File Organisations and Indexes

File organisations

- Three types of file organisations
  - Heap, Sorted and Hashed
- The COST of processing DB queries
  - SCAN, Search (Single, Range), Insert, Delete

File Organisations

- A bit of recall
  - The basic store unit on disk (in memory) is block (page)
  - We will use page/block interchangeably.
  - One page consists of multiple data records.
- Three types
  - Heap Files (The simplest)
    - Page after page, always insert at the end
  - Sorted Files
    - Records are sorted (within and among pages) w.r.t. the search key
  - Hashed Files.
    - The pages are assigned to multiple buckets, each containing several pages
    - Each page has been assigned with an ID (bucket ID) to tell where to find the page

COST MODEL

- Notations
  - **B**: the number of pages (blocks)
  - **R**: the average number of records in a page.
  - **D**: average time to read or write a disk page.
  - **C**: average time to process a record.
  - **H**: the time required to apply a hash function to a record.

COST = Time of Execution

Operations to be investigated

- ***Scan***: fetch all records in a table.
- Search with equality selection (***SWES***)
  - "Find the students record with sid = z3087251"
- Search with Range Selection (***SWRS***)
  - "Find all students with name alphabetically after 'Smith'"
- ***Insert***: Insert a given record into the table.
- ***Delete***: Delete a record with given rid.

Below, we examine the costs of these operations with respect to the 3 different file organisations.

Heap Files

- Scan:
  - B(D + RC)
- SWES:
  - 0.5B(D + RC) on average if the selection is specified on a key.
  - Otherwise B(D + RC).

B: the number of pages
R: the average number of records in a page (block).
D: average time to read or write a disk page.
C: average time to process a record.

Heap Files

- SWRS:
  - B(D + RC)
- Insert:
  - (D + C) + D = 2D + C (Always insert to the end of the file)
- Delete:
  - Only one record is involved.
    - The average cost is 0.5B(D + RC) + D if rid is not given;
    - otherwise (D + C) + D.
  - Several records are involved. Expensive.

B: the number of pages
R: the average number of records in a page (block).
D: average time to read or write a disk page.
C: average time to process a record.

Sorted Files

Sorted on a search key
(a combination of one or more fields)

Querying against the search key:

- Scan: B(D + RC).
- SWES:
  - O(D **$\log_2$ B** + C **$\log_2$** R) if single record. (log comes from binary search)
  - O(D **$\log_2$** B + C **$\log_2$** R + #matches).
- SWRS: O (D $\log_2$ B + C $\log_2$ R + #matches).
- Insert: expensive.
  - Search cost plus 2 * (0.5B(D + RC)).
- Delete: expensive.
  - Search cost plus 2 * (0.5B(D + RC)).

Hashed Files

- The pages in a file are grouped into buckets.
- The buckets are defined by a hash function.
- Pages are kept at about 80% occupancy.

Assume the data manipulation is based on the hash key.

- Scan: **1.25**B(D + RC).
- SWES: H + D + 0.5RC if each hash bucket contains only one page.
- SWRS: 1.25B(D + RC). (No help from the hash structure)
- Insert: Search cost plus C + D if one block involved.
- Delete: Search cost plus C + D if one block involved.

Summary

File Type	Scan	Equality Search	Range Search	Insert	Delete
Heap	BD	0.5 BD	BD	Search + D	Search + D
Sorted	BD	D log B	D log B + # matches	Search + BD	Search + BD
Hashed	1.25 BD	D	1.25 BD	2 D	Search + BD

A Comparison of **I/O** Costs

(**I/O** Costs: The cost of reading from or writing to disk)

Introduction to Index

- What is index?

- What is the basic organisation of index?

