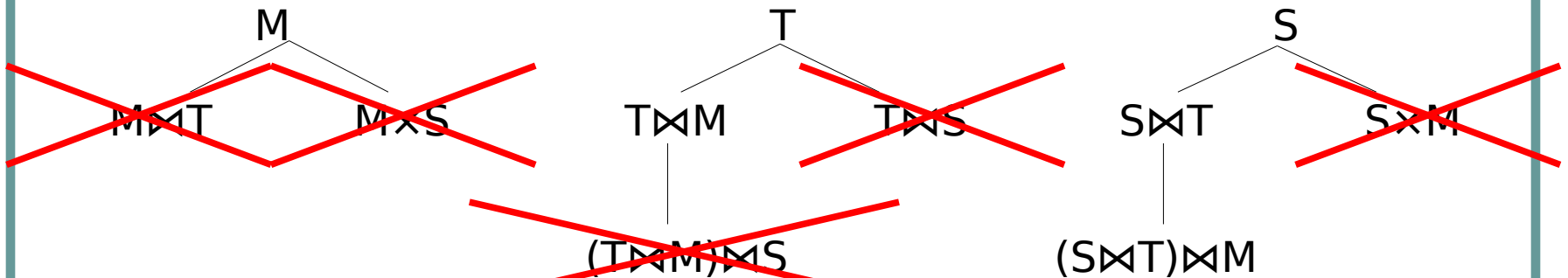
2) Erzeuge Baum der Alternativstrategien



- $cost(M⋈T) \geq cost(T⋈M)$
 $cost(T⋈S) \geq cost(S⋈T)$
- entfernen, da äquivalent (Kommutativität)

Vorgehen anhand Beispiel VII

2) Erzeuge Baum der Alternativstrategien



- $(S \bowtie T) \bowtie M$ besitzt Index für Selektion des Attributs Sname in S
- $(S \bowtie T) \bowtie M$ entfernen, teurer
- $M \times S$ kann entfernt werden, da $M \times S \bowtie T$ teurer

Vorgehen anhand Beispiel VIII

3) Wähle am Schluß den Baum mit geringsten Kosten

- $(S \bowtie T) \bowtie M$ benutzt folgende Zugriffsmethoden:
 - Wähle S mit Hilfe des Index auf Sname
 - Verbinde mit T mit Hilfe des Index auf MatrNr
 - Verbinde mit M mit Hilfe des Index auf Name

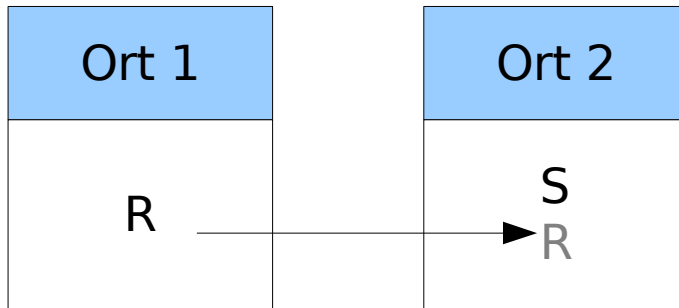
System R - Optimierung

- Anfrageoptimierung im zentralen Fall:
- Vorgehen:
 - 1) Bestimme billigsten Zugriffspfad für jede Relation, basierend auf einem SELECT-Prädikat
 - 2) Erzeuge und bewerte Kandidatenbäume, (Permutationen der Join-Bäume)
 - Menge der Alternativstrategien wird dynamisch konstruiert, indem von äquivalenten Joins der billigere Join behalten wird
 - 3) Wähle am Schluß den Baum mit geringsten Kosten
- Im Folgenden verteilter Fall (wo wird etwas ausgeführt)

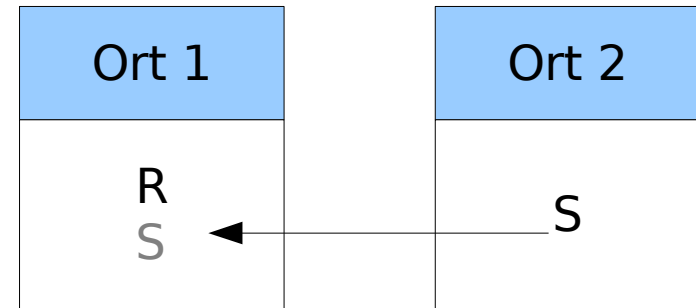
Verteilter Fall: Strategien

- Vier Strategien

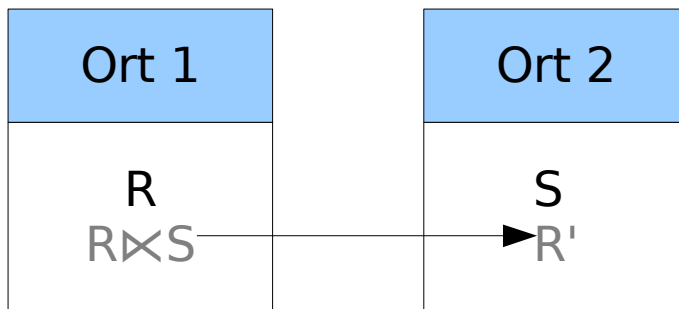
1)



