1419

Lecture 15: Indexed Sequential Files

Last Few weeks: Direct (Hashed) Files

Today: Indexed Sequential Files
- Introduction
- Flat indexes
- Multi-level indexes
- Static Reorganization

Folk & Zoellick, ch. 6.

Indexed Sequential Files

- Has many of the benefits of both sequential & direct files.
- Records can be retrieved sequentially
- Records can be retrieved in just a <u>few</u> file accesses.

  ∴ Slightly slower than direct files, <u>much</u> faster than sequential files
  an order of magnitude faster than binary search (for large files, of more than a million records).

- Range queries can be answered efficiently.

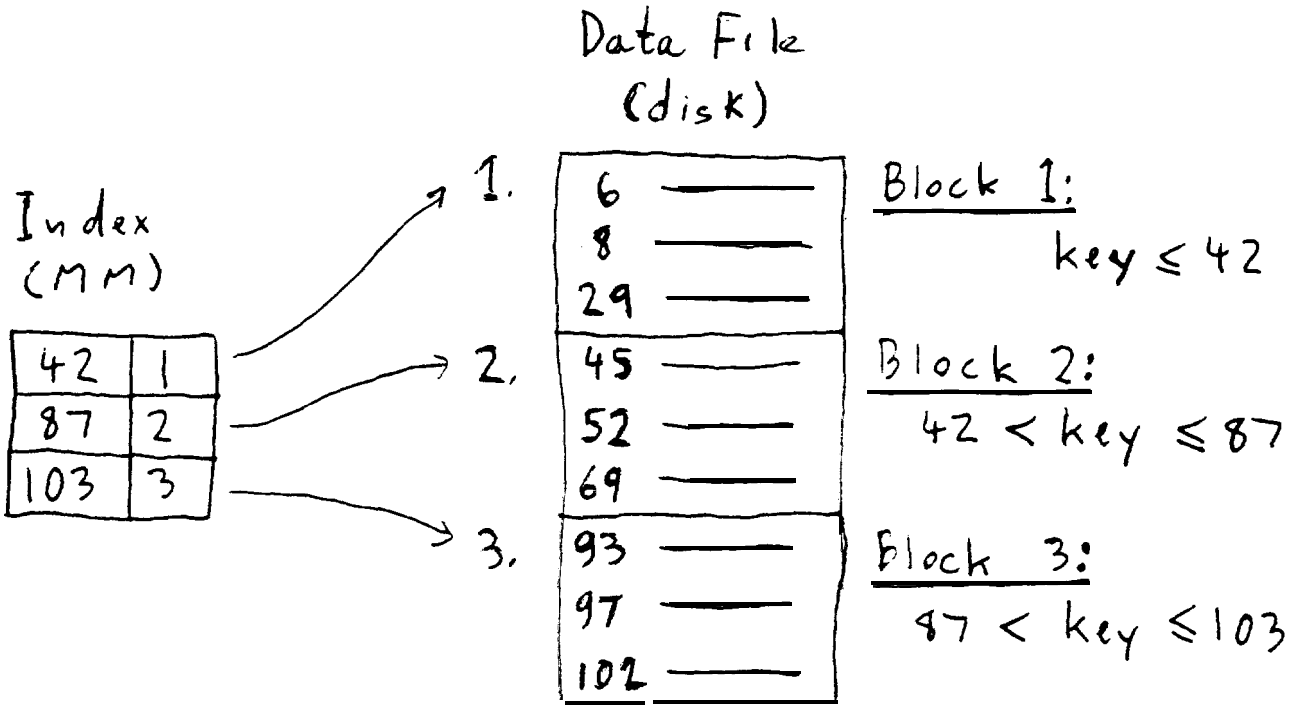
Indexed Sequential Files

- The data file is augmented with an index file, which aids in retrieving records (like the index of a book).

