

Physische Datenorganisation

- Speicherhierarchie
- Hintergrundspeicher / RAID
- B-Bäume
- Hashing
- (R-Bäume)
- Objektballung
- Indexe in SQL



Hashing

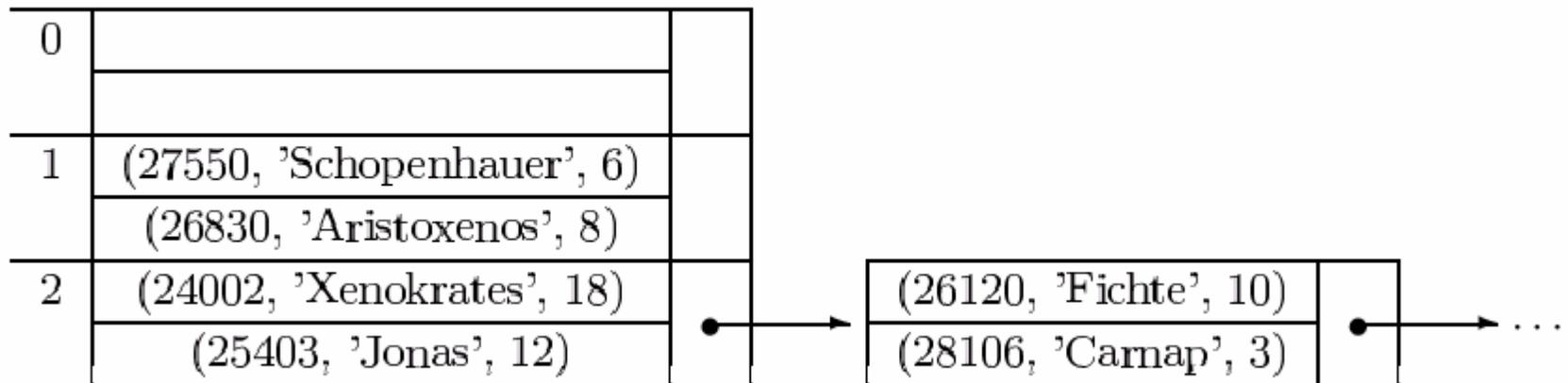
- B-Bäume haben den Vorteil, dass sie Bereichsanfragen und Sortierungen unterstützen.
- Aufwand: $O(\log n)$
- Hashing ist vom Aufwand $O(1)$ für das Suchen einzelner Tupel, unterstützt aber keine Sortierung.

Statistisches Hashing

- Hashfunktion $h(x) = x \bmod 3$

0	
1	(27550, 'Schopenhauer', 6)
2	(24002, 'Xenokrates', 18)
	(25403, 'Jonas', 12)

