

# **B+-TREE COSTS**

# Recall: Cost of Operations



- **Can we do better with indexes?**
- **B:** Number of data blocks
- **R:** Number of records per block
- **D:** Average time to read/write disk block

Recall we are interested in the average case

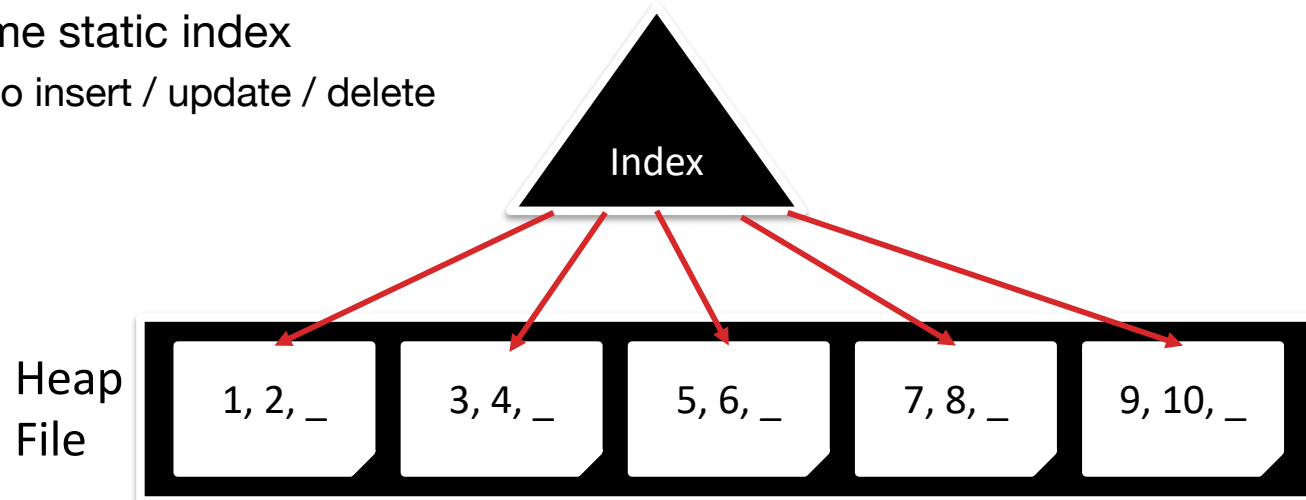
Both reading and writing to the disk cost I/Os!!

	Heap File	Sorted File
Scan all records	$B * D$	$B * D$
Equality Search	$0.5 * B * D$	$(\log_2 B) * D$
Range Search	$B * D$	$((\log_2 B) + \text{pages}) * D$
Insert	$2 * D$	$((\log_2 B) + B) * D$
Delete	$(0.5 * B + 1) * D$	$((\log_2 B) + B) * D$

# Index Assumptions



- Store data by reference (Alternative 2)
- **Clustered** index with 2/3 full heap file pages
  - Clustered → Heap file is initially sorted
  - **Fan-out** (F) (i.e., branching factor) of tree internal node:
    - Page of  $\langle \text{key}, \text{pointer} \rangle$  pairs  $\sim O(R)$  [R: Number of records per block]
    - in practice this is relatively large. Why?
  - Assume static index
    - No insert / update / delete



# Cost of Operations



- **Can we do better with indexes?**
- **B:** Number of data blocks
- **R:** Number of records per block
- **D:** Average time to read/write disk block

	Heap File	Sorted File	Clustered Index
Scan all records	$B * D$	$B * D$	
Equality Search	$0.5 * B * D$	$(\log_2 B) * D$	
Range Search	$B * D$	$((\log_2 B) + \text{pages}) * D$	
Insert	$2 * D$	$((\log_2 B) + B) * D$	
Delete	$(0.5 * B + 1) * D$	$((\log_2 B) + B) * D$	