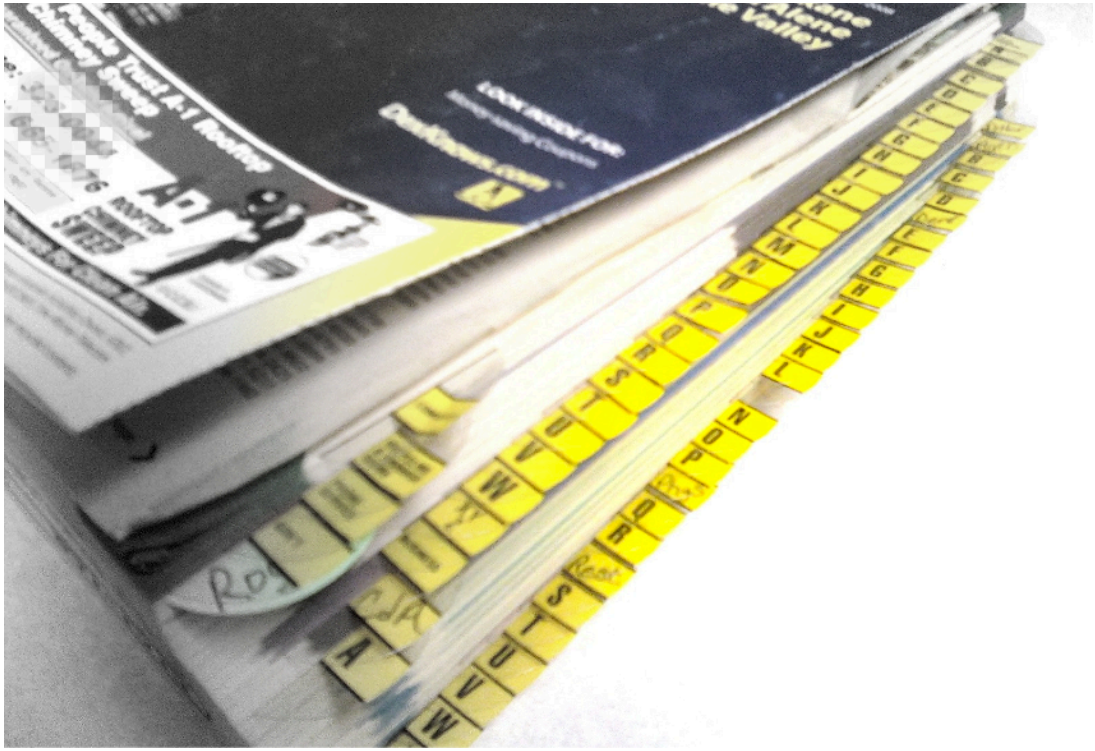
What is index?

Book has been arranged via categories, subjects.

Same categories will be stored in the same area.

Book is data, then Catalogue is the index

What is index?



Phone number has been arranged in Alphabets, groups (work, friends, …)

Phone number is the data
The Alphabets and groups are the index.

What is index?

- Auxiliary data
- Properly organised (data structure)
- To facilitate data search

Indexes

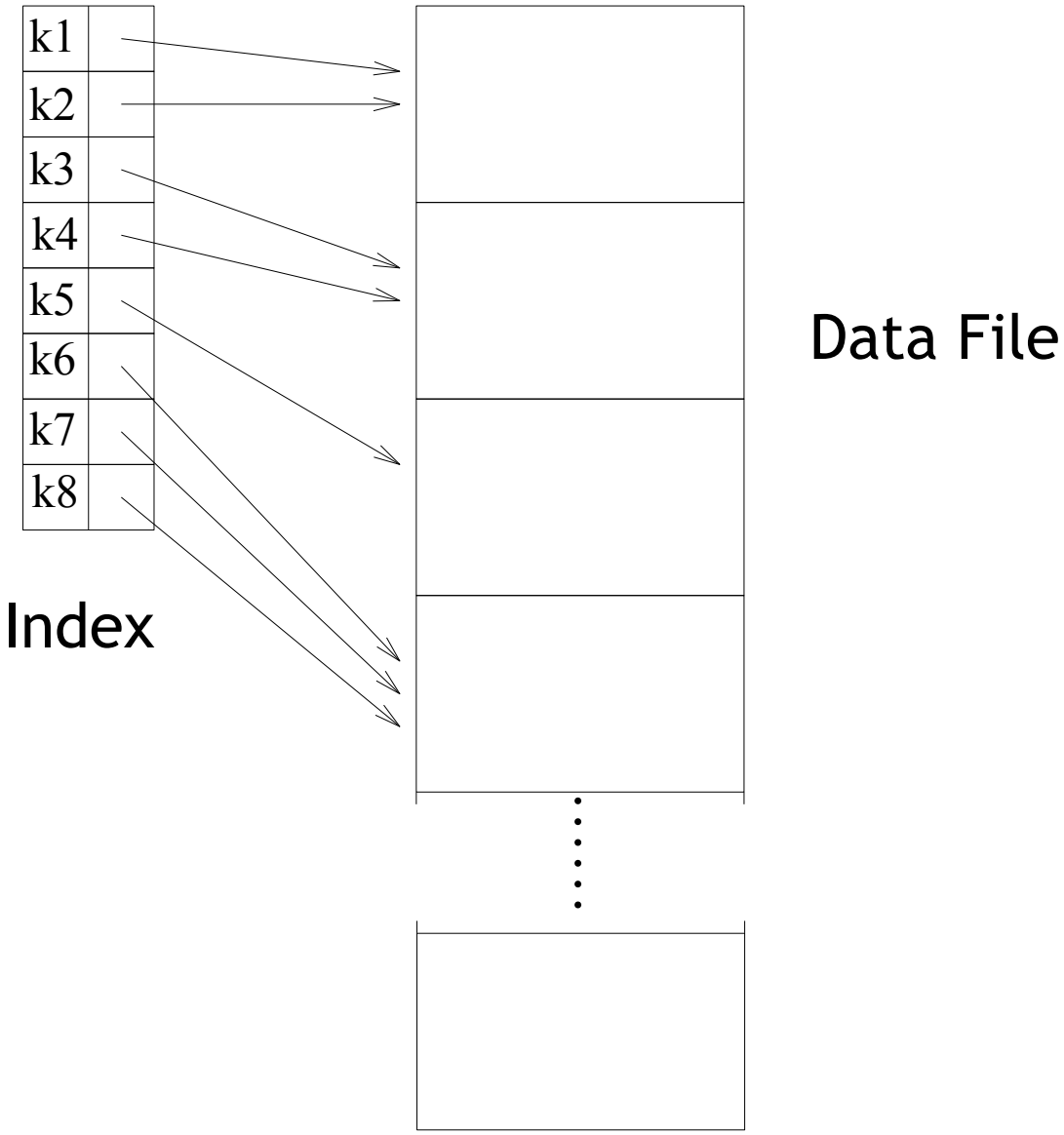
Word indexes in a book:

