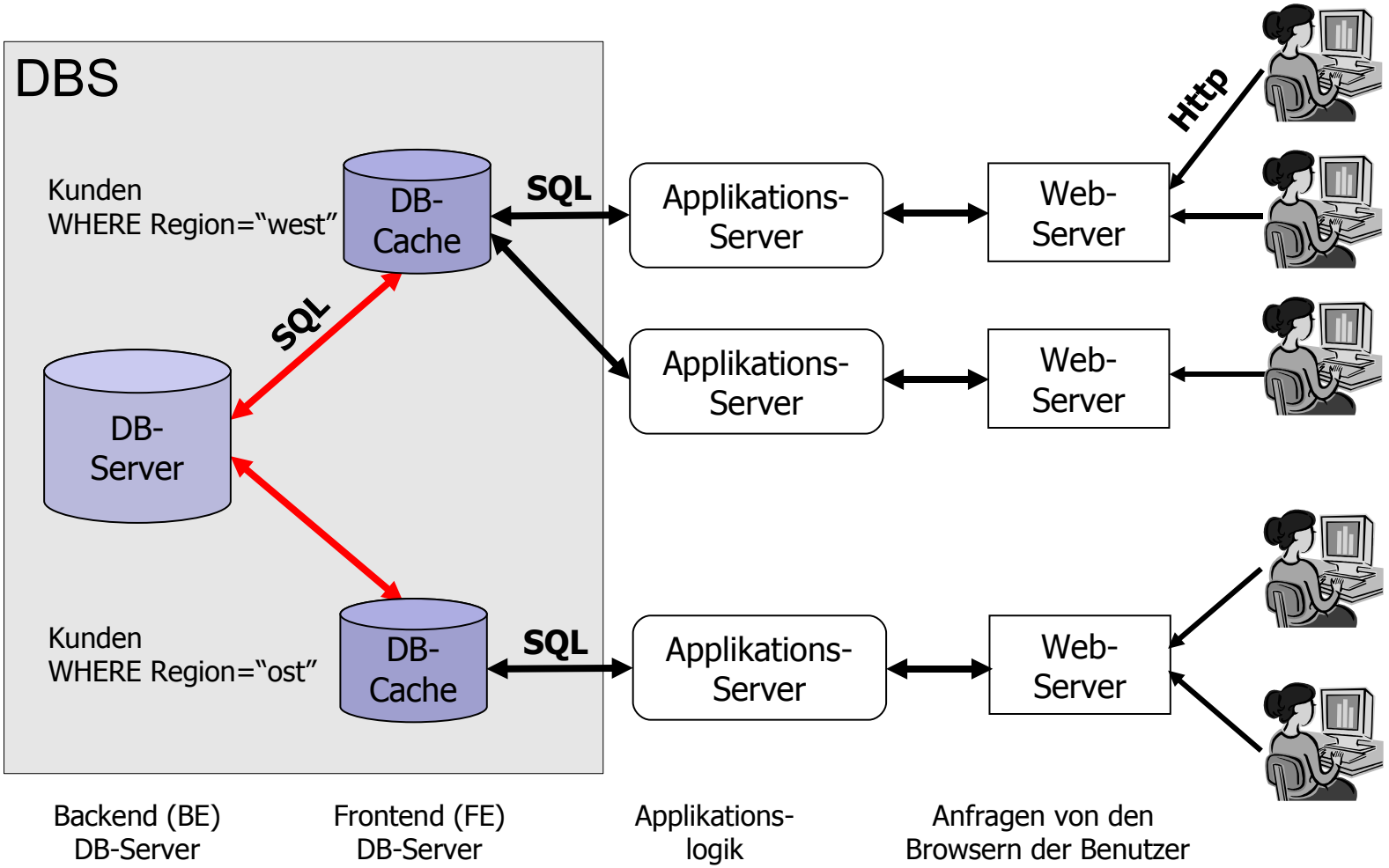
TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

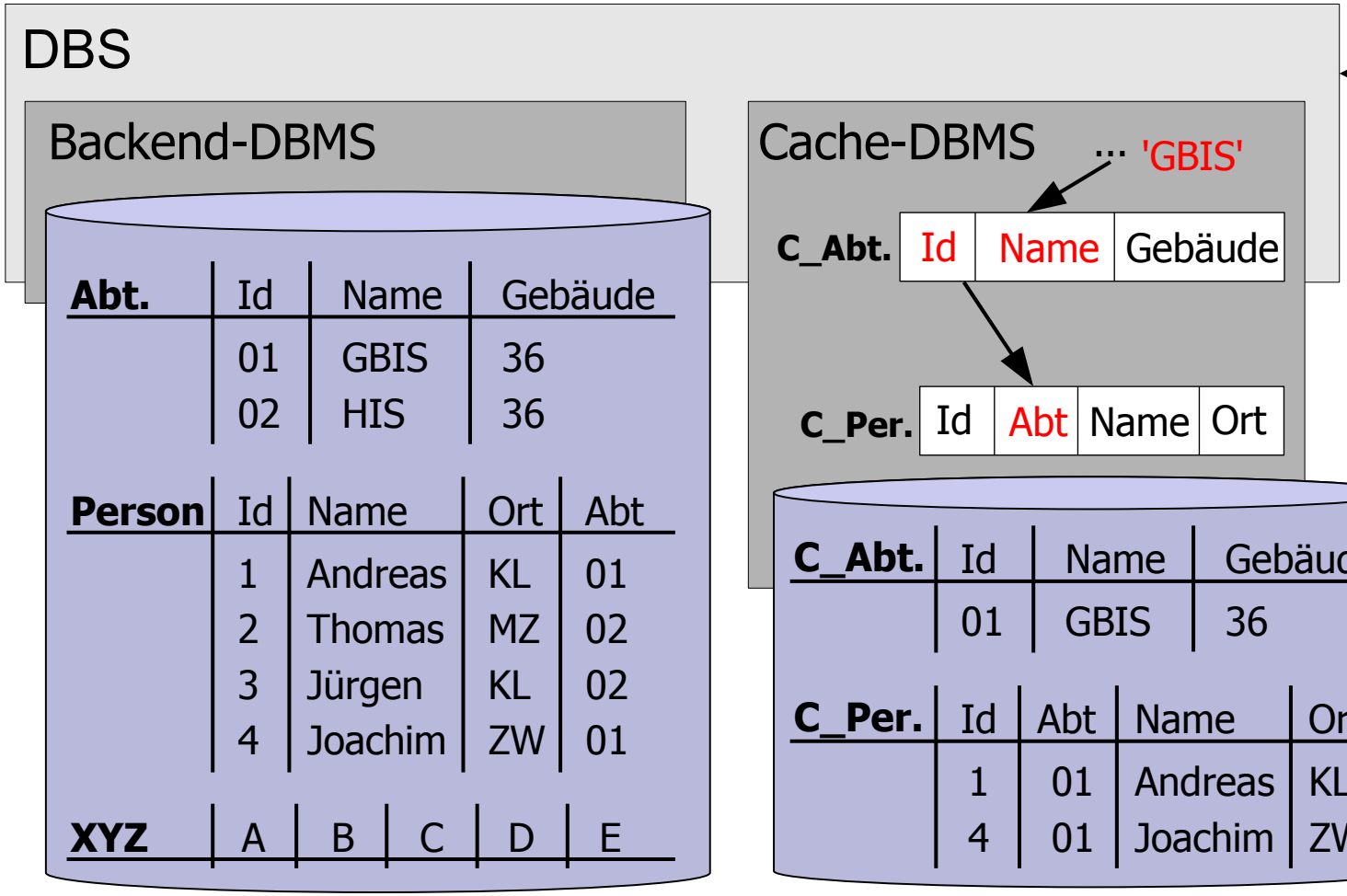
Selektives Laden und Entladen von Prädikatsextensionen beim Constraint-basierten Datenbank-Caching

