Technische Universität
Kaiserslautern

Seminar: Datenbanken und Informationssysteme SS 05
Lehrgebiet Datenverwaltungssysteme

Thema: Data Streams

Plangenerierung, Optimierung und Anfrageverarbeitung in DSMS

Sebastian Bächle

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Überblick

- Phasen der Anfrageverarbeitung
- Kostenmodelle für Anfragepläne
- Bewertung von Joins in DSMS
- Spezielle Join-Algorithmen
  - Temporal Progressive Merge Join
  - XJoin
  - Nutzung von Metadaten
- Fazit

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Phasen der Anfrageverarbeitung (1)

Die Anfrageverarbeitung verläuft klassischerweise in 4 Phasen:

1. Übersetzungsphase

- lexikalische, syntaktische und semantische Analyse
- Transformation in Anfragegraph

2. Plangenerierungs- und Optimierungsphase

- Erstellen von Anfrageplänen
- Auswahl der Planoperatoren
- Optimierung der Operatorfolgen

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Phasen der Anfrageverarbeitung (2)

3. Ausführungsphase

- Installation des Anfrageplanes
- Ableiten der Ergebnisse
- dynamische Reorganisation

4. Bereitstellungsphase

- Bereitstellen der Ergebnisse
- Sortierung

- ▶ Überschneidung einzelner Phasen
- ▶ neue Herausforderungen vor allem in den Phasen 2 und 3

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Ziele der Anfrageverarbeitung

- Schnelle Ausgabe von (unvollständigen) Ergebnissen
- Hohe Durchsatzraten
- Lastausgleich
- Skalierbarkeit
- Unterstützung von QoS-Vorgaben
- Vielseitige Anfrageoperationen und -typen
- Universelle Einsetzbarkeit
- ...

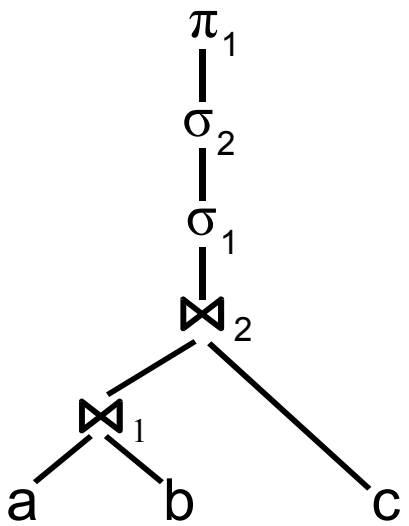
TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Übersetzung

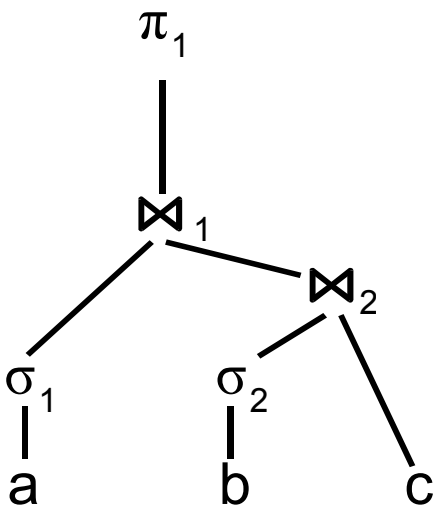
Beispielanfrage:

```
SELECT Sa.ID, Sc.ID, Sb.timestamp
FROM a Sa[500], b Sb[500], c Sc[100]
WHERE Sa.ID = Sb.targetID
AND Sb.requestID = Sc.requestID
AND Sa.type = 'DEV-4711'
AND Sb.state = 'BUSY'
```

Anfragegraph AG_1



Anfragegraph AG_2



TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Plangenerierung

Algebraische Optimierung:

- Anwenden von Transformationsregeln auf Anfragegraphen
- Äquivalenz bei Datenströmen: Snapshot-Operator muss zu jedem Zeitpunkt t gleiche Multimengen liefern
- nicht alle Regeln aus DBMS anwendbar

Nicht-algebraische Optimierung:

- Auswahl der Planoperatoren
- Anpassen des Schedulings
- wird während der Anfrageplanbewertung vorgenommen

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Ausführung und Bereitstellung

Ausführung:

- Installation des Anfrageplanes
- kein Vorkompilieren von Anfrageprogrammen sondern z.B. Initialisieren der Planoperatoren aus XML-Dateien
- Überwachung des Stromflusses und eventuell erneutes Optimieren
- Lastkontrolle: Scheduling, Load shedding, ...