- Kollisionsbehandlung durch Überlaufseiten



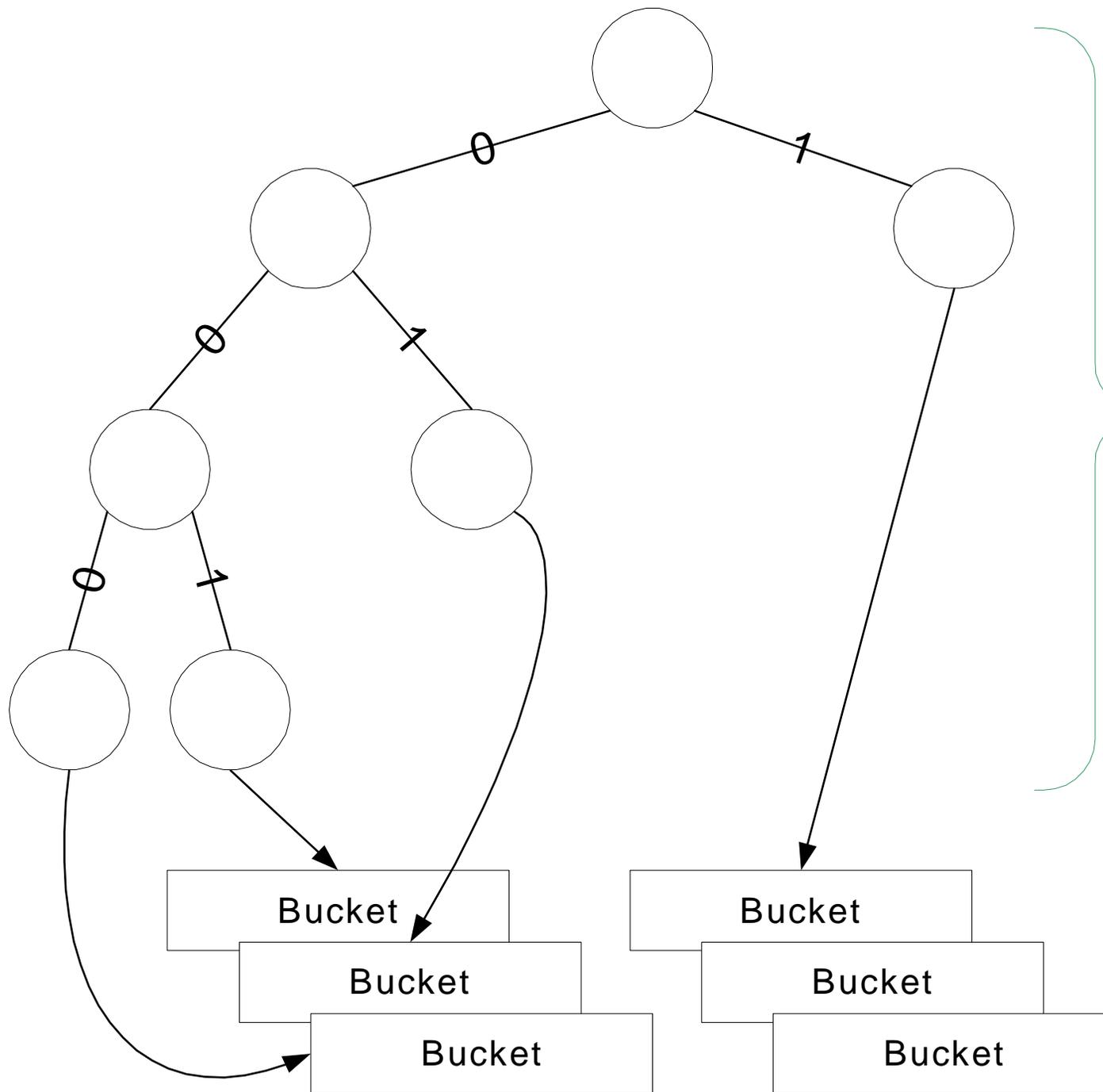
⇒ ineffizient bei nicht vorhersehbarer Datenmenge

Statisches Hashing

- Einfacher Ansatz:
 - Hashfunktion $h(x) = x \bmod N$, wobei N die Größe der Tabelle in Seiten ist.
 - $h(x)$ gibt die Seite an, auf der der Eintrag zu Schlüssel x zu finden ist.
- Probleme:
 - À priori Allokation des Speichers
 - Nachträgliche Vergrößerung der Hashtabelle ist „teuer“
 - Hashfunktion $h(\dots) = \dots \bmod N$
 - Rehashing der Einträge
 - $h(\dots) = \dots \bmod M$
 - In Datenbank Anwendungen viele GB

Erweiterbares Hashing

- Zusätzliche Indirektion über ein Directory
- Ein zusätzlicher Zugriff auf ein Directory, das den Zeiger (Verweis, BlockNr) des Hash-Bucket enthält
- Dynamisches Wachsen (und Schrumpfen) ist möglich
- Der Zugriff auf das Directory erfolgt über einen binären Hashcode