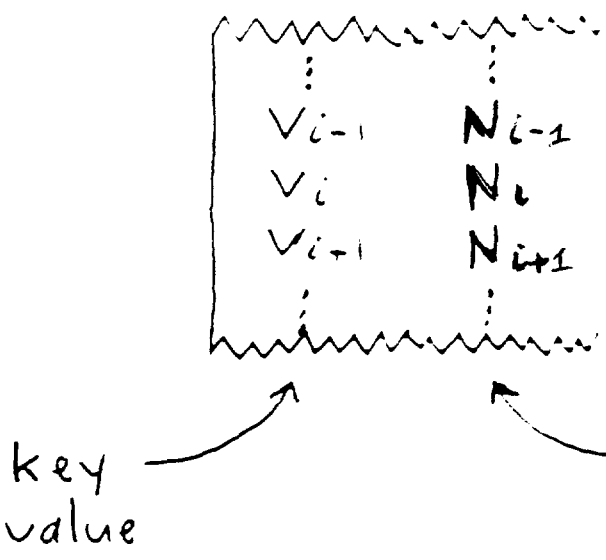
- Like direct files, indexed files may be
  - static (don't expand & contract)
  - Dynamic (expand & contract incrementally).

- Today, we will look at static files.

Static Index Example



General Structure of Index:



Block N_i in data File contains all records s.t. $V_{i-1} < key \leq V_i$

Examples

(1) Retrieve record with key = 97.

- First, scan the index in M.M.
- $87 < 97 \leq 103$
- ∴ Read block 3 from the file into M.M.
- Scan the block for key 97.
- Extract the record, & return it to user.
- Cost = 1 file access

(2) Retrieve record with key = 59

- First, scan the index in M.M.
- $42 < 59 \leq 87$
- ∴ Read block 2 from the file into M.M.
- Scan the block for key 59.
- Not found. Report failure to user.
- Cost = 1 file access.

Problem

What if the index is too big to fit in main memory?

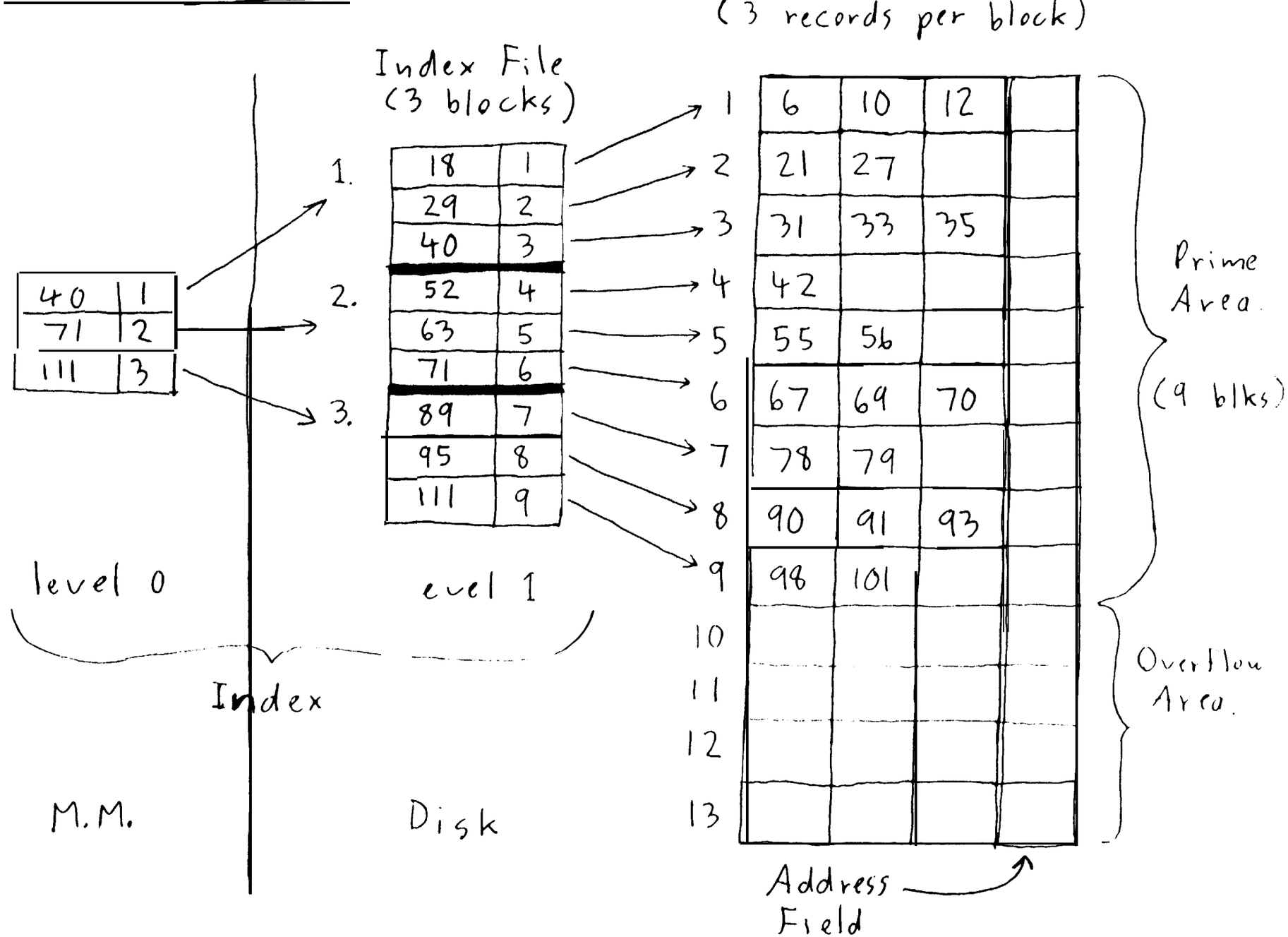
(This happens when the data file is very big.)