Bereitstellung:

- parallel zur Ausführung
- evtl. Triggern von Aktionen

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Rate-based Optimization (1)

Idee: Bestimme die Ausgaberate eines Planes durch Abschätzen der produzierten Ergebnisse und der dafür benötigten Zeit:

- r_{OUT} für Selektion und Projektion einfach zu berechnen:

$$c_{OP} > 1/r_{IN} \rightarrow r_{OUT} = 1/c_{OP} \qquad c_{OP} \leq 1/r_{IN} \rightarrow r_{OUT} = r_{IN}$$

- Herleitung von r_{OUT} eines Joins:

1. Bestimme die Anzahl der erzeugten Ergebnisse im Intervall t:

$$n = f_1 r_a r_b t \cdot (t-1)$$

2. Bestimme Kosten des Joins im Intervall t:

$$C_{\Delta t} = t \cdot r_a \cdot c_a + t \cdot r_b \cdot c_b = t \cdot (C_a + C_b)$$

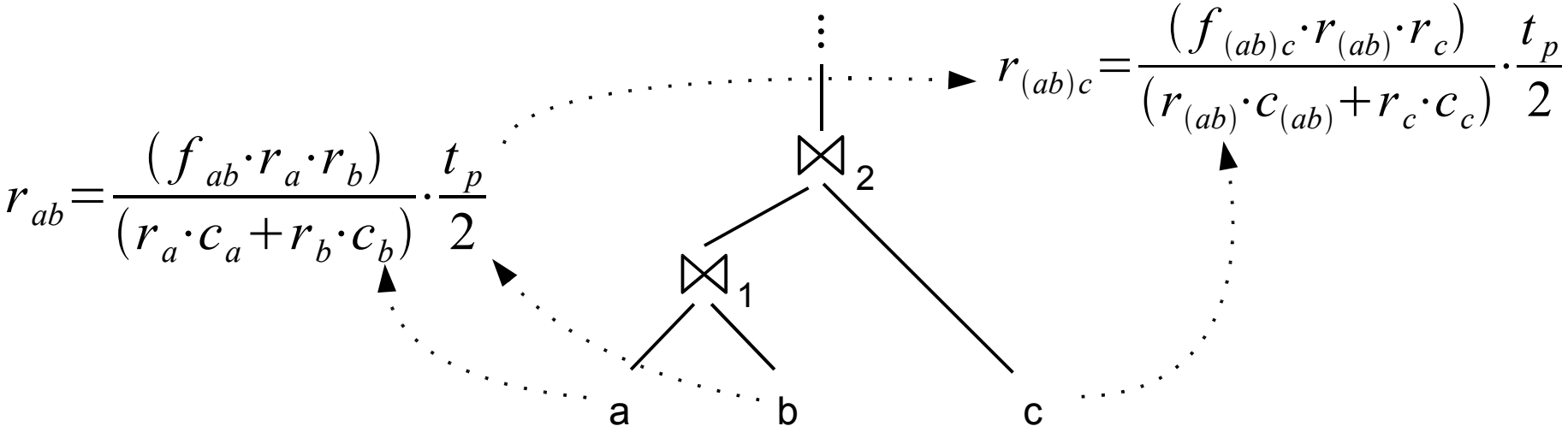
TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Rate-based Optimization (2)

3. Bestimme Outputrate des Joins als Funktion der Zeit:

$$r_{out} = \frac{f_1 r_a r_b t \cdot (t-1)}{t \cdot (C_a + C_b)} \approx \frac{f_1 r_a r_b t}{C_a + C_b}$$

► Propagiere Outputraten für Gesamtbewertung:

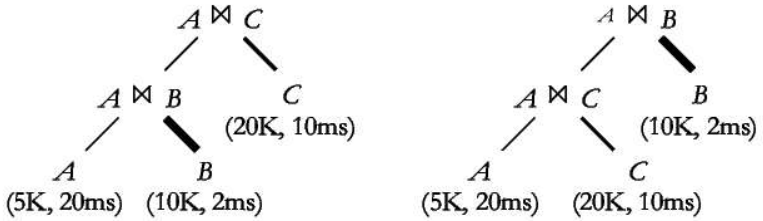


► Wähle $t_p/2$ als Heuristik zum Bestimmen der mittleren Gesamtrate im Intervall t