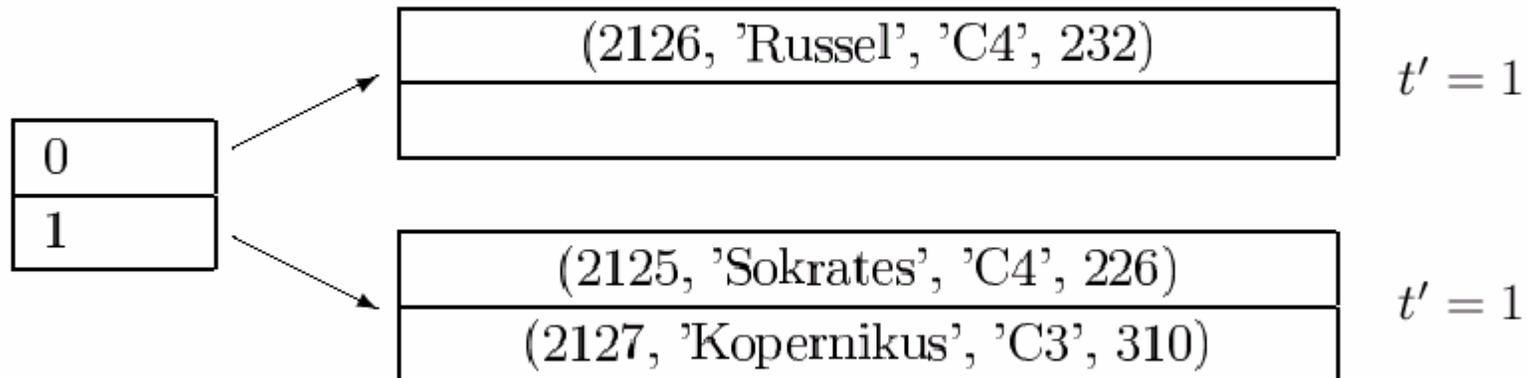
binärer
Trie,
Entschei-
dungs-
baum
Directory

Hashfunktion für erweiterbares Hashing

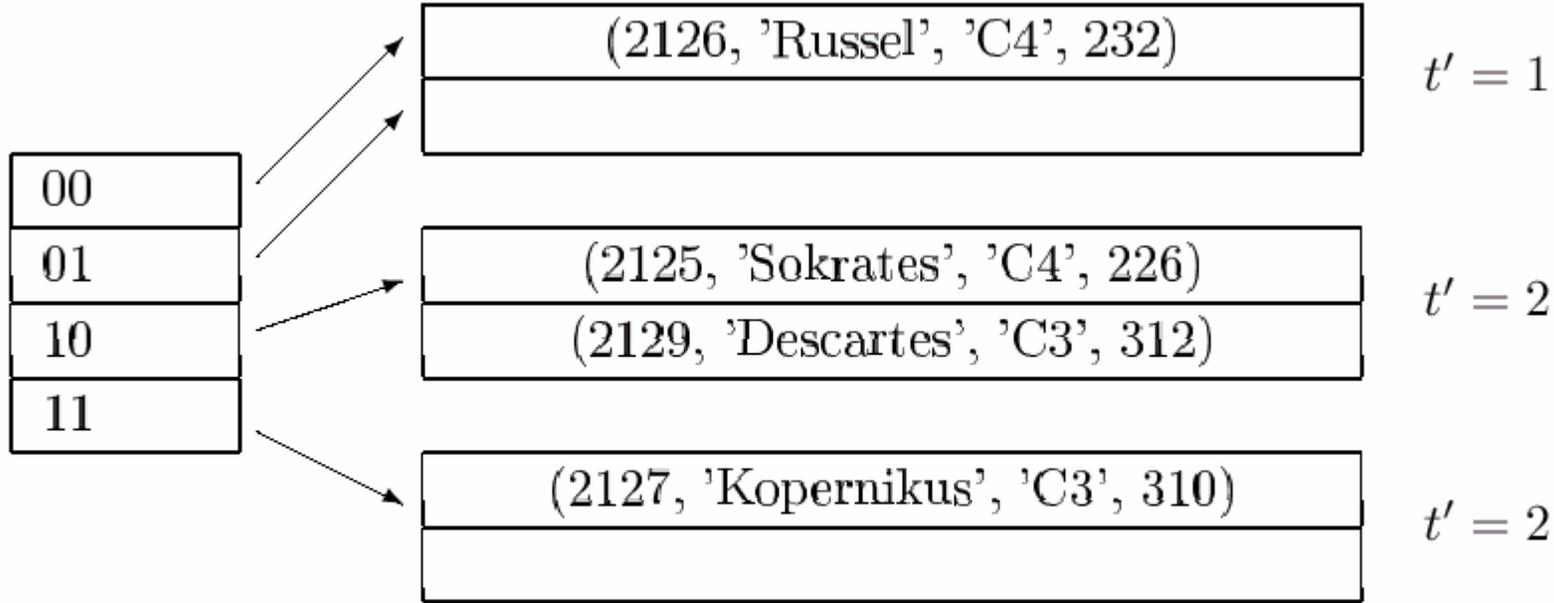
- h : Schlüsselmenge $\rightarrow \{0,1\}^*$
- Der Bitstring muss lang genug sein, um alle Objekte auf ihre Buckets abbilden zu können
- Anfangs wird nur ein (kurzer) Präfix des Hashwertes (Bitstrings) benötigt
- Wenn die Hashtabelle wächst wird aber sukzessive ein längerer Präfix benötigt
- Beispiel-Hashfunktion: gespiegelte binäre PersNr
 - $h(004) = 001000000\dots$ (4=0..0100)
 - $h(006) = 011000000\dots$ (6=0..0110)
 - $h(007) = 111000000\dots$ (7 =0..0111)
 - $h(013) = 101100000\dots$ (13 =0..01101)
 - $h(018) = 010010000\dots$ (18 =0..010010)
 - $h(032) = 000001000\dots$ (32 =0..0100000)
 - $H(048) = 000011000\dots$ (48 =0..0110000)