Joachim Klein, Susanne Braun, Gustavo Machado
05.03.2009

Datenbank-Caching



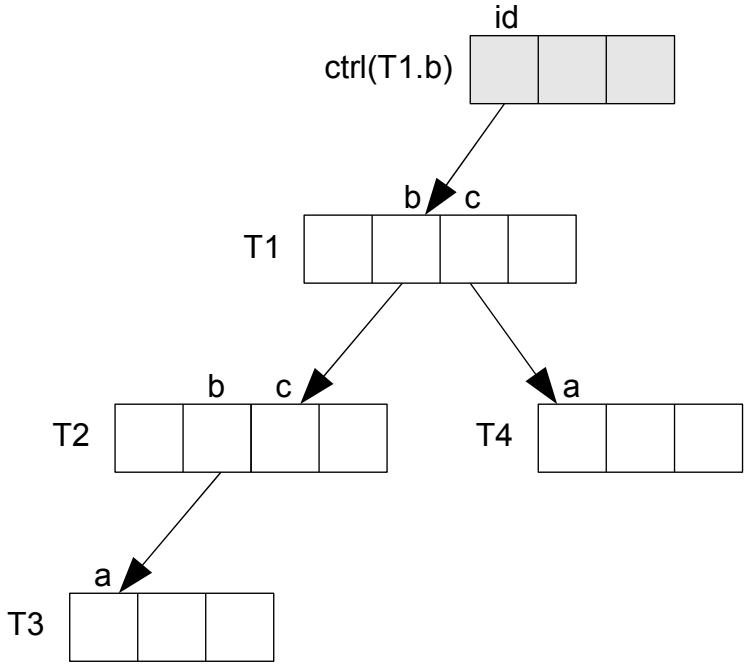
Funktionsprinzip
(Constraint-basiertes Datenbank-Caching)



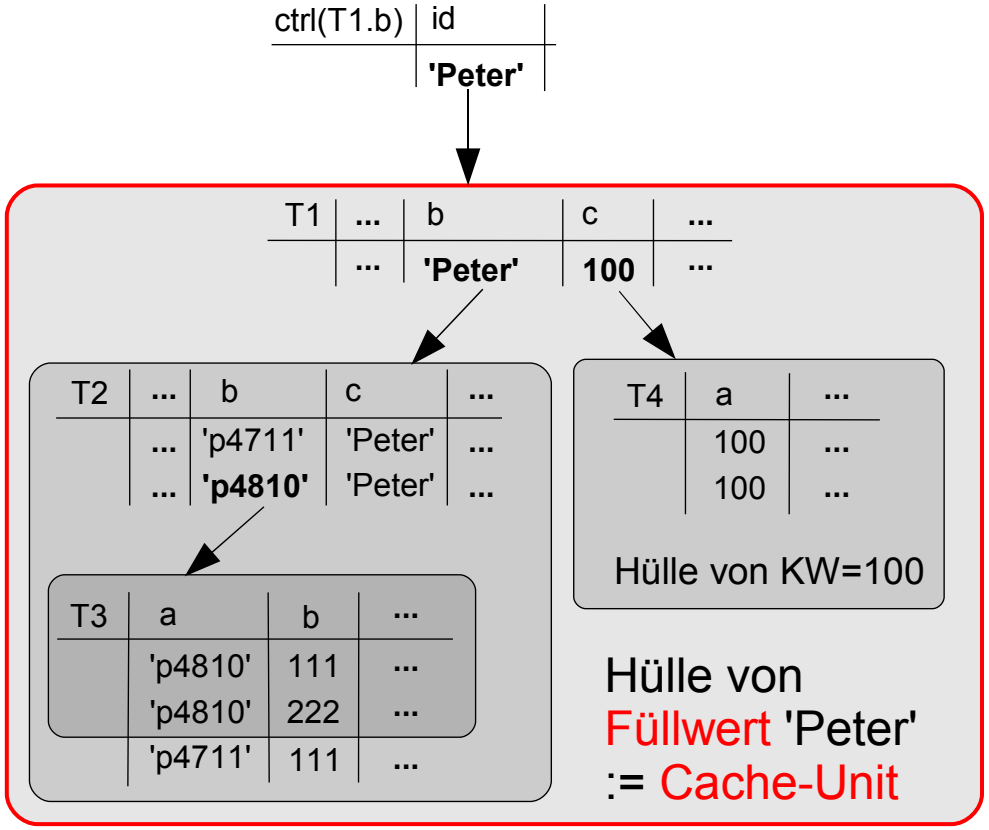
```
SELECT p.Id, p.Name
FROM Person, Abt
WHERE p.Abt=a.Id
AND a.Name='GBIS'
order by p.Name
```