Solution

Use a multi-level index

- Put the index in a file on disk
- Build an index for the index in M.M.

Multi-Level Index



Examples

Retrieve 91:

Scan level-0 index in M.M

$71 < 91 \leq 111$

∴ Read block 3 from level-1 index file.

Scan this block in MM.

$89 < 91 \leq 95$

∴ Read block 8 from data file

Scan this block in MM for key 91

Extract the record, & return it to user

Cost = 2 file accesses (both reads)

Examples (Cont)

Insert 57:

Scan level-0 index in MM.

$40 < 57 \leq 71$

$\therefore$ Read block 2 from level-1 index file.

Scan this block in MM.

$52 < 57 \leq 63$

$\therefore$ Read block 5 from data file.

In MM, insert the record (with key 57) into an empty space in this block.

Write the block back to the data file.

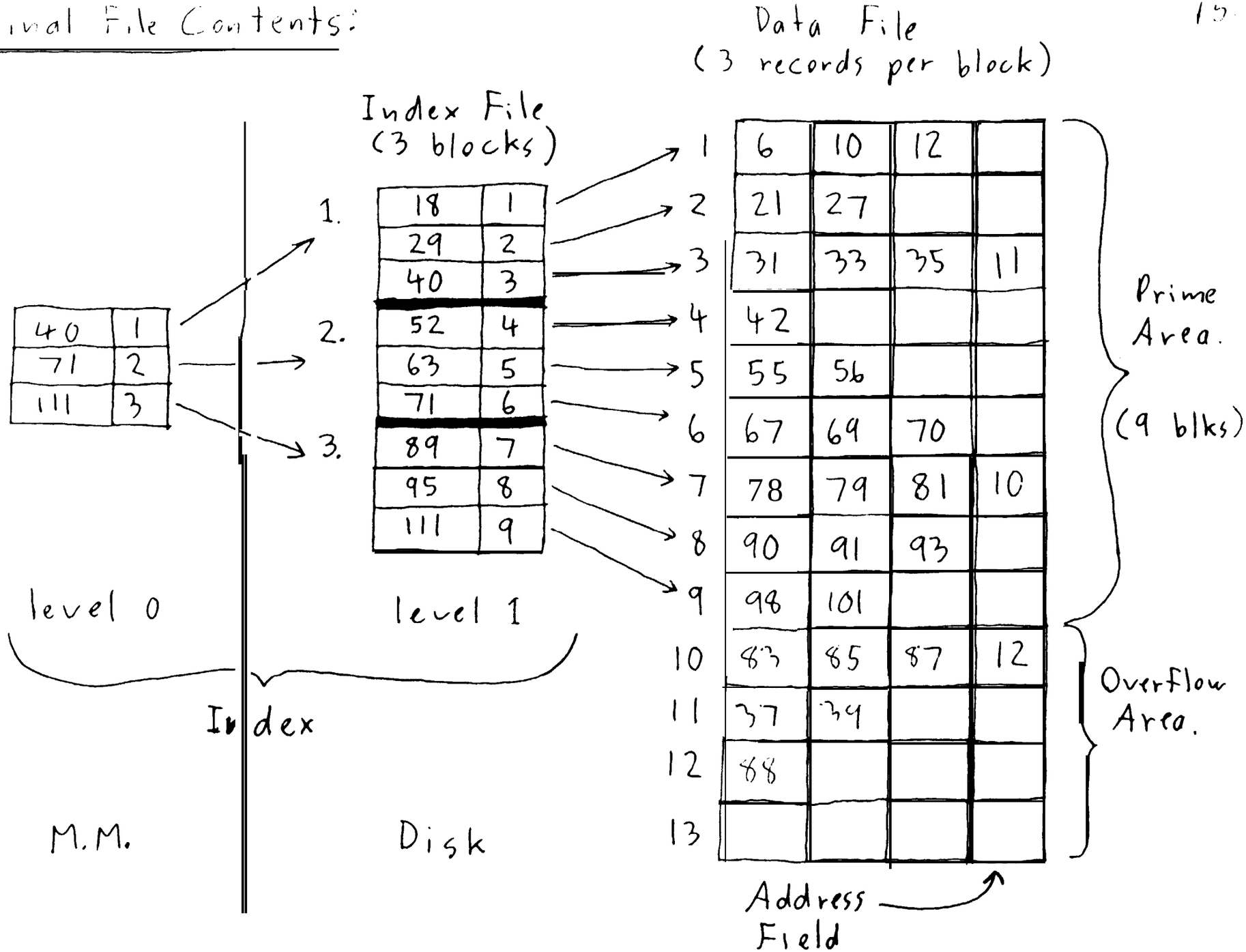
Cost = 2 file reads + 1 file write
= 3 file accesses