# Scan all the Records

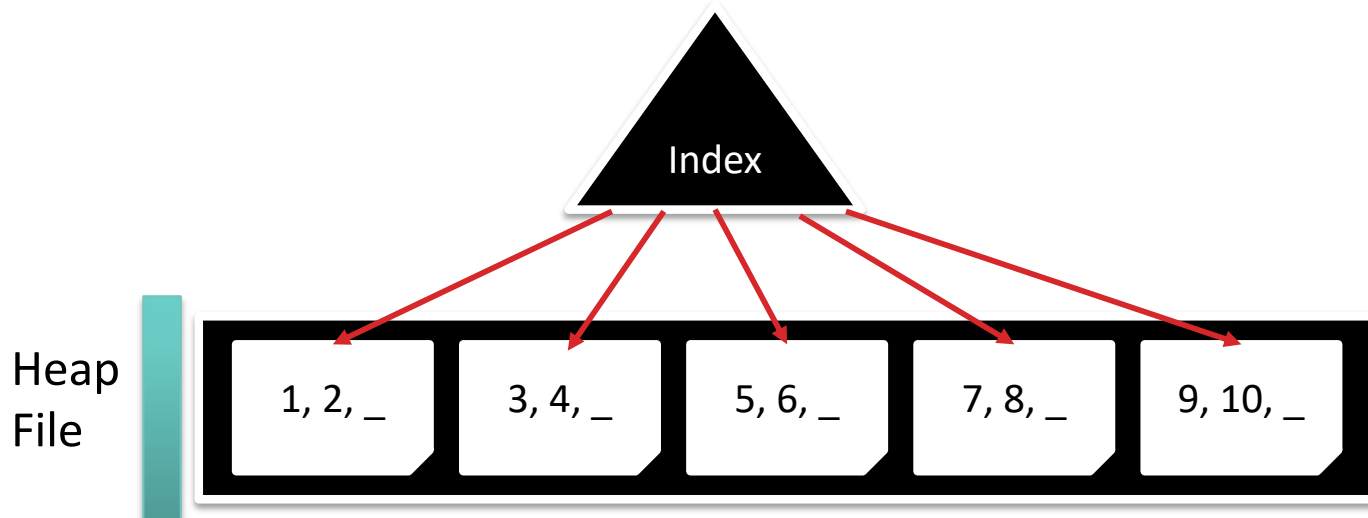


- Do we need an Index?
  - No
- Cost? =  $1.5 * B * D$ 
  - Why?

**B:** Number of data blocks

**D:** Average time to read/write disk block

Recall assumption on clustered indexes:  
heap file pages are only **2/3** full.



# Cost of Operations: Scan



	Heap File	Sorted File	Clustered Index
Scan all records	$B * D$	$B * D$	$3/2 * B * D$
Equality Search	$0.5 * B * D$	$(\log_2 B) * D$	
Range Search	$B * D$	$((\log_2 B) + \text{pages}) * D$	
Insert	$2 * D$	$((\log_2 B) + B) * D$	
Delete	$(0.5 * B + 1) * D$	$((\log_2 B) + B) * D$	

- **B:** Number of data blocks
- **R:** Number of records per block
- **D:** Average time to read/write disk block

# Cost of Operations: Equality Search?



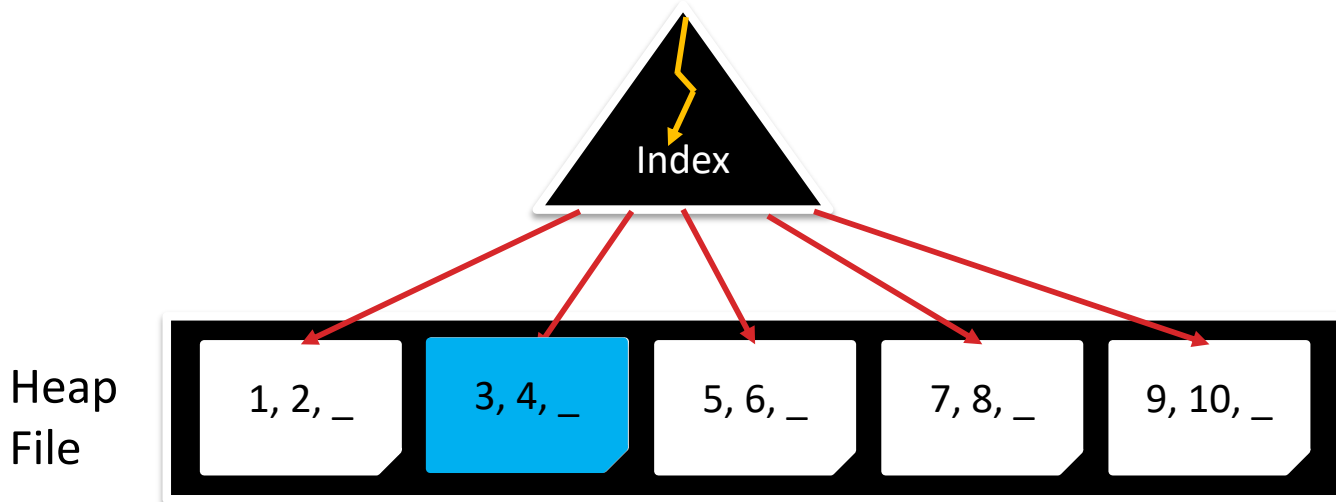
	Heap File	Sorted File	Clustered Index
Scan all records	$B * D$	$B * D$	$3/2 * B * D$
Equality Search	$0.5 * B * D$	$(\log_2 B) * D$	
Range Search	$B * D$	$((\log_2 B) + \text{pages}) * D$	
Insert	$2 * D$	$((\log_2 B) + B) * D$	
Delete	$(0.5 * B + 1) * D$	$((\log_2 B) + B) * D$	