Demonstration des erweiterbaren Hashings

x	$h(x)$	
	d	p
2125	1	01100100001
2126	0	11100100001
2127	1	11100100001

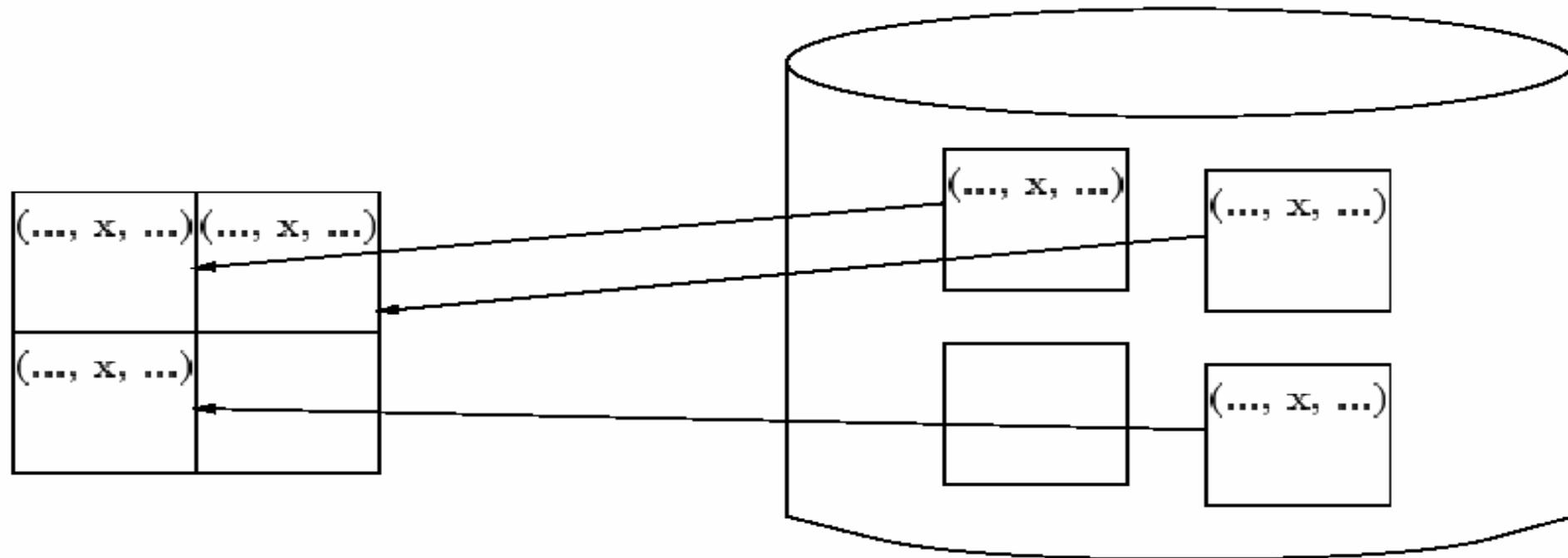


x	$h(x)$	
	d	p
2125	10	1100100001
2126	01	1100100001
2127	11	1100100001
2129	10	0010100001



Objektballung / Clustering logisch verwandter Daten

```
select *  
from R  
where A = x;
```