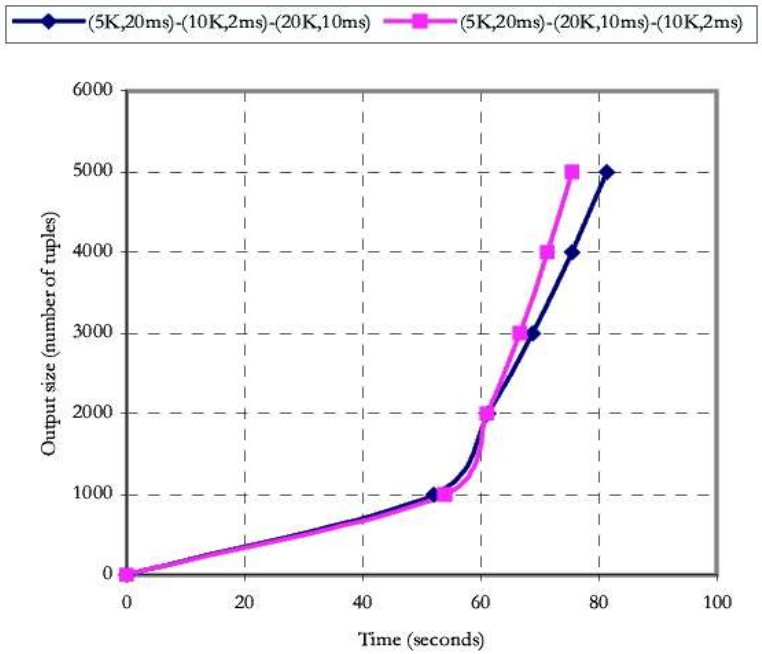
TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Ergebnisse und Messungen

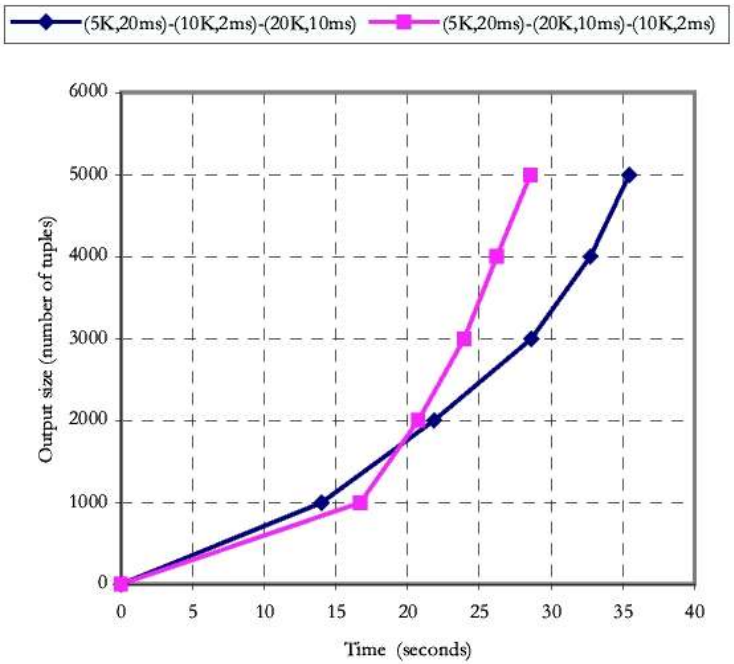
Bestimme die beste Join-Reihenfolge:



Erwartete Performance:



Gemessene Performance:

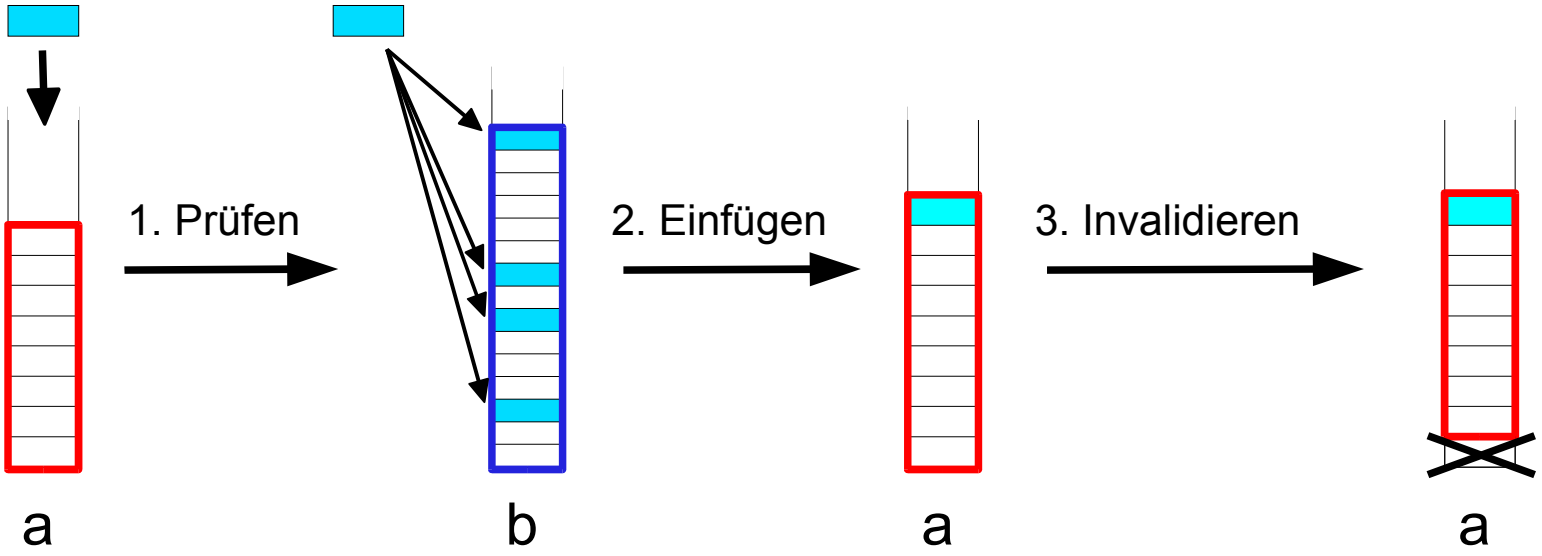


TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Bestimmung der Operatorkosten

Frage: Wie können die Kosten eines Join-Operators besser bestimmt werden?

- Betrachte die einzelnen Schritte beim Eintreffen eines neuen Tupels:



TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

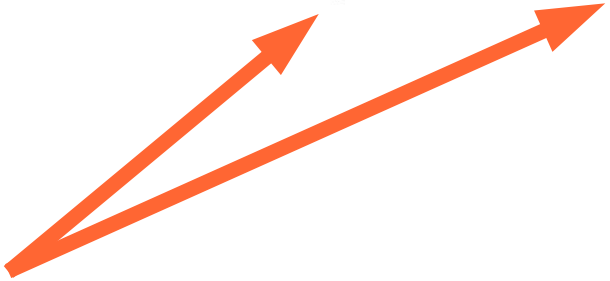
Kosten des Nested Loops Joins

- Für einen einseitigen Join berechnet man allgemein die Kosten:

$$C_{a \rtimes b} = r_a \cdot (Prüfen(b)) + r_b \cdot (Einfügen(b) + Invalidieren(b))$$

- Der Nested Loops Join hat somit die Kosten:

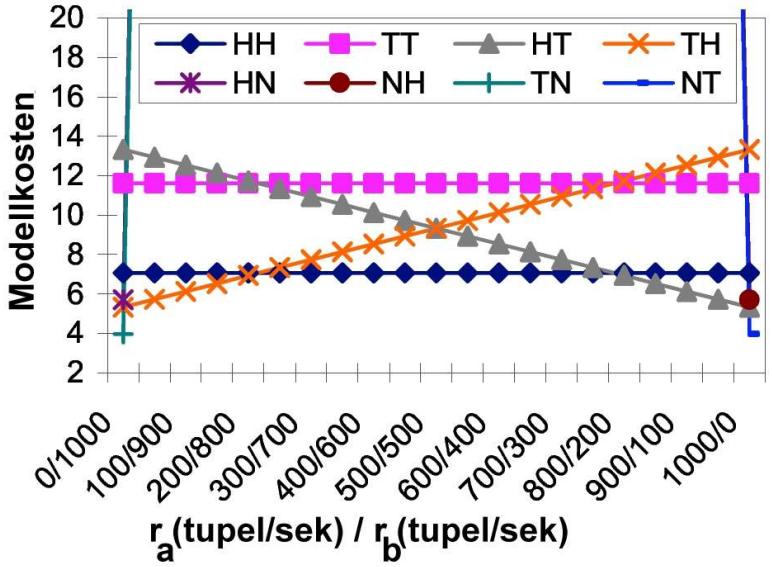
$$C_{a \rtimes b}(NJ) = r_a \cdot B \cdot c_{p,n} + 2 \cdot r_b \cdot c_{i,n}$$