- **B:** Number of data blocks
- **R:** Number of records per block
- **D:** Average time to read/write disk block
- **F:** Average internal node fanout
- **E:** Average # data entries per leaf

# Find the record with key 3



- Two steps
  1. Search index to find the page and record-id
  2. Fetch record-id from heap file

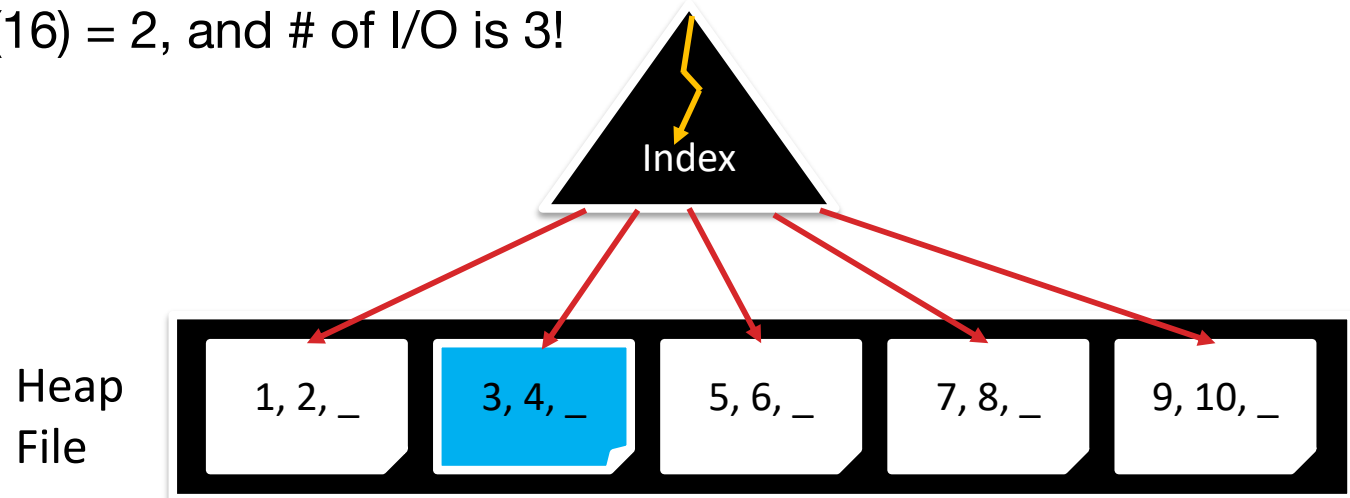




# Find the record with key 3



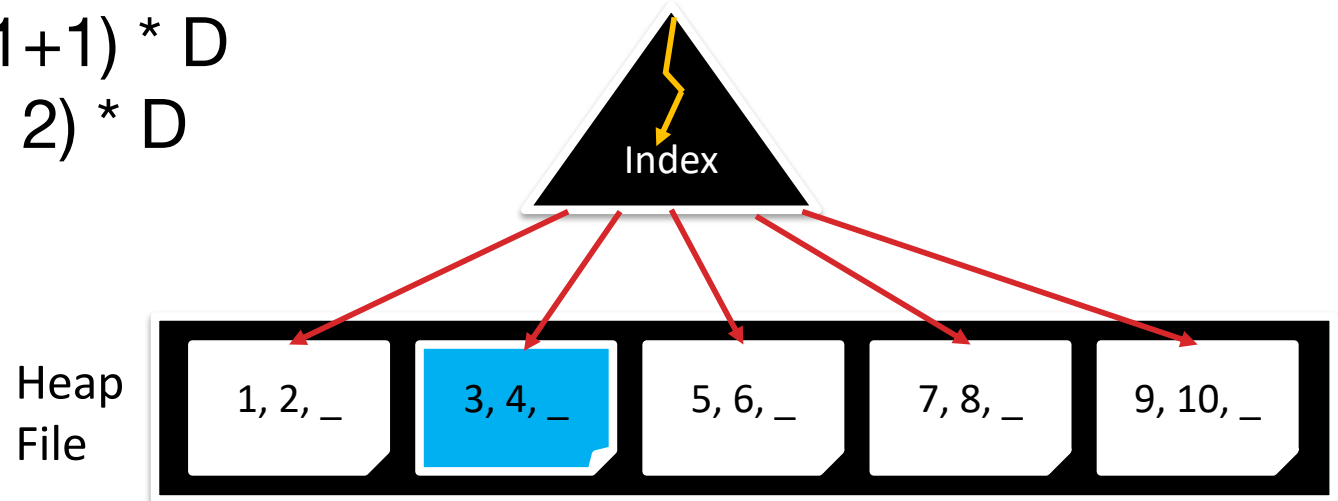
- I/Os for index search = height of index + 1
- $= \log_F (\# \text{ of leaves}) + 1 = \log_F (B \cdot R / E) + 1$ 
  - $B \cdot R$  is the total number of records;  $E$  is the #records per leaf
  - Why +1? Catches the cost of fetching the leaf from index
    - E.g.,  $F = 4$ ,  $BR/E = 16$ : root  $\rightarrow$  intermediate  $\rightarrow$  leaf
    - But  $\text{Log}_4(16) = 2$ , and # of I/O is 3!



# Find the record with key 3



- I/Os for lookup record in heap file by record-id: 1
  - Recall record-id = <page, slot #>
- Total cost: (# of I/Os) \* D