Laden und Entladen

Cache Units

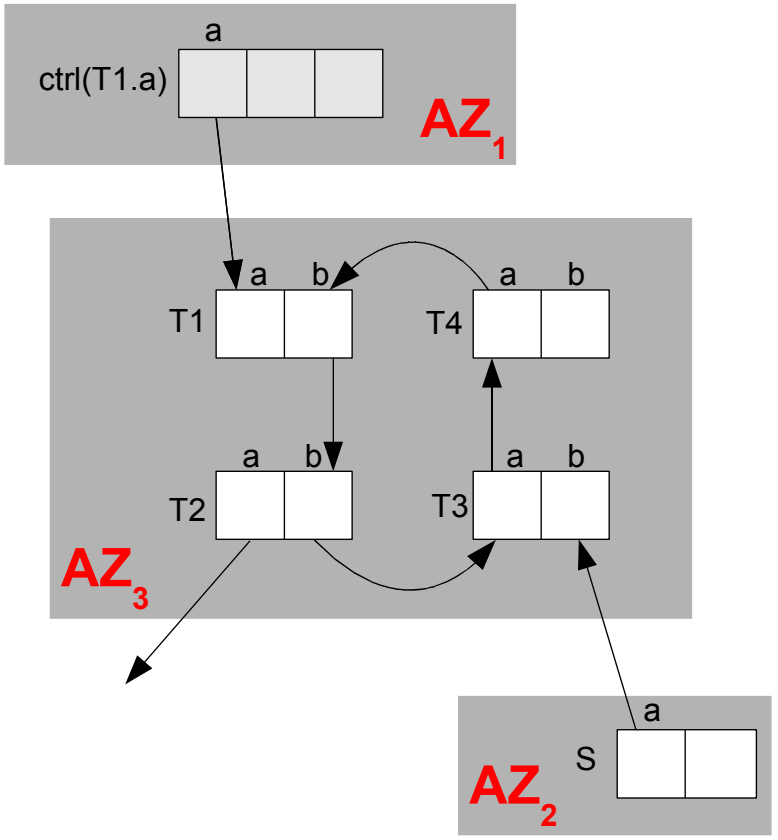


RCC-Quellspaltenwert := Kontrollwert
Kontrollwert in Kontrolltabelle := Füllwert

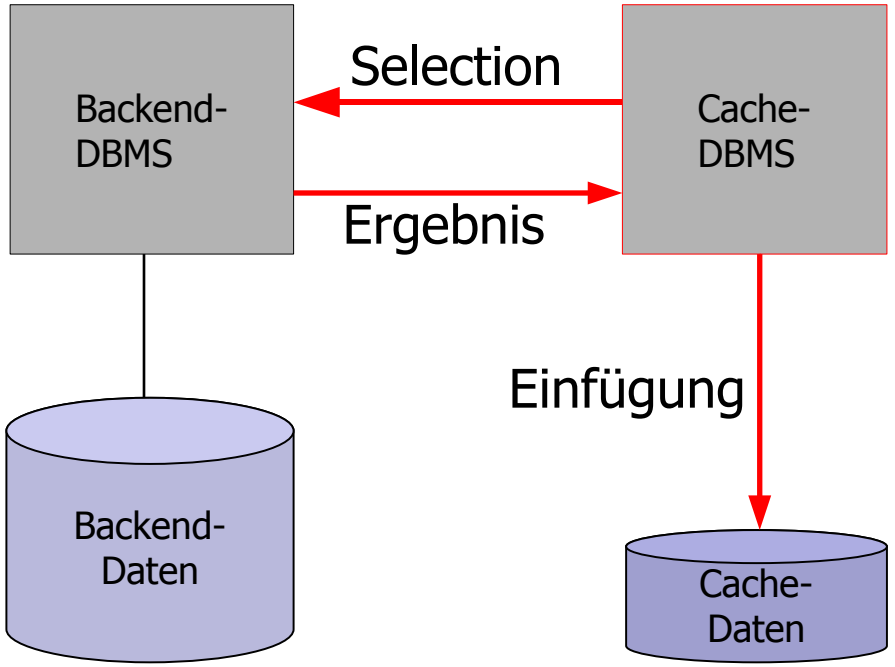


Hauptproblem:
Zyklen → Atomare Zonen

- Zyklen werden durch atomare Zonen gekapselt
  - Homogener Zyklus in AZ_3
  - Heterogene Zyklen
    - → unsicher: nicht erlaubt
- Unterschieden werden:
  - interne RCCs, externe RCCs
- Laden (bottom-up):
  - AZ_3, AZ_2, AZ_1



Direktes Laden



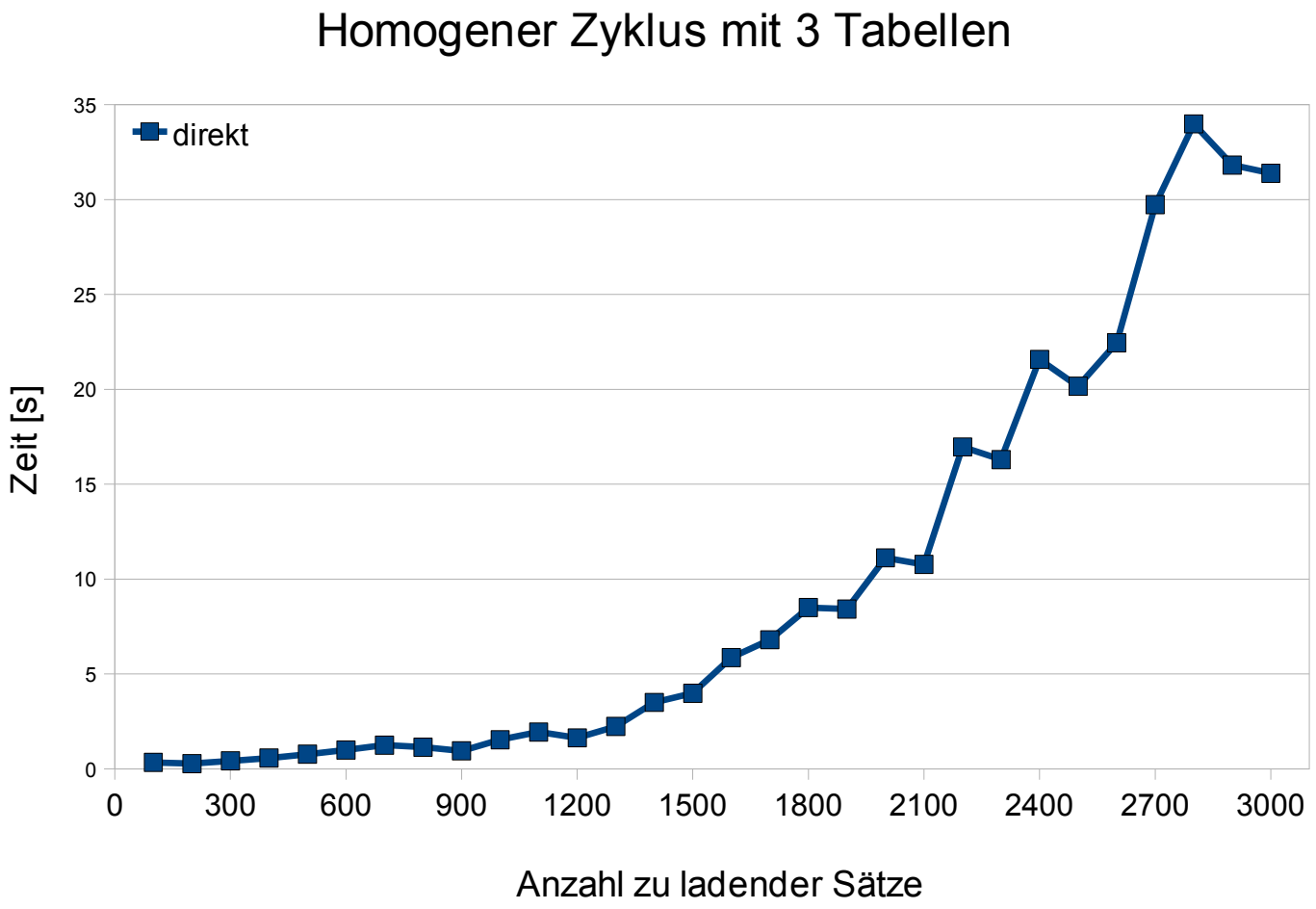
✔ Anfragen+Einfüllen (bottom-up)
✔ Backend-DBMS entlastet
✔ Backend benötigt keinerlei Cache-Metadaten