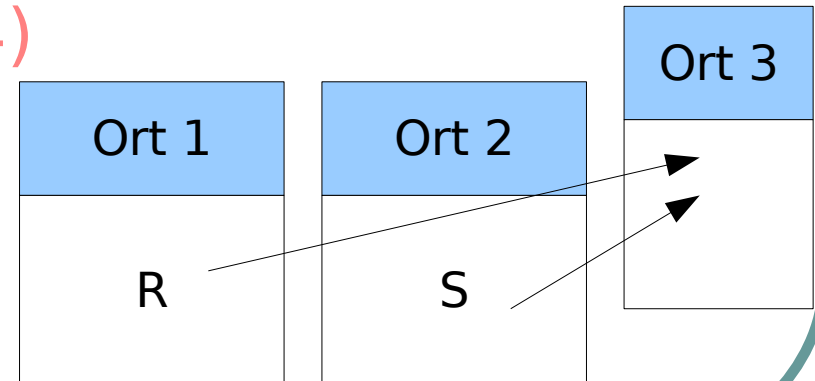
2)



3)



4)



Semijoin: Beispiel

- Join der äußeren Relation R mit der Projektion des Verbundattributs der inneren Relation

R	S	$R \bowtie_{R.A=S.E} S$																																					
<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th><th>D</th></tr></thead><tbody><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>4</td><td>5</td><td>6</td><td>7</td></tr><tr><td>7</td><td>8</td><td>9</td><td>0</td></tr></tbody></table>	A	B	C	D	1	2	3	4	4	5	6	7	7	8	9	0	<table border="1"><thead><tr><th>E</th><th>F</th><th>G</th></tr></thead><tbody><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>7</td><td>8</td><td>9</td></tr></tbody></table>	E	F	G	1	2	3	7	8	9	<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th><th>D</th></tr></thead><tbody><tr><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>7</td><td>8</td><td>9</td><td>0</td></tr></tbody></table>	A	B	C	D	1	2	3	4	7	8	9	0
A	B	C	D																																				
1	2	3	4																																				
4	5	6	7																																				
7	8	9	0																																				
E	F	G																																					
1	2	3																																					
7	8	9																																					
A	B	C	D																																				
1	2	3	4																																				
7	8	9	0																																				

- $R \bowtie_{R.A=S.E} S$ enthält die Daten, die für den $R \bowtie S$ notwendig sind
- Es werden also nur notwendige Tupel verschickt

Beispiel

- Anfrage: $S \bowtie_{\text{MatrNr}} T$
S (Student): externe Relation an Ort 1
T (Teilnehmer): interne Relation an Ort 2
Index auf MatrNr für T
- Strategien:
 - 1) Gesamtes S zu T schicken
 - 2) T zu S schicken
 - 3) Benötigte T Tupel für jedes Tupel von S holen
 - 4) S und T an einen dritten Ort schicken

Beispiel II

- Strategie 4 wählen, falls Anfrage nicht von Ort 1 oder 2 stammt (aber: $card(S)$ groß)
- Da $size(S) \gg size(T)$, minimiert Strategie 2 die Kommunikationszeit
 - gute Wahl, falls lokale Abwicklungszeit geringer ist als bei Strategie 1 und 3
 - Bei 1 und 3 aber Ausnutzung des Index
- Lokale Abwicklungskosten bei 1 und 3 sind identisch
- Falls wenige Tupel mit T übereinstimmen: 3 besser, sonst 1