  - = (I/Os for index search + I/Os for heap file read) \* D
  - =  $(\log_F (BR/E) + 1 + 1) * D$
  - =  $(\log_F (BR/E) + 2) * D$



# Cost of Operations: Equality Search



	Heap File	Sorted File	Clustered Index
Scan all records	$B * D$	$B * D$	$3/2 * B * D$
Equality Search	$0.5 * B * D$	$(\log_2 B) * D$	$(\log_F(BR/E) + 2) * D$
Range Search	$B * D$	$((\log_2 B) + \text{pages}) * D$	
Insert	$2 * D$	$((\log_2 B) + B) * D$	
Delete	$(0.5 * B + 1) * D$	$((\log_2 B) + B) * D$	

- **B:** Number of data blocks
- **R:** Number of records per block
- **D:** Average time to read/write disk block
- **F:** Average internal node fanout
- **E:** Average # data entries per leaf

# Cost of Operations: Range Search?



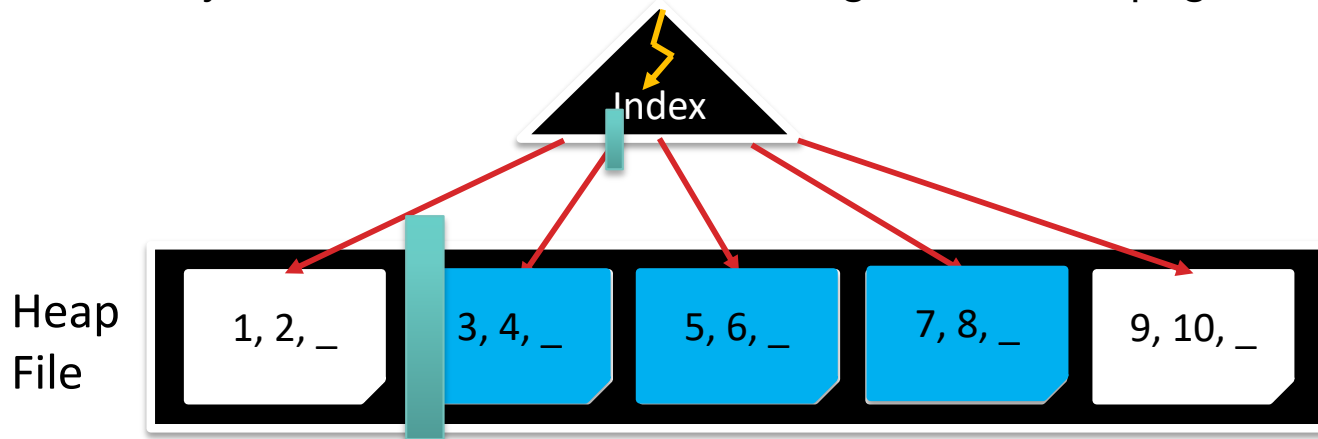
	Heap File	Sorted File	Clustered Index
Scan all records	$B * D$	$B * D$	$3/2 * B * D$
Equality Search	$0.5 * B * D$	$(\log_2 B) * D$	$(\log_F(BR/E) + 2) * D$
Range Search	$B * D$	$((\log_2 B) + \text{pages}) * D$	
Insert	$2 * D$	$((\log_2 B) + B) * D$	
Delete	$(0.5 * B + 1) * D$	$((\log_2 B) + B) * D$	