Examples (cont.)

Continue inserting 81, 83, 85, 37, 39, 87, 88.

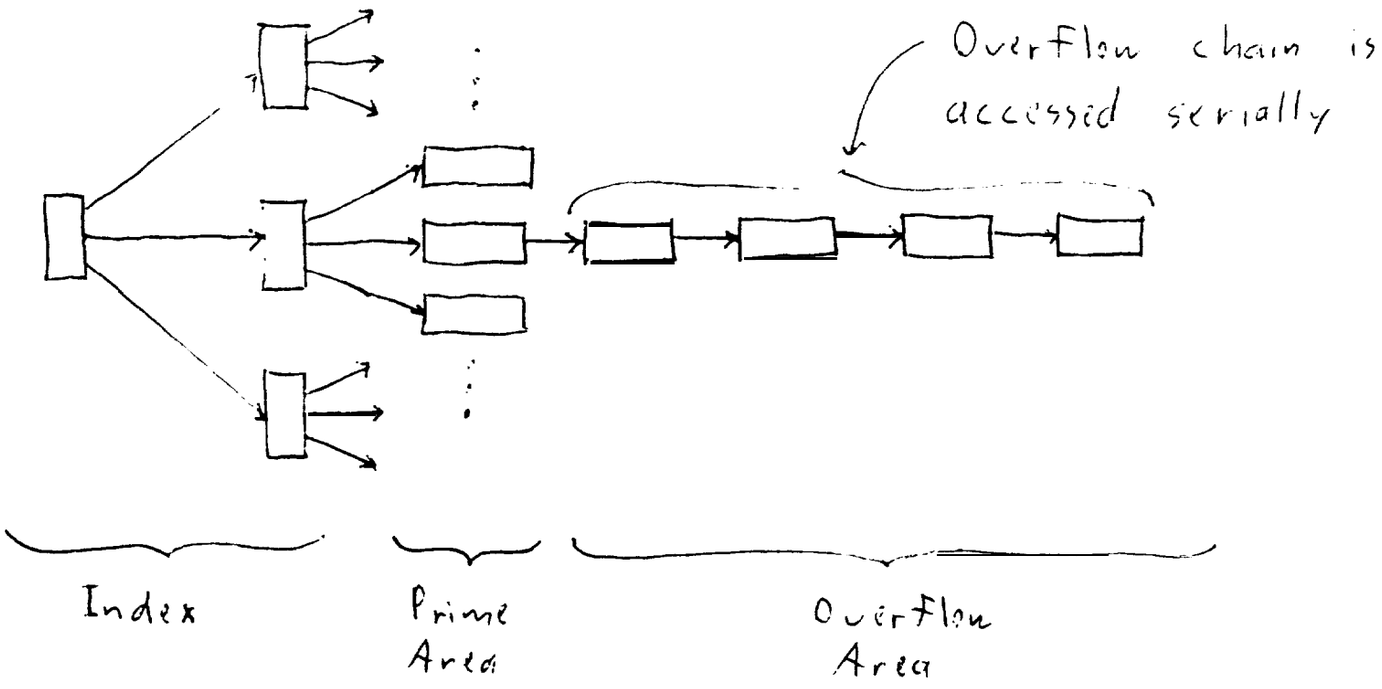
Retrieve 78, 83, 87, 76, 84, 88.