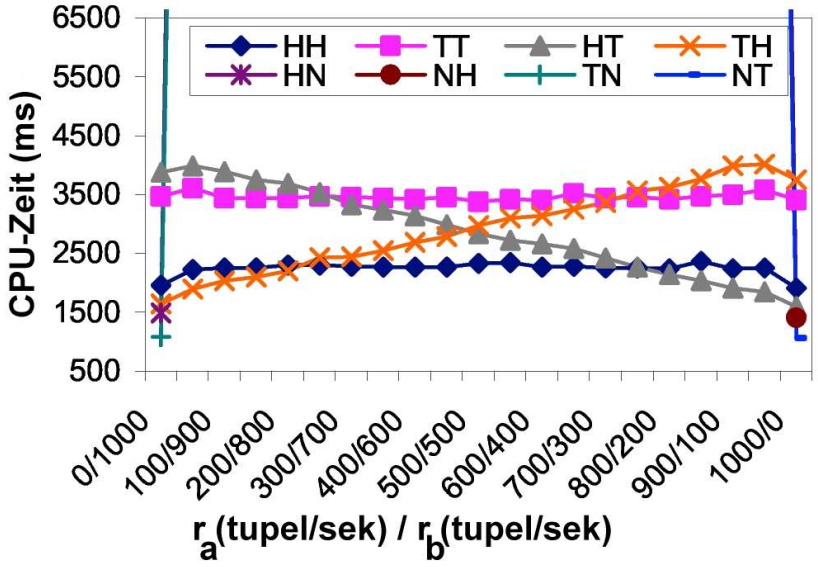


- Die Aktionskosten zum Prüfen und Einfügen müssen durch Messungen bestimmt werden.

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Beispiel-Messungen der Kosten





- Getrennte Betrachtung der Join-Richtungen erlauben die Bewertung von Join-Kombinationen!
- Vorteilhaftigkeit eines Planoperators in Intervallen bestimmbar

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Temporal Progressive Merge Join

Nutze Prinzip des externen Merge Sort:

- Sortiere Teilfolgen und schreibe sie als Run aus
- Erstelle Join-Ergebnisse schon beim Sortieren und Mischen
- Fenstersemantik: Erzeuge Verbund nur, wenn sich die Gültigkeitsintervalle zweier Tupel nicht leer schneiden