✗ Mehrfaches Anfragen
➔ Latenzproblematik
✗ Große Abhängigkeit vom Füllgrad der Cache-Tabellen
✗ Unkontrollierbar lange Join-Ketten in Anfragen

- Ablauf:
  - nur schematisch
  - wird mindestens ein mal pro zu füllender Tabelle durchlaufen

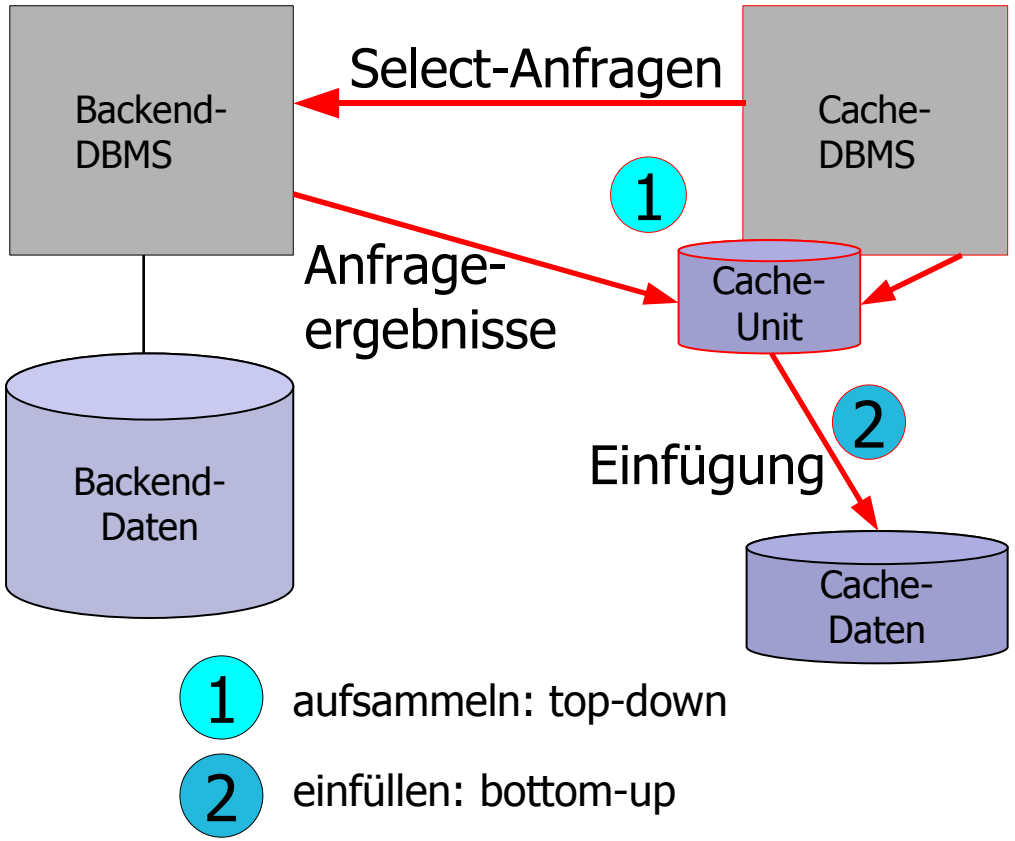
Ergebnisse

Direktes Laden



Indirektes Laden

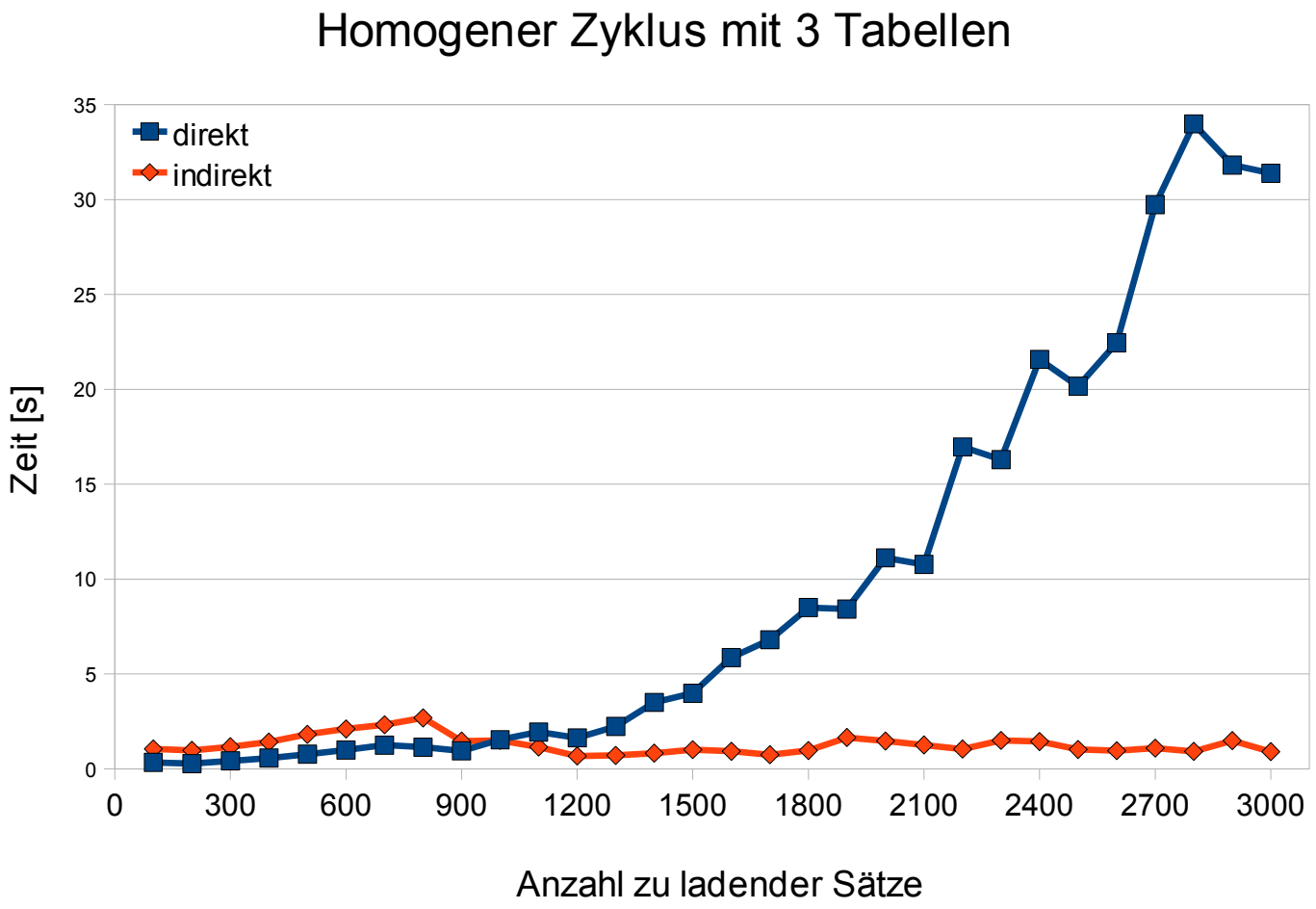
Vorteile



- ✔ Cache-DBMS organisiert Ladevorgang (Backend-DBMS entlastet)
- ✔ Backend benötigt keinerlei Cache-Metadaten
- ✔ Geringe Abhängigkeit vom Füllgrad der Cache-Tabellen

