Physical Level Issues
Slide Examples

Example 1, Slide 49
Compare the speed of retrieval for a file with r=30,000 records of fixed length R=100B, given block size of 1024B and V=9B, P=6B, in case of

- **an ordered file**
- **a file with a primary index**
- **a file with a secondary index**

Base information assumed:
r (number of records) = 30,000
BS (Block Size in bytes) = 1024
R (record size in bytes) – 100
V (size of index field data, in bytes) = 9
P (size of a pointer reference to a block or record, in bytes) = 6
Also assume unspanned blocks (a record must be stored in one block)

1a) Ordered File, Linear Search
Calculate BFR (blocking factor)
    1024 / 100 = 10.24, but as unspanned, can only store 10 records/block
    So BFR = 10
Calculate number of data blocks in file
    30000 / 10 = 3000 data blocks in file
Best case for linear search: only need to read 1 data file block from disk
Worst case for linear search: need to read all 3000 data file blocks from disk
Average case for linear search: need to read 1500 blocks from disk

1b) Ordered File, Binary Search
<same BFR, number of data blocks in file>
Will need to access, on average, $\log_2(3000)$ = ~11.55
So, maximum number of data file blocks read will be 12
Note: insert and delete into ordered file are expensive; this cost is significant

2. Primary Index on Ordered File
Index is non-dense (one index entry for each data file block) and single level
So, need 3000 index entries, as need to point to 3000 data file blocks
Calculate size of index entry
    V = 9, P = 6, size of index entry = V + P = 15 bytes
Calculate BFR-I (blocking factor for index block)
    1024 / 15 = 68.266…, so truncated => 68 index entries / index block

Calculate number of index blocks needed

3000 index entries / 68 (BFR-I) = 44.117, round up to 45 blocks needed

Note: index will be kept sorted, with index values for anchor record for each data file block

So, to find an individual index entry using binary search on index, will average $\log_2 (45)$ = ~5.49 = maximum 6 index blocks read from disk to find index entry with pointer to desired data file record

Also need to read the 1 data file record with the full data

Need to do linear search on records within data block, but this is within memory and relatively fast

So, total maximum of 7 disk block reads for this primary index

Note: insert and delete on ordered data file are still expensive; also some overhead from building and maintaining the primary index

3. Secondary index on Unordered data file

Assume: using block pointers (not record pointers) and single level

Index entry size remains as 15 bytes (V + P = 9 + 6 = 15)

BFR-I (blocking factor for index) remains as 68

Calculate number of index entries needed

Now need one index entry for each data file record = 30,000 index entries

Calculate number of index blocks needed

30,000 index entries / 68 entries/block =>

441.17… = 442 index blocks

To find a particular index entry in this (ordered) index, can do binary search; $\log_2 442$ = ~8.78 => maximum of 9 disk block reads for index blocks

To get correct data block through index entry pointer: read 1 data file block

Still need to do linear search on records within data block, but this is within memory and relatively fast

Total maximum number of disk blocks read: max 9 + 1 = max 10

Note: don't have to maintain order in data file, so insertion and deletion of data file records is not a significant cost, but a secondary index requires perhaps significant additional storage and maintenance beyond the data file storage

Example 2, Slide 52

How many levels of indexing would be needed for the file from example 1?

Assume: ordered file with primary index

Information from previous examples:

Data file blocks: 3000

Primary index blocks = 45

BFR-I = 68

Will need to add 45 second-level index entries to access 45 primary-level index blocks

45 entries fits in one block, for second-level index is enough

Total number of disk blocks read: 3 (1 at second level index, 1 at primary level of index, 1 data file block)

Note: insertions and deletion still expensive for ordered data file, increase storage for index

Example 3 – Slide 59

Calculate the order p of a B+ tree given:

Block size B = 512
V (size of index value) = 9B
Pr (data record pointer) = 7B
P (index tree pointer) = 6B

Note: order p of a B+ is similar to blocking factor: how many tree pointers can you have in each B+ tree interior index block?

We can store some number p of tree pointers + one less key value in one block

$$\begin{aligned}
(p * 6) + ((p - 1) * 9) &<= 512 \\
6p \quad + \quad 9p - 9 &<= 512 \\
15p \quad - 9 &<= 512 \\
15p &<= 521 \\
p &<= 34.73 \\
p &= 34
\end{aligned}$$

Extra question: Approximately how many levels in a B+ tree would you need to index 1 million data records with p = 50 ?

Level 1: 1 index block (indexing 50 subtrees)
Level 2: 50 index blocks (indexing 2500 subtrees)
Level 3: 2500 index blocks (indexing 125,000 subtrees)
Level 4: 125,000 index blocks (indexing 6,250,000 *data file record entries*)

Answer: ~4 levels

Note: Level 4 nodes may have different p; need data file record pointers in addition to key values, but only need 1 tree pointer (at end, to link blocks in order)