► Nutzung von Sekundärspeicher

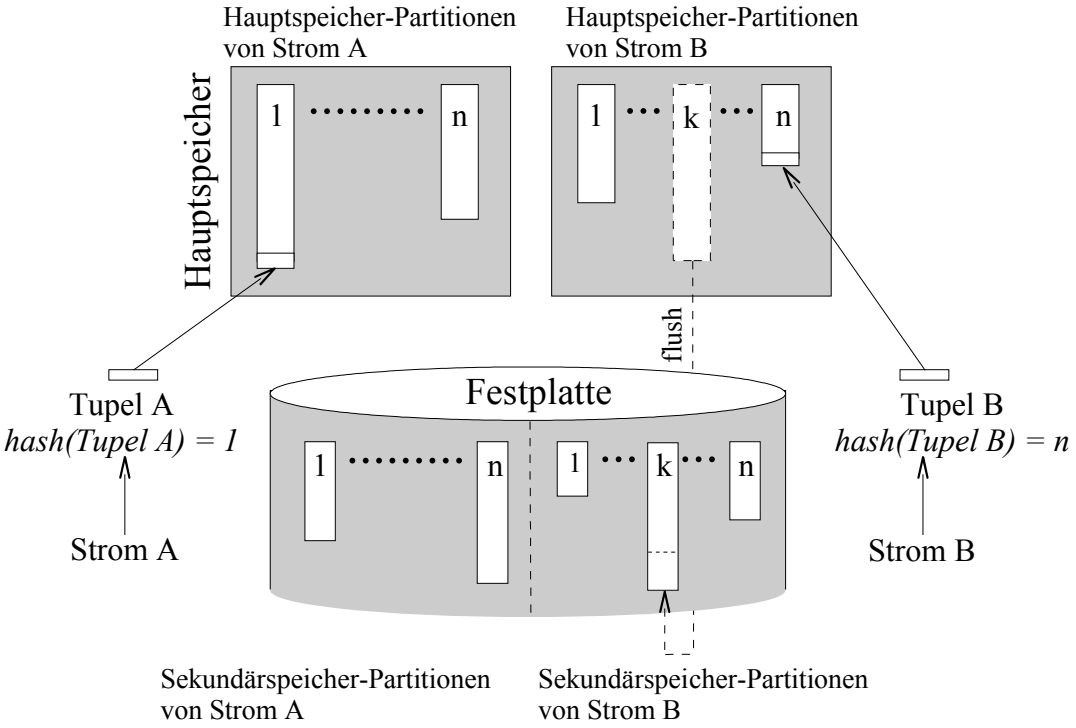
► Benötigt Strategien zur Duplikatvermeidung

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

XJoin (1)

- Erzeuge möglichst schnell aktuelle Ergebnisse
- Ergebnisse werden in drei Phasen erzeugt
- Variante des symmetrischen Hash Joins: Teile Hash-Partitionen in memory-resident und disc-resident portions

Phase 1:

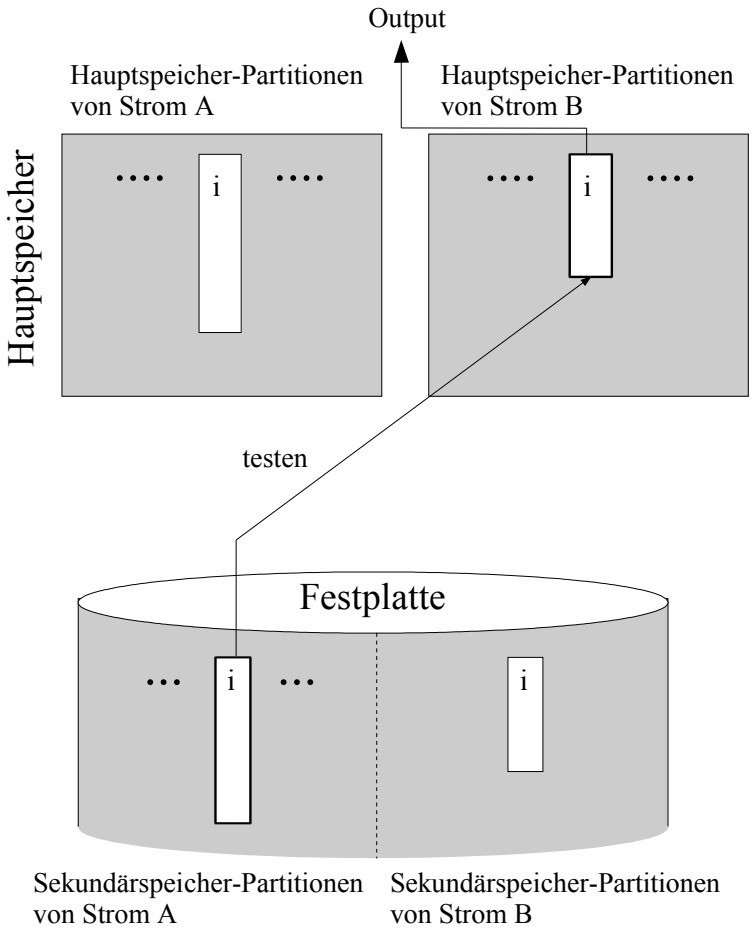


TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

XJoin (2)

- Phase 2 abwechselnd zu Phase 1
- Phase 3 ist reine Aufräumphase

Phase 2:



TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Punctuations

- unterteile einen unendlichen Strom in eine unendliche Menge endlicher Subströme
- Punctuations beschreiben Muster die nicht mehr auftreten
- Tupel können früher verworfen werden

```
<stream:NewsGeneral>
<info>
<sno> 79 </sno>
<title> DFB-Elf schlägt Brasilien mit 2:1 </title>
<keyword> Fußball </keyword>
</info>
...
</stream:NewsGeneral>
```