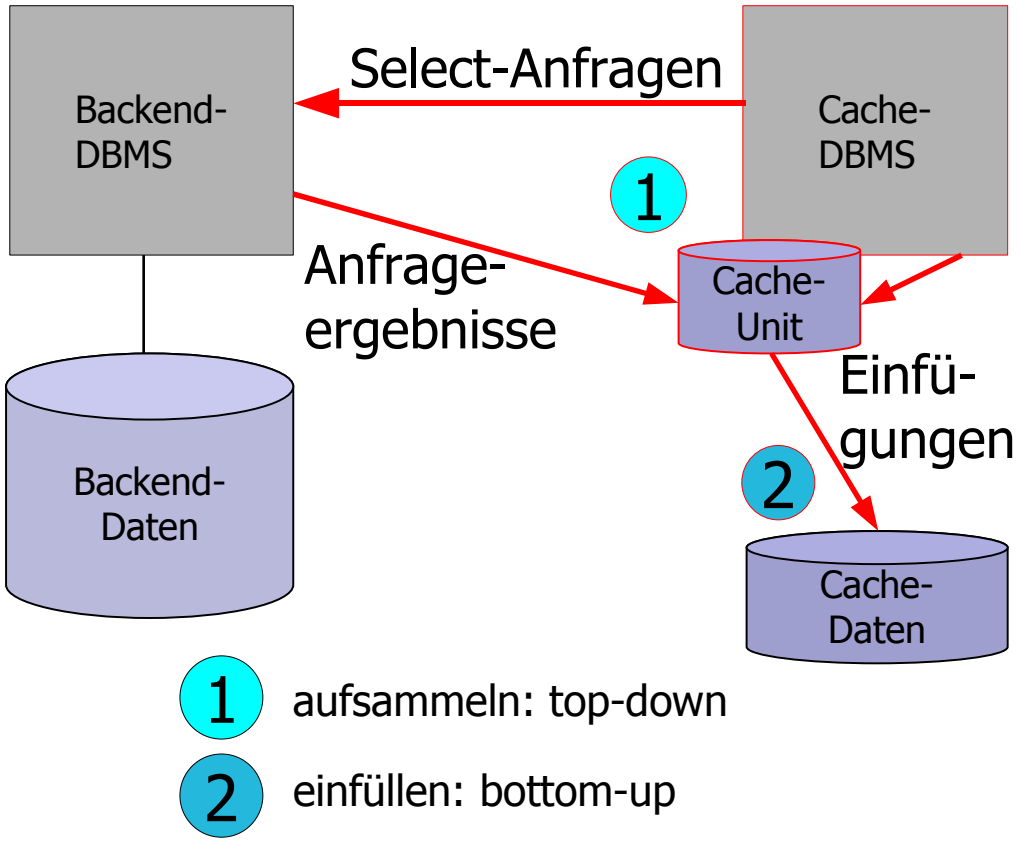
Ergebnisse

Direktes Laden vs. indirektes Laden



Indirektes Laden

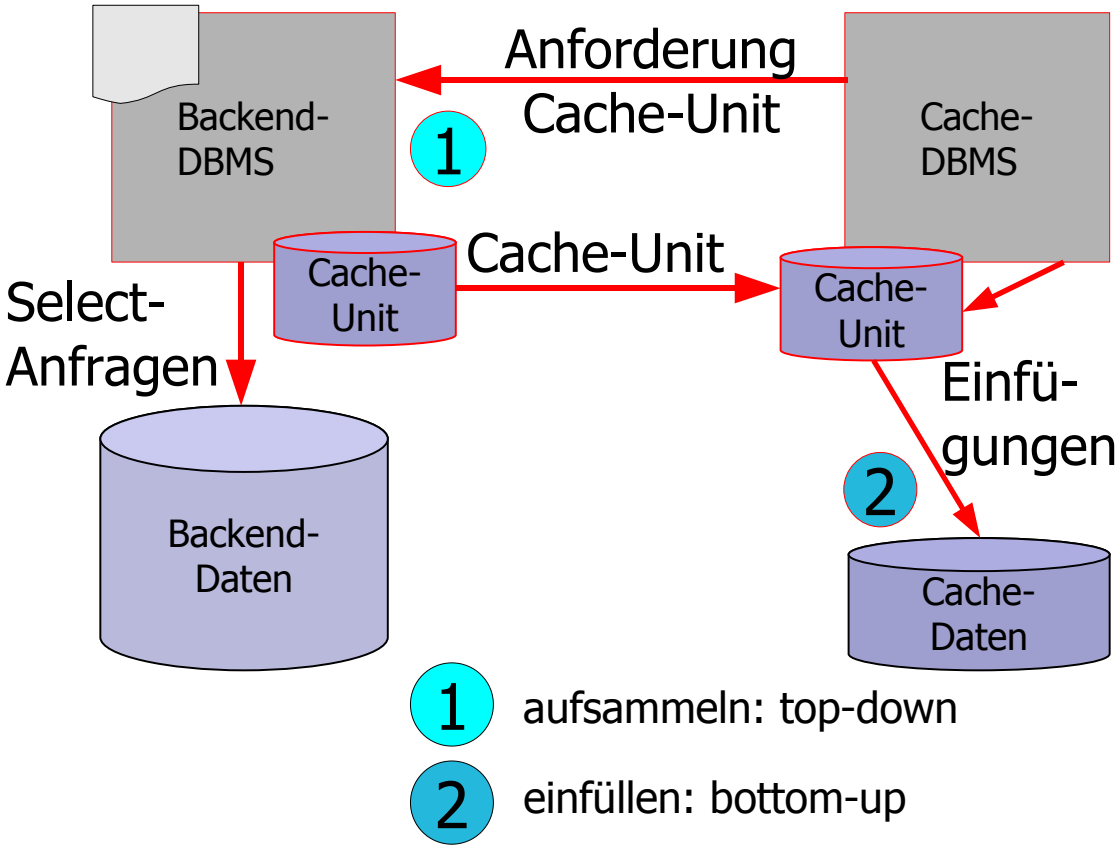
Nachteile



- ✗ Indirektes Einladen
- ✗ Mehrfaches Anfragen vom Cache an Backend