Hauptspeicher

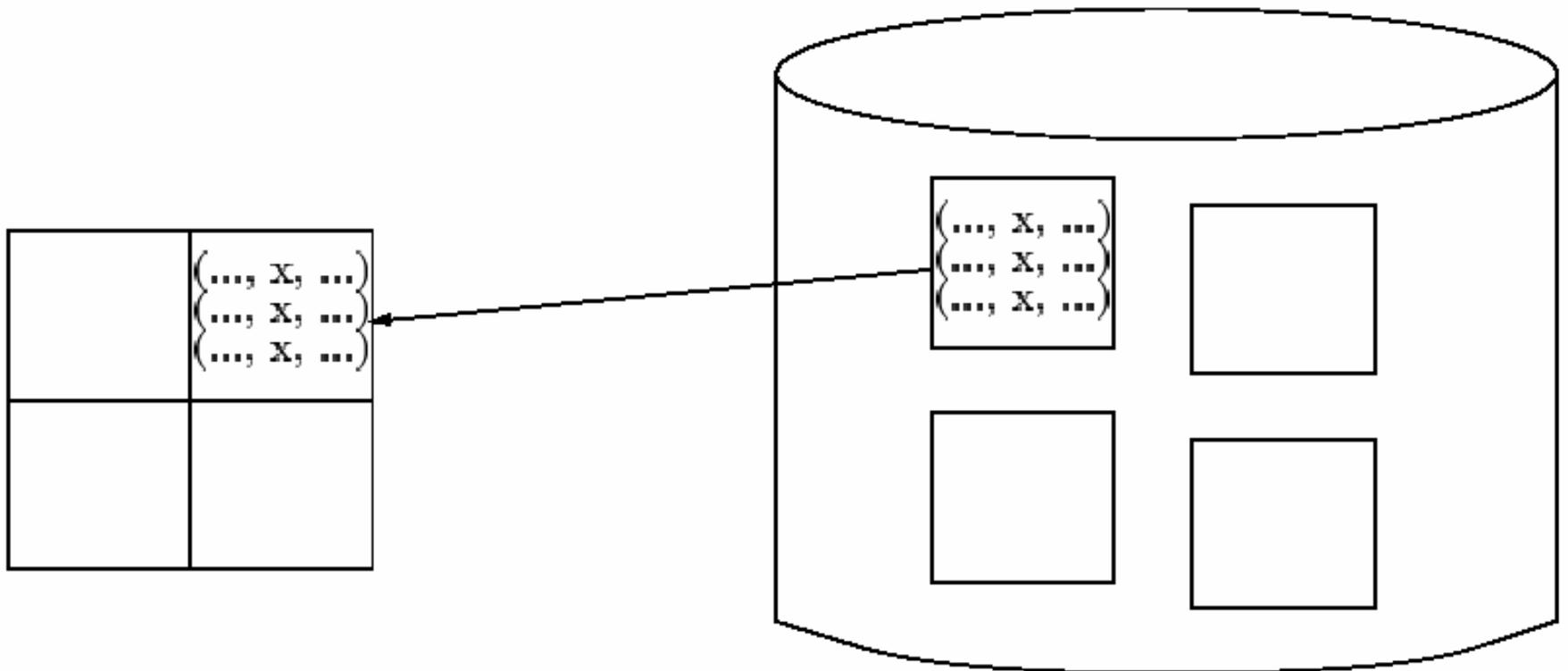
← Zugriffslücke →

Hintergrundspeicher

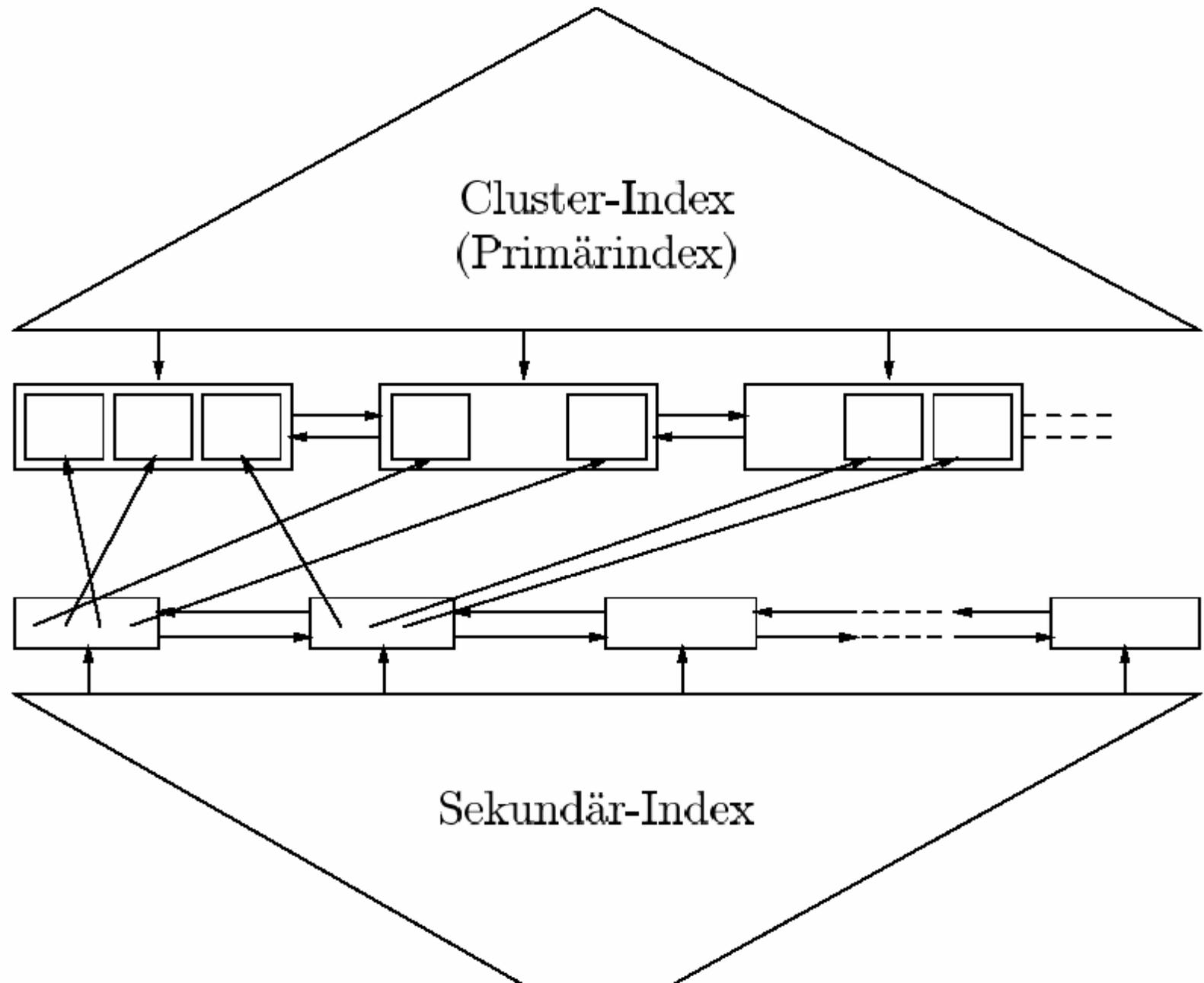
Hauptspeicher

← Zugriffslücke →

Hintergrundspeicher



Indexe und Ballung



Seite P_i

2125	o Sokrates	o C4	o 226	•
5041	o Ethik	o 4	o 2125	•
5049	o Mäeutik	o 2	o 2125	•
4052	o Logik	o 4	o 2125	•
2126	o Russel	o C4	o 232	•
5043	o Erkenntnistheorie	o 3	o 2126	•
5052	o Wissenschaftstheorie	o 3	o 2126	•
5216	o Bioethik	o 2	o 2126	•

Seite P_{i+1}

2133	o Popper	o C3	o 52	•
5259	o Der Wiener Kreis	o 2	o 2133	•
2134	o Augustinus	o C3	o 309	•
5022	o Glaube und Wissen	o 2	o 2134	•
2137	o Kant	o C4	o 7	•
5001	o Grundzüge	o 4	o 2137	•
4630	o Die 3 Kritiken	o 4	o 2137	•

:

Unterstützung eines Anwendungsverhaltens

```
Select Name  
From Professoren  
Where PersNr = 2136
```

```
Select Name  
From Professoren  
Where Gehalt >= 90000 and Gehalt <= 100000
```

Indexe in SQL

```
Create index SemsterInd  
on Studenten  
(Semester)
```

```
drop index SemsterInd
```