Punctuation: Es folgen keine Tupel mehr mit diesem Muster

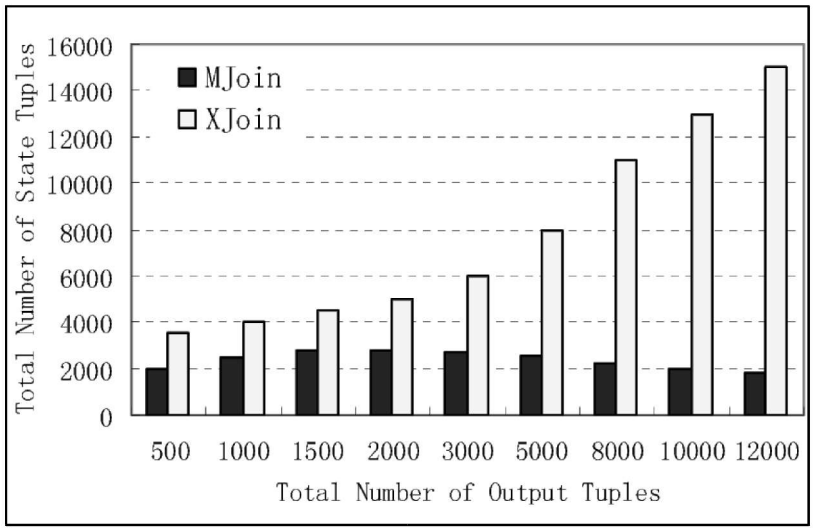
sno	*ipaddr*
79	*

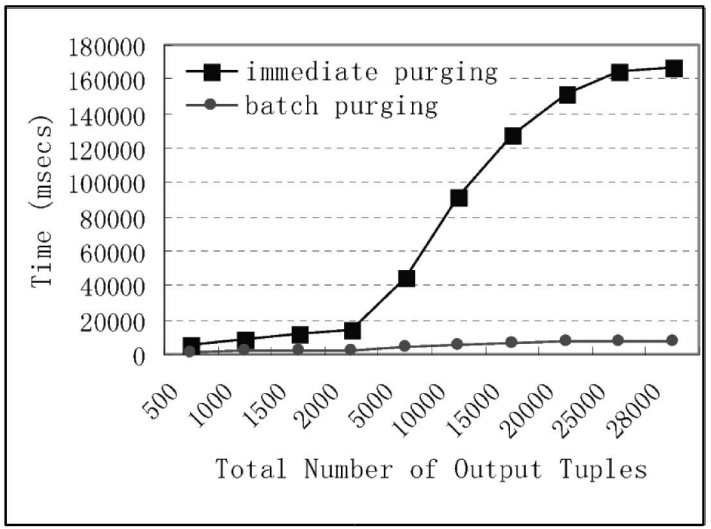
```
<stream:AccessRecord>
<record>
 <sno> 79 </sno>
 <ipaddr> 64.58.76.224 </ipaddr>
</record>
<record>
 <sno> 81 </sno>
 <ipaddr>207.68.172.234</ipaddr>
</record>
<record>
 <sno> 79 </sno>
 <ipaddr> 202.112.39.8 </ipaddr>
</record>
<punct:record>
 <sno> 79 </sno>
 <ipaddr> * </ipaddr>
</punct:record>
...
</stream:AccessRecord>
```

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

MJoin

- Erweiterung des XJoins
- Erkennung 1:N-Joins mit Hilfe der Schemadaten
- Effektiv, da in praktischen Anwendungen oft Bursts gleichartiger Tupel
- Scheduling ist entsprechend anzupassen!





TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Fazit

- Zumindest Grundzüge der Anfrageverarbeitung ähnlich zu DBMS
- Anfrageplanbewertung muss z. T. sehr unterschiedliche Planoperatoren vergleichbar machen
- Stetige, dynamische Reorganisation von Anfrageplänen sollte möglichst effizient unterstützt werden
- Multi-Query-Optimierung besonders bei vielen kontinuierlichen Anfragen notwendig
- Viele Ansätze müssen noch besser aufeinander abgestimmt werden

Noch Fragen?