  ➔ Latenzproblematik

Vorbereitetes Laden

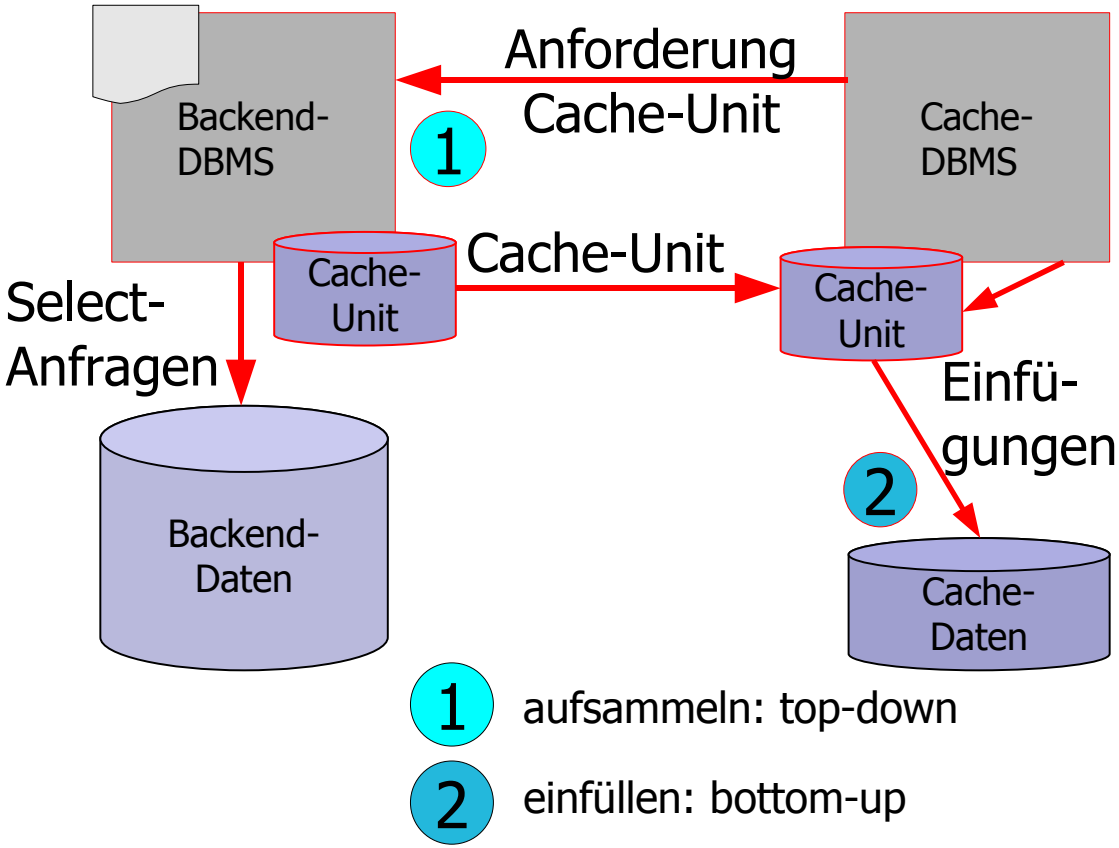
Vorteile



- ✔ Konstante Prädikatlänge in Anfragen
- ✔ Geringe Abhängigkeit vom Füllgrad der Cache-Tabellen
- ✔ Einmalige Anfrage an das Backend