- **B:** Number of data blocks
- **R:** Number of records per block
- **D:** Average time to read/write disk block
- **F:** Average internal node fanout
- **E:** Average # data entries per leaf

# Find records with keys between 3 and 7



- Three steps:
  1. Search index to find first page to read
  2. Scan index leaf pages to find which heap file pages to read
  3. Read the corresponding records from heap file
- I/Os for 1:  $\log_F (B \cdot R / E) + 1$  [+1 for the index leaf page]
- I/Os for 3:  $(3/2 * \text{\#pages})$  [#pages: # of pages storing records between 3 and 7]
- I/Os for 2:  $(3/2 * \text{\#pages})$  [over-approximate and assume same as 2]
- Total cost :  $((\log_F (B \cdot R / E) + 1) + 2 * (3/2 * \text{\#pages}) - 1) * D = (\log_F (B \cdot R / E) + 3 * \text{\# pages}) * D$ 
  - Why -1? We overcounted accessing the first leaf page in the index



# Cost of Operations: Range Search



	Heap File	Sorted File	Clustered Index
Scan all records	$B * D$	$B * D$	$3/2 * B * D$
Equality Search	$0.5 * B * D$	$(\log_2 B) * D$	$(\log_F(BR/E) + 2) * D$
Range Search	$B * D$	$((\log_2 B) + \text{pages}) * D$	$(\log_F(BR/E) + 3 * \text{pages}) * D$
Insert	$2 * D$	$((\log_2 B) + B) * D$	
Delete	$(0.5 * B + 1) * D$	$((\log_2 B) + B) * D$	

- **B:** Number of data blocks
- **R:** Number of records per block
- **D:** Average time to read/write disk block
- **F:** Average internal node fanout
- **E:** Average # data entries per leaf

# Cost of Operations: Insert?



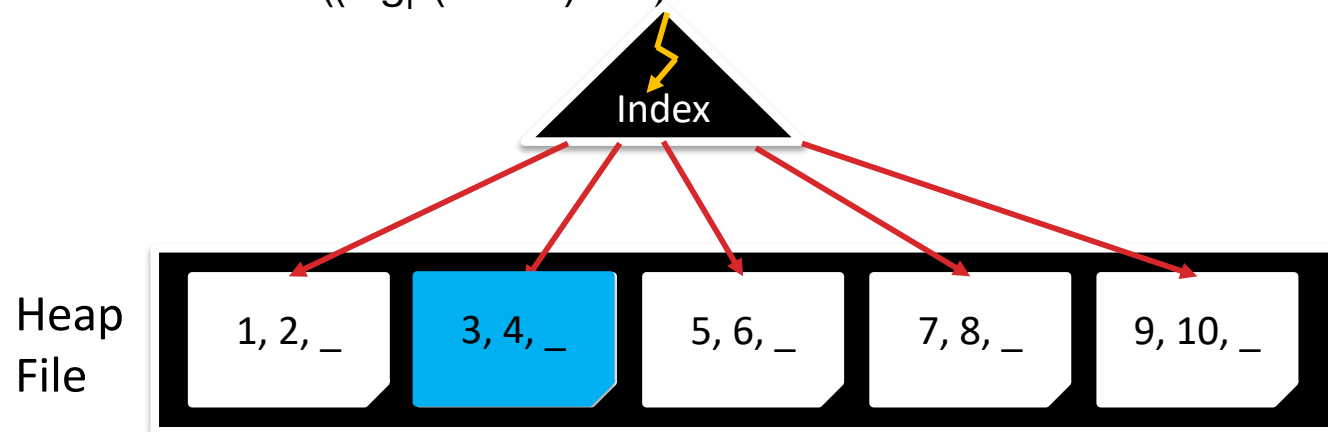
	Heap File	Sorted File	Clustered Index
Scan all records	$B * D$	$B * D$	$3/2 * B * D$
Equality Search	$0.5 * B * D$	$(\log_2 B) * D$	$(\log_F(BR/E) + 2) * D$
Range Search	$B * D$	$((\log_2 B) + \text{pages}) * D$	$(\log_F(BR/E) + 3 * \text{pages}) * D$
Insert	$2 * D$	$((\log_2 B) + B) * D$	
Delete	$(0.5 * B + 1) * D$	$((\log_2 B) + B) * D$	