Final File Contents:



Problem

As data (records) is inserted into the file, the number of overflow blocks increases, the overflow chains lengthen, and retrievals become slower (on average).



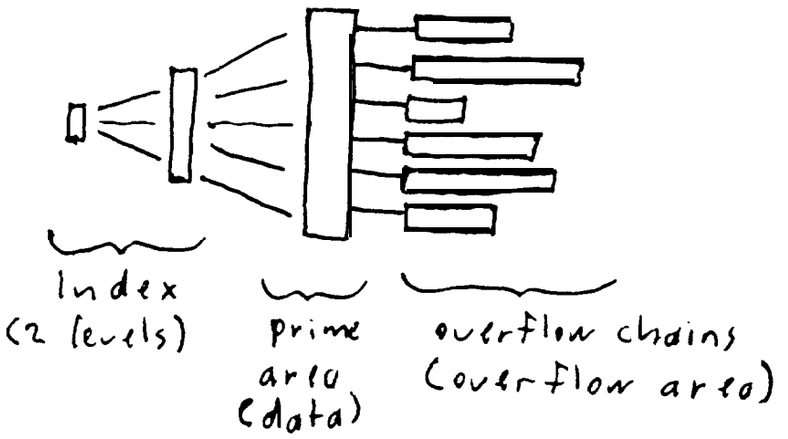
Solution : File Reorganization

- From time to time, the file and index must be reorganized

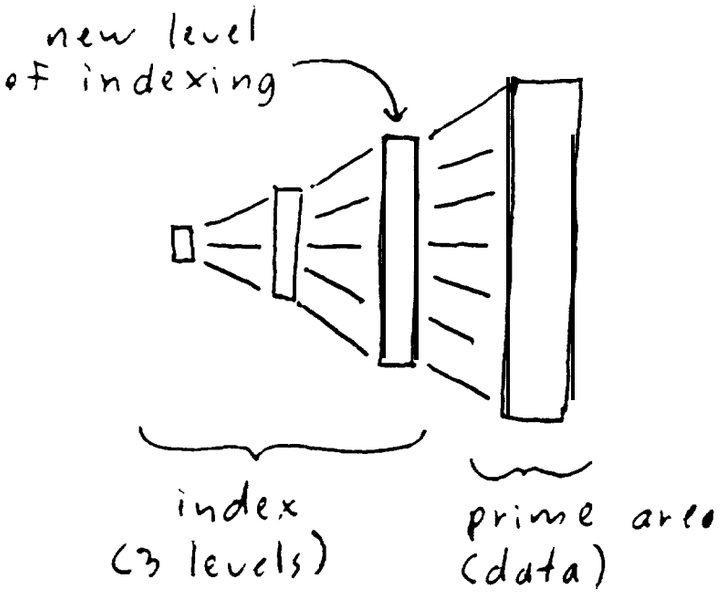
- ie, the overflow area is "merged" with the prime area, and a new level of indexing is created

Reorganizing Static Indexed Files

Before Reorganization:



After Reorganization:



Disadvantages of Static Reorganization

- It is time consuming.
- Users cannot access the data during reorganization.
- ∴ Must be done after business hours (eg. at night or on weekends)
- ∴ Not suitable for 7×24 operations (7 days a week, 24 hours a day), like airline reservations, ATM banking, etc

Next Day: Dynamic Indexed Files