

Mehrere Indexe auf den selben Daten

Primärindex - Sekundärindexe

Studenten		
MatrNr	Name	Semester
25403	Jonas	12
29120	Theophrastos	2
29555	Feuerbach	2
27550	Schopenhauer	6
⋮	⋮	⋮

Wann

- Index auf MatrNr?
- Index auf Name?
- Index auf Semester?

Sekundärindexe, Zusammengesetzte Indexe

Tabelle Gepäckstück (Id, Gewicht, Passagier-Name, Ziel)

Anfrage: Welches sind – ohne Duplikate - die Ziele aller Gepäckstücke mit einem Gewicht ≥ 50 kg?

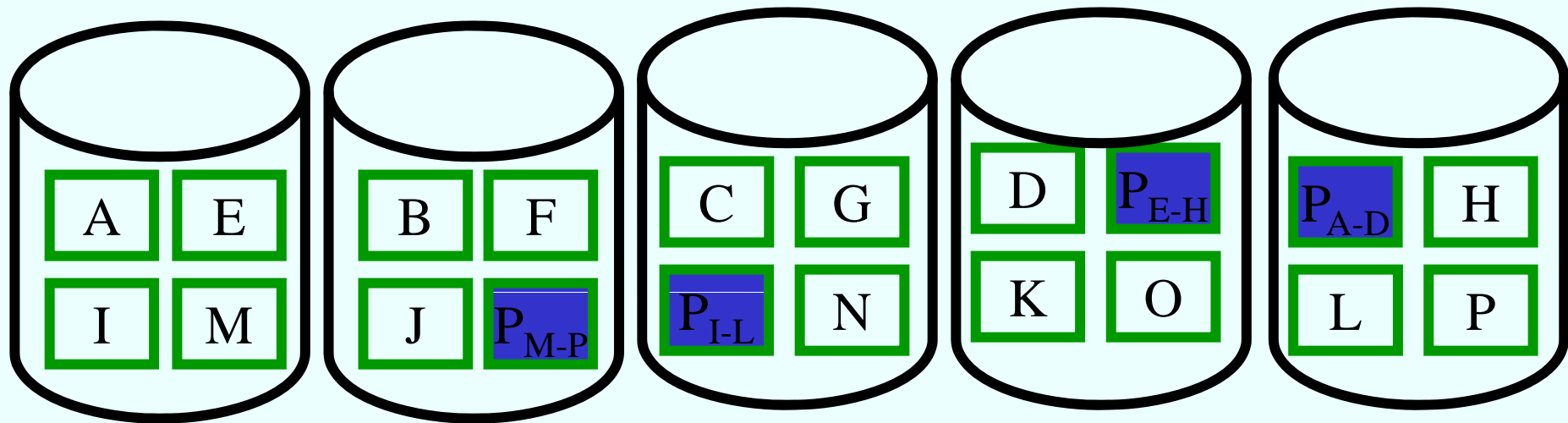
1. Hilft ein Sekundärindex über Gewicht zur Beschleunigung der Anfrage? Warum (nicht)?
2. Hilft ein Sekundärindex über Ziel zur Beschleunigung der Anfrage? Warum (nicht)?
3. Hilft ein Sekundärindex über (Gewicht, Ziel) zur Beschleunigung der Anfrage? Warum (nicht)?

RAIDs

- RAID: Redundant Array of Independent (Inexpensive) Disks
- Stichworte: Fehlertoleranz, Ausfallsicherheit, Verfügbarkeit, Datensicherheit, Durchsatz, Austausch im laufenden Betrieb
- Wichtigste RAID-Typen:
 - RAID 0: Striping (kein "echtes" RAID, da keine Redundanz)
 - RAID 1: Spiegelung
 - RAID 0+1: Striping und Spiegelung
 - RAID 3/4: Striping mit Parity-Platte für Fehlerkorrektur (unterschiedliche Stückelung)
 - RAID 5: Striping mit verteilter Parity
 - RAID 6: Striping mit doppelt verteilter Parity (für noch mehr Ausfallsicherheit)
 - weitere, z.B. RAID 10 (RAID 0 über mehrere RAID 1)

RAIDs, cont.