- **B:** Number of data blocks
- **R:** Number of records per block
- **D:** Average time to read/write disk block
- **F:** Average internal node fanout
- **E:** Average # data entries per leaf



# Insert record with key 4.5

- Three steps:
  1. Search index to find heap file page to modify
  2. Read corresponding page and modify
  3. Write back the modified index leaf and heap file pages
- I/Os for 1:  $\log_F (B \cdot R / E) + 1$
- I/Os for 2: 1
- I/Os for 3: 2 [1 for index leaf, 1 for heap file page]
- Total cost :  $((\log_F (B \cdot R / E) + 4) * D$





# Cost of Operations: Insert



	Heap File	Sorted File	Clustered Index
Scan all records	$B * D$	$B * D$	$3/2 * B * D$
Equality Search	$0.5 * B * D$	$(\log_2 B) * D$	$(\log_F(BR/E) + 2) * D$
Range Search	$B * D$	$((\log_2 B) + \text{pages}) * D$	$(\log_F(BR/E) + 3 * \text{pages}) * D$
Insert	$2 * D$	$((\log_2 B) + B) * D$	$(\log_F(BR/E) + 4) * D$
Delete	$(0.5 * B + 1) * D$	$((\log_2 B) + B) * D$	

- **B:** Number of data blocks
- **R:** Number of records per block
- **D:** Average time to read/write disk block
- **F:** Average internal node fanout
- **E:** Average # data entries per leaf

# Cost of Operations: Delete



	Heap File	Sorted File	Clustered Index
Scan all records	$B * D$	$B * D$	$3/2 * B * D$
Equality Search	$0.5 * B * D$	$(\log_2 B) * D$	$(\log_F(BR/E) + 2) * D$
Range Search	$B * D$	$((\log_2 B) + \text{pages}) * D$	$(\log_F(BR/E) + 3 * \text{pages}) * D$
Insert	$2 * D$	$((\log_2 B) + B) * D$	$(\log_F(BR/E) + 4) * D$
Delete	$(0.5 * B + 1) * D$	$((\log_2 B) + B) * D$	$(\log_F(BR/E) + 4) * D$

- **B:** Number of data blocks
- **R:** Number of records per block
- **D:** Average time to read/write disk block
- **F:** Average internal node fanout
- **E:** Average # data entries per leaf

# Cost of Operations: Big O Notation



	Heap File	Sorted File	Clustered Index
Scan all records	$O(B)$	$O(B)$	$O(B)$
Equality Search	$O(B)$	$O(\log_2 B)$	$O(\log_F B)$
Range Search	$O(B)$	$O(\log_2 B)$	$O(\log_F B)$
Insert	$O(1)$	$O(B)$	$O(\log_F B)$
Delete	$O(B)$	$O(B)$	$O(\log_F B)$

- **B:** Number of data blocks
- **R:** Number of records per block
- **D:** Average time to read/write disk block
- **F:** Average internal node fanout
- **E:** Average # data entries per leaf

# Constant factors



- Assume you can do 100 sequential I/Os in the time of 1 random I/O
- For a particular lookup, is a B+-tree better than a full-table scan?
  - Better be very “selective”!
    - Visit  $< \sim 1\%$  of pages!
  - Two ways to make that happen:
    - Use a clustered index so that most reads are sequential
    - Use SSD so that random and sequential reads have the same cost

# Summary



- Query Structure
  - Understand composite search keys
  - Lexicographic order and search key prefixes
- Data Storage
  - Data Entries: Alt 1 (tuples), Alt 2 (recordIds), Alt 3 (lists of recordIds)
  - Clustered vs. Unclustered
    - Only Alt 2 & 3!

# Summary



- Variable length key refinements
  - Fill factors for variable-length keys
  - Prefix and suffix key compression
- B+-tree Cost Model
  - Attractive big-O
  - But don't forget constant factors of random I/O
    - Hard to beat sequential I/O of scans unless very selective
  - Indexes beyond B+-trees for more complex searches