



**Hasso
Plattner
Institut**

IT Systems Engineering | Universität Potsdam



Index Compression Search Engines – SS 2009

Sebastian Klose

Vitaliy Kats

Strategien

2

- Delta Encoding oder Gap Encoding
 - 1000, 1001, 1002, 1003 → 1000, 1, 2, 3
- Variable Byte Encoding
 - 7 Bit je Byte für Zahl, 1 Bit Continuation Bit
- Block Compression

Block Compression

3

...7systile9syzygetic8syzygial6syzygy11szaibelyite6szecin...

freq.	postings ptr.	term ptr.
9	→	
92	→	
5	→	
71	→	
12	→	
...

- Blöcke á 1024 Byte
- Blöcke werden mit 0x00 aufgefüllt

Quelle: Introduction to Information Retrieval

Dateiformat

4

- Zwei Dateien
 - terms.dat
 - postings.dat

terms.dat

5



postings.dat

6



Blocklänge | Anzahl DocIDs | DocID 1 | DocID 2 | Anzahl DocIDs | DocID 1 | ...

Suche

7

1. Binäre Suche in terms.dat um BlockIndex zu finden
2. Lineare Suche um TermIndex im Block zu finden
3. Block #BlockIndex aus postings.dat laden
4. #TermIndex'te DocumentID Liste finden

- postings.dat wird beim Starten gescannt
 - Offset zu jedem BlockIndex zwischenspeichern
- Einmal gelesene Blöcke (aus beiden Dateien) werden in Dictionary gespeichert
- Naives Caching
 - Wenn das Dictionary voll ist, ändert sich der Cache nicht mehr
 - Für binäre Suche gut, da diese unabhängig vom gesuchten Term mit den gleichen Blöcken startet

	MySQL	Compressed Index
Daten	2 GB	900 MB
Index	183 MB	81 MB

> 50%

Performance

10

	MySQL	Compressed Index
"java" #1	78 ms	188 ms
"java" #2	16 ms	15 ms
"spider" #1	47 ms	78 ms
"spider" #2	16 ms	15 ms
"spider" #3	31 ms	15 ms
"crawler", #1	78 ms	579 ms
"crawler", #2	16 ms	0 ms
"crawler", #3	0 ms	0 ms

Platzverschwendung

11

- Padding in terms.dat kostet unnötigt Speicherplatz
- Messung
 - 81326 Blöcke insgesamt
 - 7803 voll (1024 Byte ausgenutzt)
 - 449.646 Bytes Padding insgesamt
 - ~0,5% Platzverschwendung