Beispiel: RAID 5



Neue Entwicklungen

- Hauptspeicher-Datenbanksysteme, z.B.
 - Times Ten (Oracle)
 - VoltDB (einige Datenbankforscher, open source)
 - Monet DB (CWI, Amsterdam, open source)
 - TREX (SAP)
 - HYPER (Informatik, TUM)
- Columns Store Datenbanksysteme, z.B.
 - C-Store / Vertica (HP)
 - Monet DB (CWI, Amsterdam, open source)
 - TREX (SAP)
 - HYPER (Informatik, TUM)

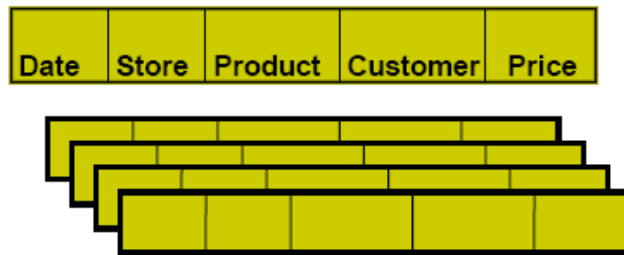
Column Stores

Re-use permitted when acknowledging the original © Stavros Harizopoulos, Daniel Abadi, Peter Boncz (2009)



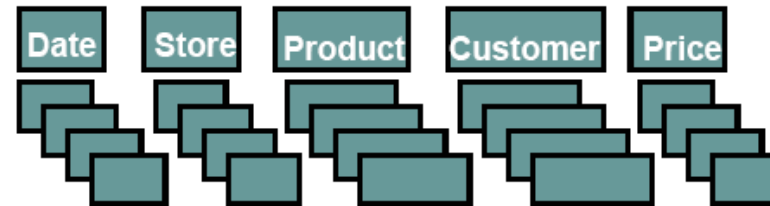
What is a column-store?

row-store



- + easy to add/modify a record
- might read in unnecessary data

column-store



- + only need to read in relevant data
- tuple writes require multiple accesses

=> *suitable for read-mostly, read-intensive, large data repositories*



Row Store versus Column Store

Verkäufe				
Produkt	Kunde	Preis	Filiale	...
Handy	Kemper	345	Schwabing	...
Radio	Mickey	123	Bogenhausen	...
Handy	Minnie	233	Schwabing	...
Kühlschrank	Urmel	240	Augsburg	...
Beamer	Bond	740	London	...
Handy	Lucie	321	Bogenhausen	...

Row Store versus Column Store

Produkt	
ID	Produkt
0	Handy
1	Radio
2	Handy
3	Kühlschrank
4	Beamer
5	Handy

Kunde	
ID	Kunde
0	Kemper
1	Mickey
2	Minnie
3	Urmel
4	Bond
5	Lucie

Preis	
ID	Preis
0	345
1	123
2	233
3	240
4	740
5	321

Filiale	
ID	Filiale
0	Schwabing
1	Bogenhausen
2	Schwabing
3	Augsburg
4	London
5	Bogenhausen

Komprimierung

Interessant sind insbesondere die Komprimierungsmöglichkeiten:

Dictionary	
ID	Wort
0	Augsburg
1	Beamer
2	Bogenhausen
3	Handy
4	Kühlschrank
5	London
6	Radio
7	Schwabing
...	...

Produkt	
ID	Produkt
0	3
1	6
2	3
3	4
4	1
5	3

Filiale	
ID	Filiale
0	7
1	2
2	7
3	0
4	5
5	2

Partitionierung

- Bäume brauchen im Schnitt $\log_k(n)$ Seitenzugriffe, um ein Datenelement zu lesen
(k =Verzweigungsgrad, n =Anzahl indexierter Datensätze)
 - Hashtabellen (partitionierende Verfahren) brauchen im Schnitt zwei Seitenzugriffe
- warum B-Bäume und nicht Hashing?