Überblick

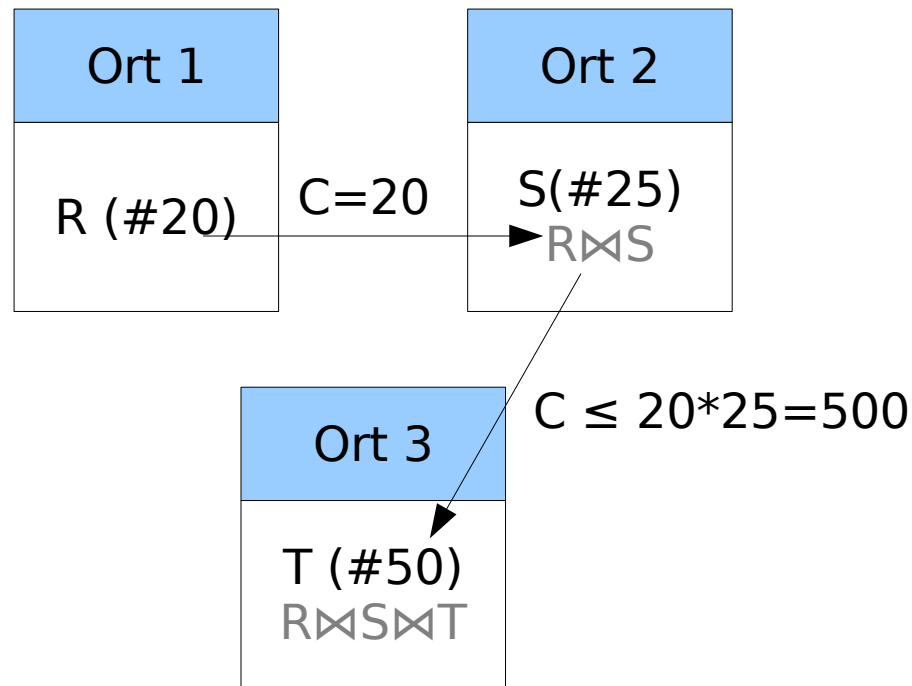
- 1) Motivation
- 2) Anfrageoptimierung in INGRES
- 3) Anfrageoptimierung in System R
- 4) Kritik**
- 5) Vergleich der Algorithmen

Kritik an den Algorithmen I

- Distributed INGRES
 - Anfragebearbeitungsdauer schwierig abschätzbar, da INGRES jedes mal optimiert, nachdem die nächsten zwei Relationen für einen Verbund gewählt wurden
 - Zwischenergebnisse werden zwar minimiert, aber nur aus lokaler Sicht (im Folgenden)

