

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?

Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

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)?

Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

RAIDs

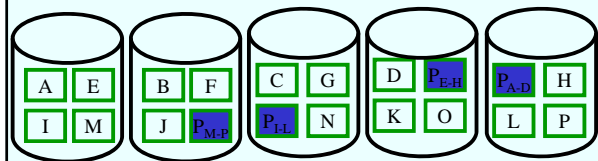
- RAID: Redundant Array of Independent (Inexpensive) Disks
- Stichworte: Fehlertoleranz, Ausfallsicherheit, Verfügbarkeit, Datensicherheit, Durchsatz, Austausch im laufenden Betrieb
- Wichtigste RAID-Typen:
 - RAID 0: Striping (kein "echtes" RAID, da keine Redundanz)
 - RAID 1: Spiegelung
 - RAID 0+1: 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)

Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

RAIDs, cont.

Beispiel: RAID 5



Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

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)

Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

Column Stores

Re-use permitted when acknowledging the original © Steven Hertzogovics, Daniel Ahoak, Peter Boncz (2006)

What is a column-store?



- + easy to add/modify a record
 - + only need to read in relevant data
 - might read in unnecessary data
 - tuple writes require multiple accesses
- => suitable for read-mostly, read-intensive, large data repositories

VLDB 2009 Tutorial

Column-Oriented Database Systems

2

Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

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	...

Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

Row Store versus Column Store

Produkt		Kunde		Preis		Filiale	
ID	Produkt	ID	Kunde	ID	Preis	ID	Filiale
0	Handy	0	Kemper	0	345	0	Schwabing
1	Radio	1	Mickey	1	123	1	Bogenhausen
2	Handy	2	Minnie	2	233	2	Schwabing
3	Kühlschrank	3	Urmel	3	240	3	Augsburg
4	Beamer	4	Bond	4	740	4	London
5	Handy	5	Lucie	5	321	5	Bogenhausen

Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

Komprimierung

Interessant sind insbesondere die Komprimierungsmöglichkeiten:

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

Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

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?

Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

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 $\{0, \dots, N-1\}$

Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

Beispiel Hashing

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

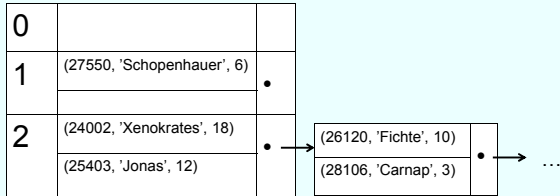
0	
1	(27550, 'Schopenhauer', 6)
2	(24002, 'Xenokrates', 18) (25403, 'Jonas', 12)

Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

Kollisionen

- Kollisionsbehandlung

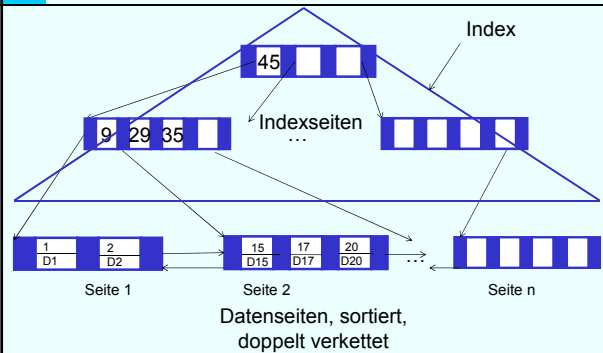


- Ineffizient bei nicht vorhersehbarer Datenmenge
- Ausweg: erweiterbares (dynamisches) Hashing
→ zusätzliche Indirektion über Directory

Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

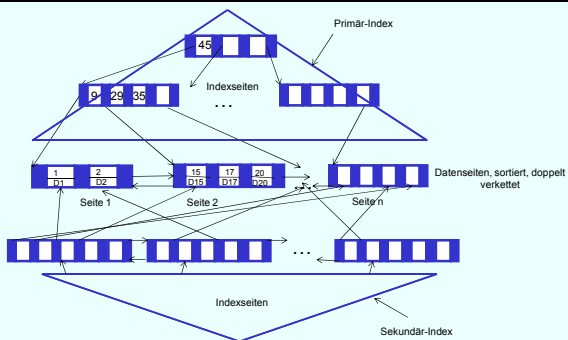
Struktur B*-Baum



Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012

Indexe und Ballung



Datenbanksysteme für Hörer anderer Fachrichtungen WS 2011/2012

11.01.2012