Was ist Hashing?

- to hash = zerhacken
- Speicherung der Tupel in einem festgelegten Speicherbereich
- Hashfunktion: Abbildung von Tupeln (Schlüsselwerte) in eine festgelegte Menge von Funktionswerten
- optimale Hashfunktion:
 - injektiv (keine gleichen Funktionswerte für unterschiedliche Argumente)
 - surjektiv (kein Speicherverschnitt)
- typische Hashfunktion $h: h(x) = x \bmod N$
Menge von Funktionswerten somit $\{0, \dots, N-1\}$

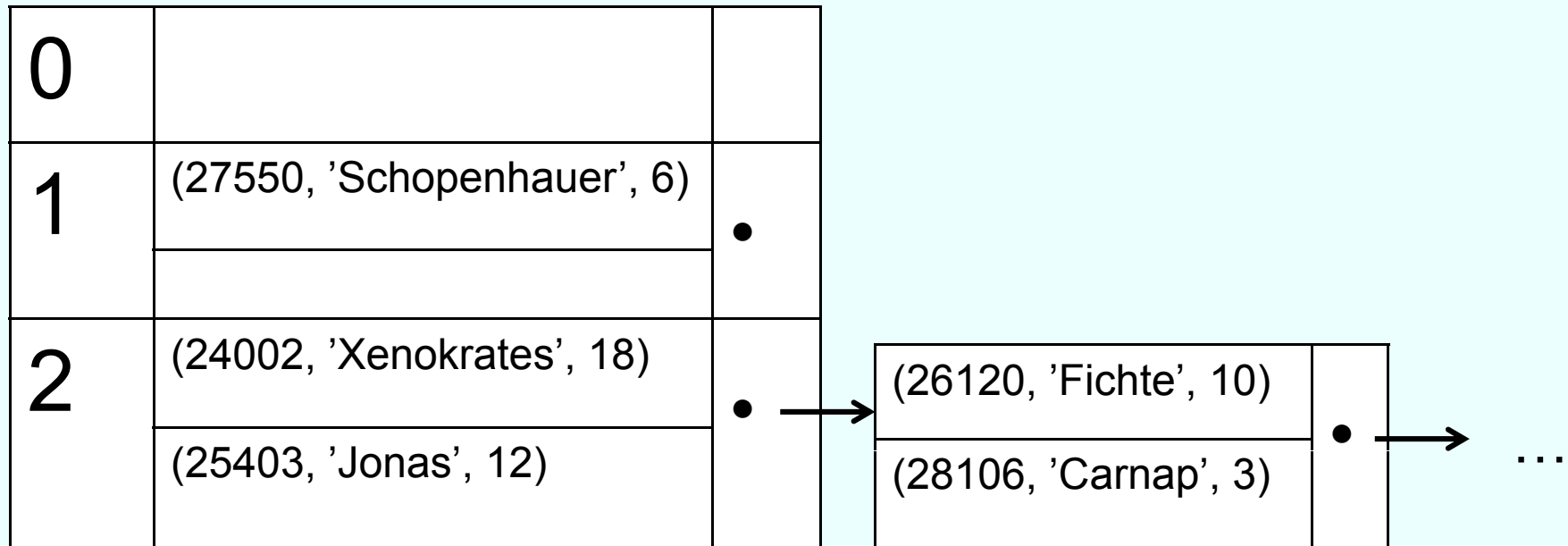
Beispiel Hashing

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

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

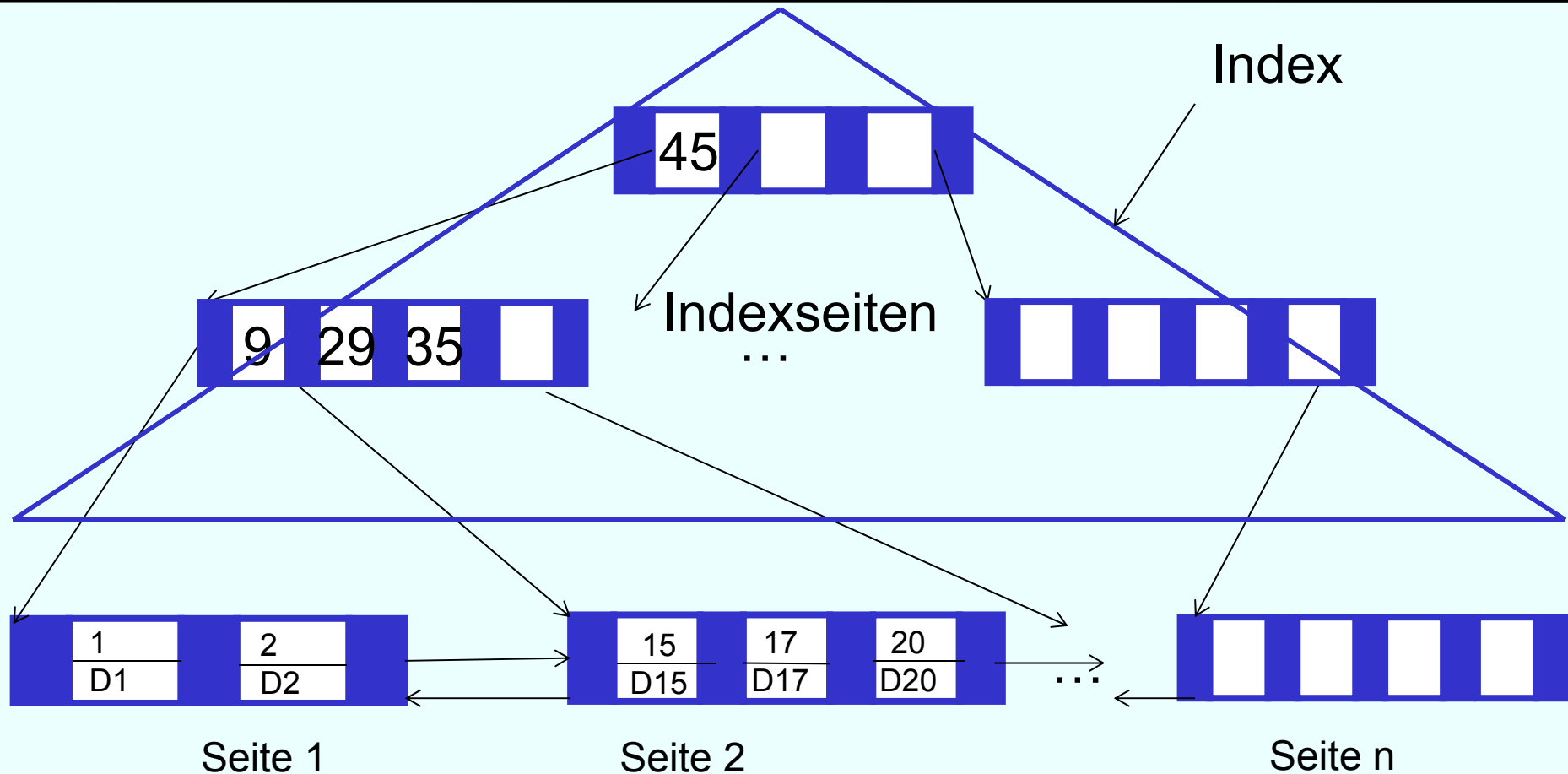
Kollisionen

- Kollisionsbehandlung



- Ineffizient bei nicht vorhersehbarer Datenmenge
- Ausweg: erweiterbares (dynamisches) Hashing
→ zusätzliche Indirektion über Directory

Struktur B*-Baum



Datenseiten, sortiert,
doppelt verkettet

Indexe und Ballung