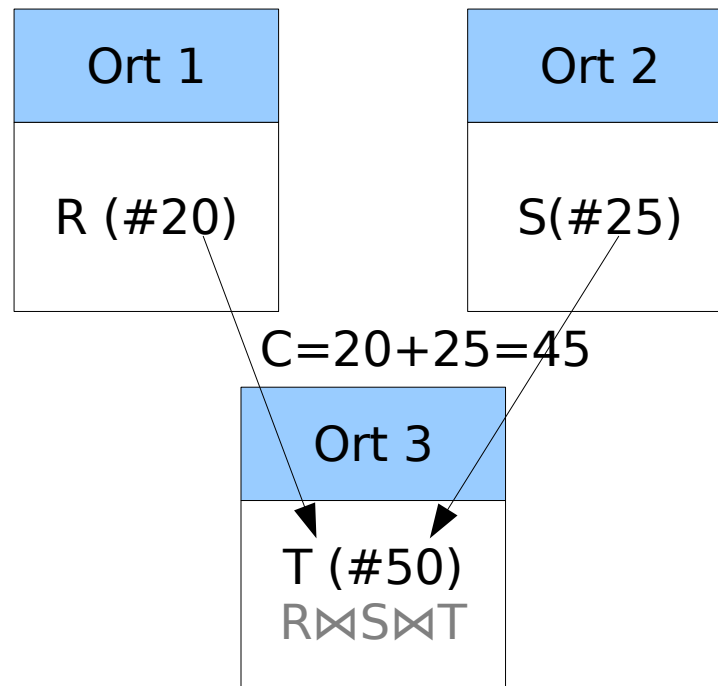
Konsequenz der lokalen Sicht

- Anfrage von Ort 3: $R \bowtie S \bowtie T$



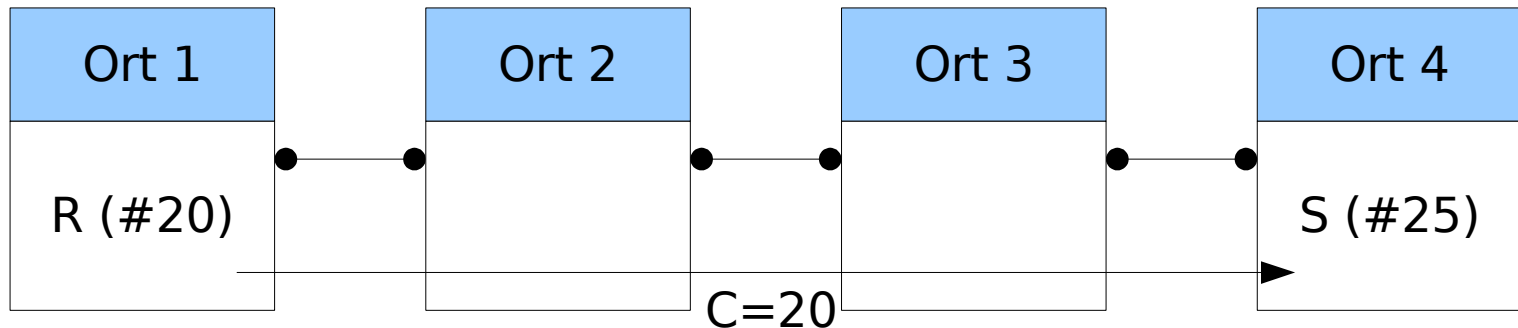
Es geht aber besser...

- Folgende Alternative wird gar nicht betrachtet:

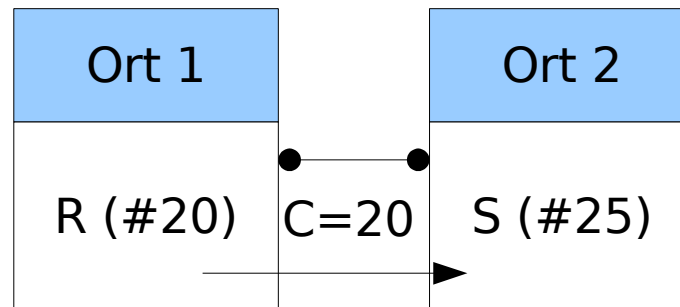


Kritik an den Algorithmen II

- Distributed INGRES (Fortsetzung)
 - Kostenmodell mangelhaft (Anzahl Hops unberücksichtigt)



kostet das gleiche wie

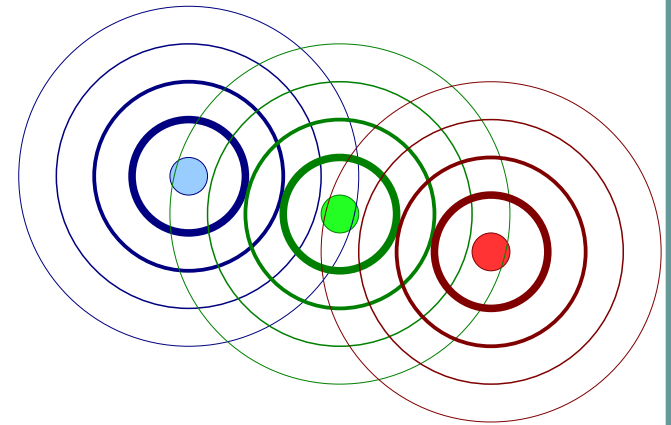


Kritik an den Algorithmen III

- R*
 - Gleiches Kostenmodell wie Distr. INGRES
 - Fehler bei Abschätzungen mit Statistiken werden propagiert
 - Für ständig wechselnde Anfragen ungeeignet, da Aufwand nicht amortisiert wird
 - Für Anfragen mit vielen Relationen (≥ 5) ungeeignet (da zu aufwändig)

Ungeeignet für Sensornetze

- Schlechter Umgang mit fragmentierten Relationen
- Fragmente werden wie normale Relationen betrachtet
- Keine tieferen Überlegungen dahinter
- Kostenmodell betrachtet zentrale Verarbeitung auf einem Knoten als optimal → falsch



Überblick

- 1) Motivation
- 2) Anfrageoptimierung in INGRES
- 3) Anfrageoptimierung in System R
- 4) Kritik
- 5) Vergleich der Algorithmen**

Algorithmen im Vergleich

Algorithmus	Verteilter INGRES	R*
Typ	Erweiterung des zentralen Alg.	Erweiterung des zentralen Alg.
Optimierungsart	Dynamisch	Statisch
Technik	Heuristik	Dynamisches Programmieren
Zielfunktion	Antwortzeit oder Gesamtkosten	Gesamtkosten
Optimierungsfaktoren	Nachrichtengröße, Abwicklungskosten	Nachrichtengröße, I/O, CPU
Verwendete Statistiken	card(Relation)	card(Relation), #eind. Werte/ Attr.
Fragmente	Horizontal	Nein

Fragen

Gibt es Fragen?

Danke

Danke für Eure Aufmerksamkeit.