Vorbereitetes Laden

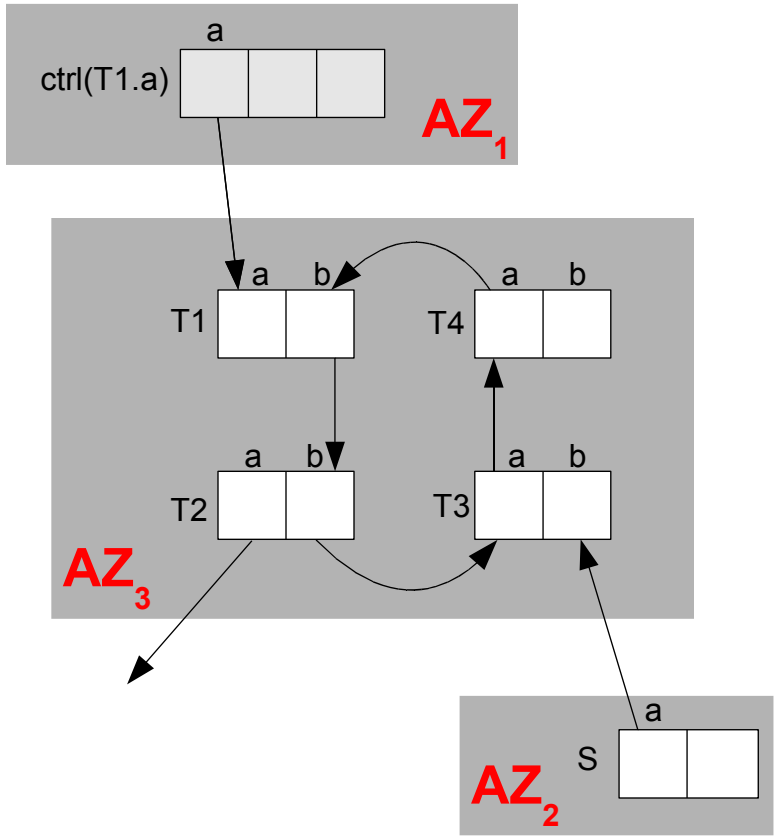
Nachteile



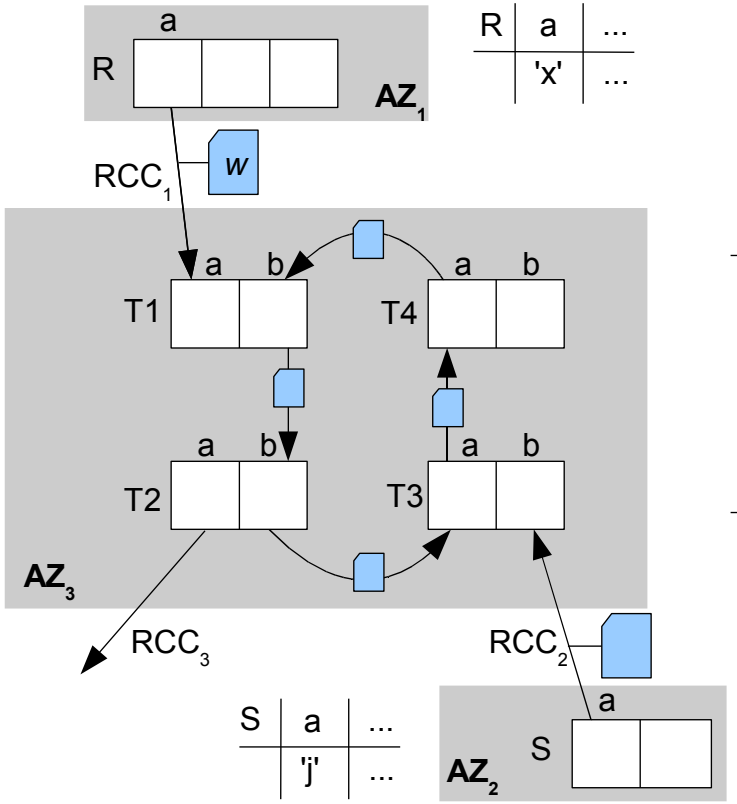
- ✗ Indirektes Einladen
- ✗ Backend muss Cache-Group-Definitionen kennen
- ✗ Backend muss das Aufsammeln der Daten organisieren

Entladen

- interne RCCs
  - interne Abhängigkeiten
- externe RCCs
  - externe Abhängigkeiten
- Entladen (top-down):
  - AZ_1, AZ_2, AZ_3



Entladen
homogener Zyklus



R	a	...
	'x'	...

S	a	...
	'j'	...

Anfänglicher Cache-Inhalt: AZ_3

T1	a	b
	'w'	1000
	'w'	2000
	'x'	2000
	'w'	3000

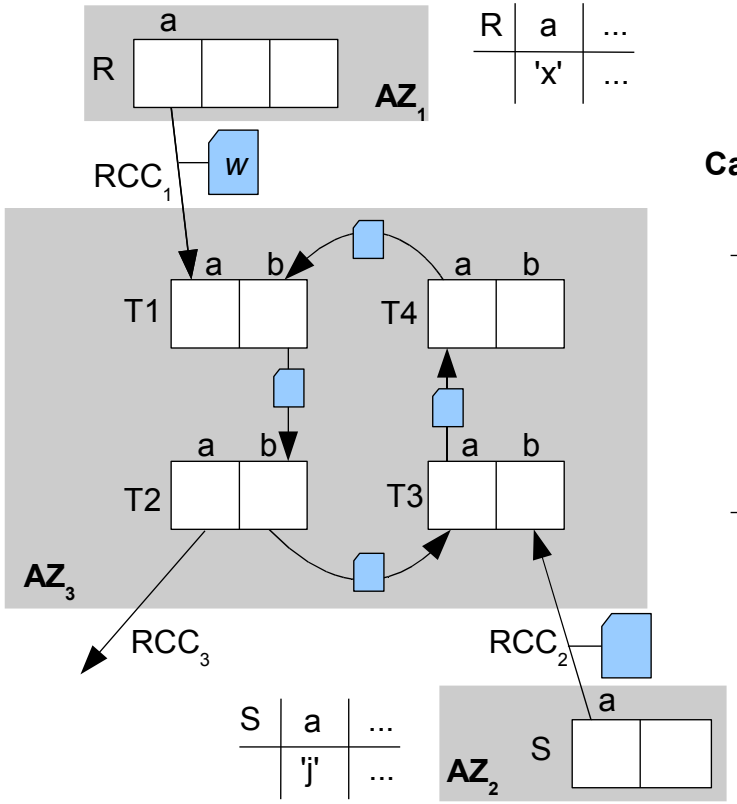
T4	a	b
	1000	...

T2	a	b
	11	1000
	47	3000

T3	a	b
	1000	'k'
	3000	'j'

Entladen

homogener Zyklus: globales Löschen



R	a	...
	'x'	...

S	a	...
	'j'	...

Cache-Inhalt nach globalem Löschen: AZ_3

T1	a	b
	~~'w'~~	~~1000~~
	'w'	2000
	'x'	2000
	'w'	3000

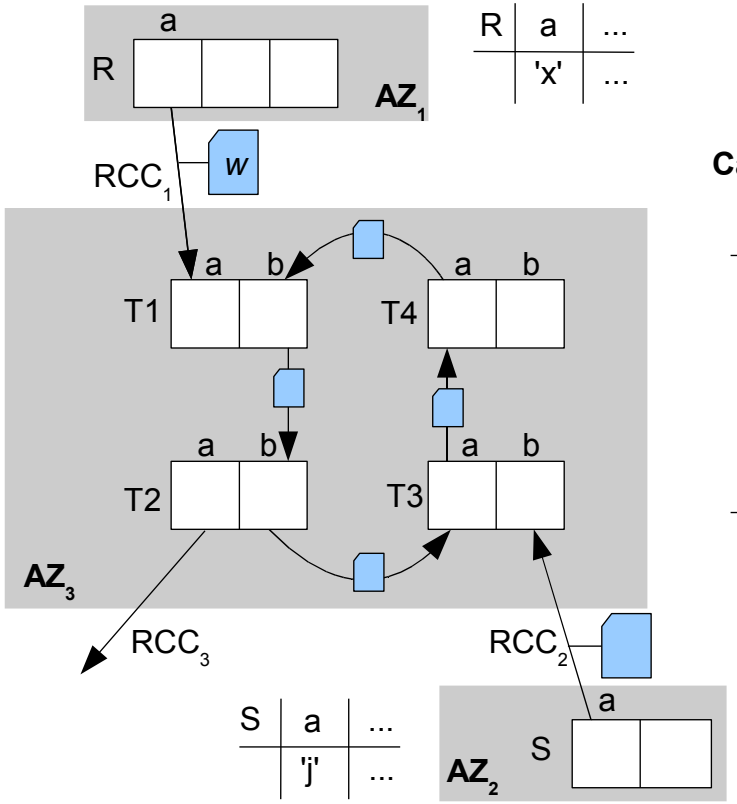
T4	a	b
	~~1000~~	...

T2	a	b
	~~11~~	~~1000~~
	47	3000

T3	a	b
	~~1000~~	~~'k'~~
	3000	'j'

Entladen

homogener Zyklus: internes Löschen



R	a	...
	'x'	...

S	a	...
	'j'	...

Cache-Inhalt nach internem Löschen: AZ3

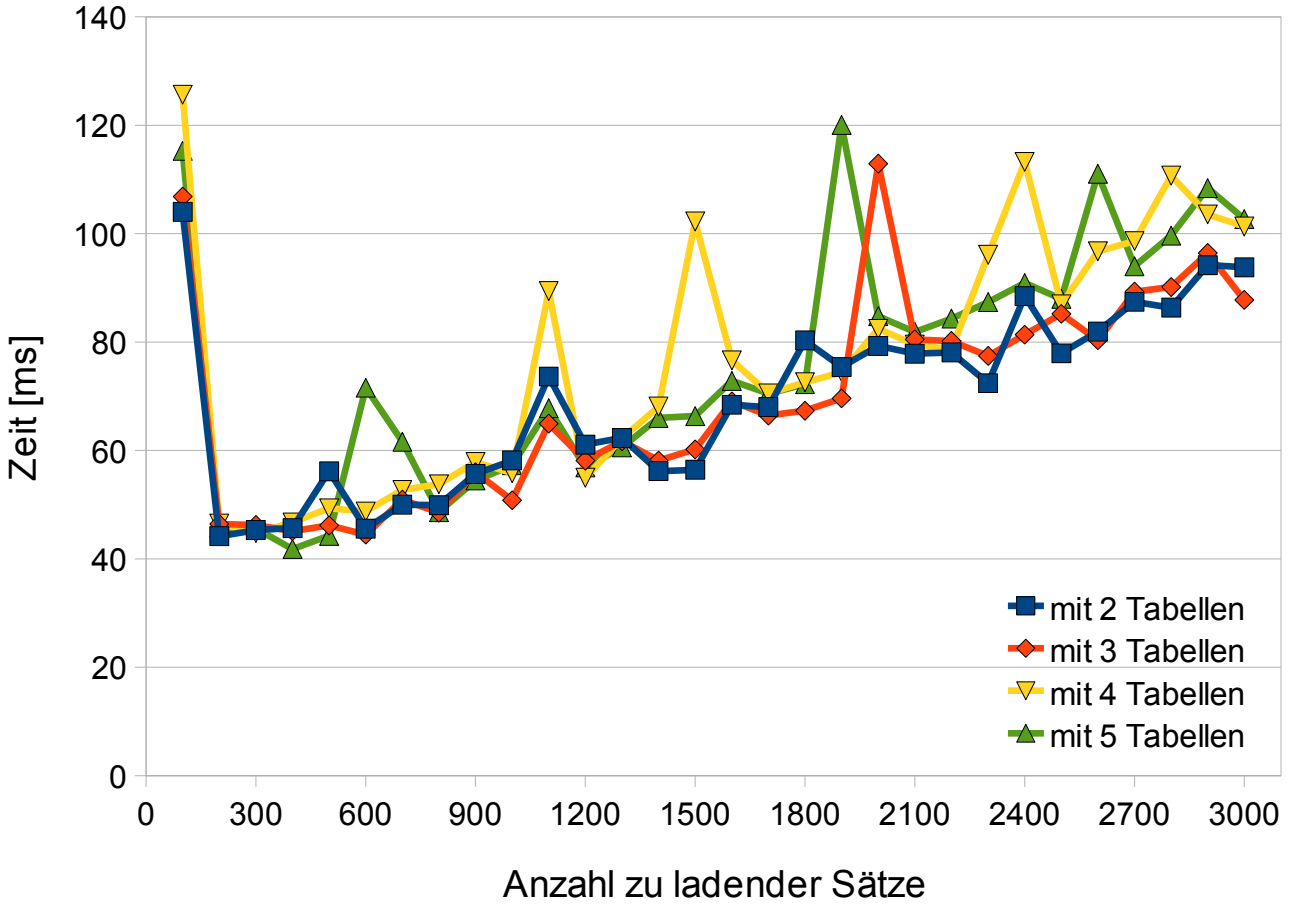
T1	a	b
	'w'	2000
	'x'	2000
	~~'w'~~	~~3000~~

T4	a	b

T2	a	b
	~~47~~	~~3000~~

T3	a	b
	3000	'j'

Ergebnisse
homogene Zyklen



Fazit

- Neue Verfahren
  - Indirektes Laden
  - Vorbereitetes Laden
  - Entladen in homogenen Zyklen
  - **Mehr physische Operatoren**
    → erhöhen Adaptivität
- Verbesserte Performance führt zu
  - erhöhter Ausnutzbarkeit vorhandener Lokalität

Danke, für Ihre Aufmerksamkeit

Fragen?