INDEXES

aardvark 25,36	lion 18
bat 12	llama 17,21,22
cat 1,5,12	sloth 18
dog 3	tiger 18
elephant . . 17	wombat . . . 27
emu 28	zebra 19

- A table of key values, where each entry gives places where key is used.
- Aim: efficient access to records via key.

Indexing Structure



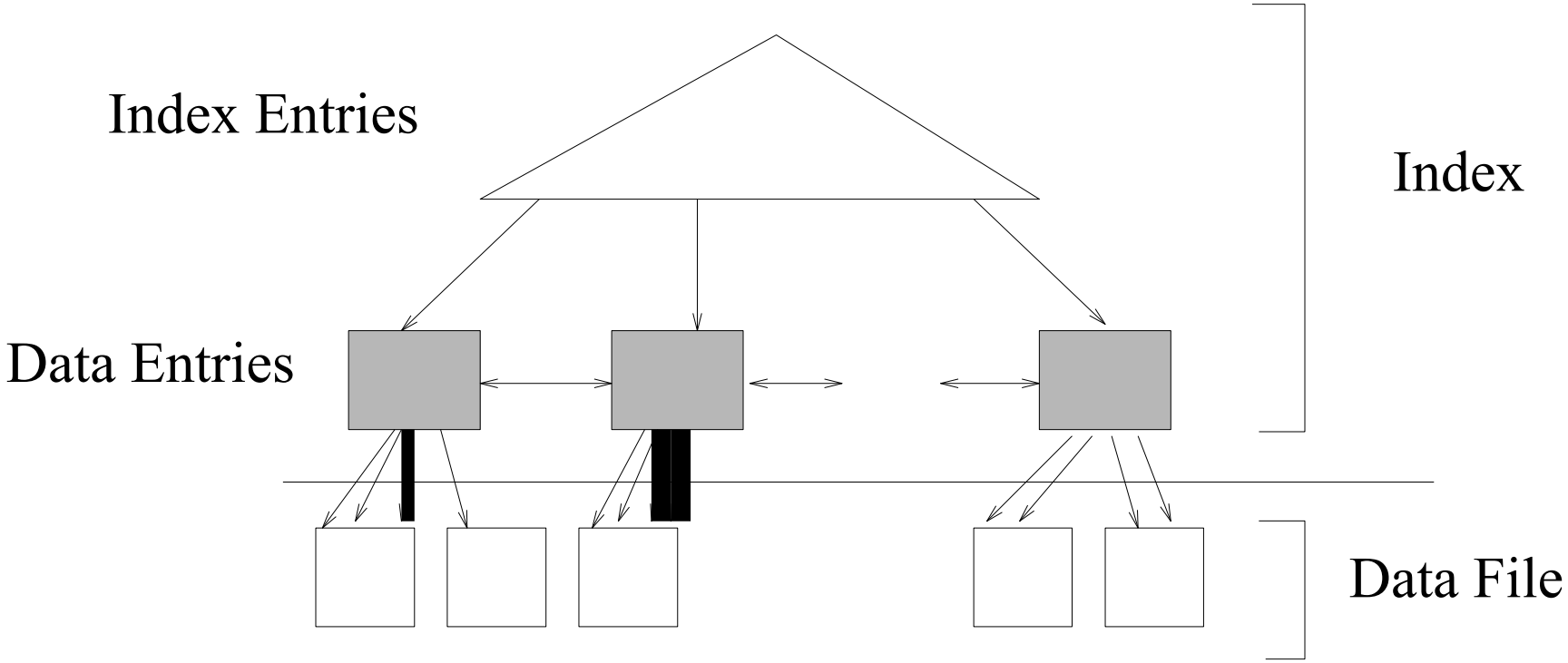
Indexing Structure

- Index is collection of data entries k*:
  - Each data entry $k*$ contains enough information to retrieve (one or more) records with search key value k.

- Indexing:
  - How are data entries organised in order to support efficient retrieval of data entries with a given search key value?
  - What is stored as a data entry?

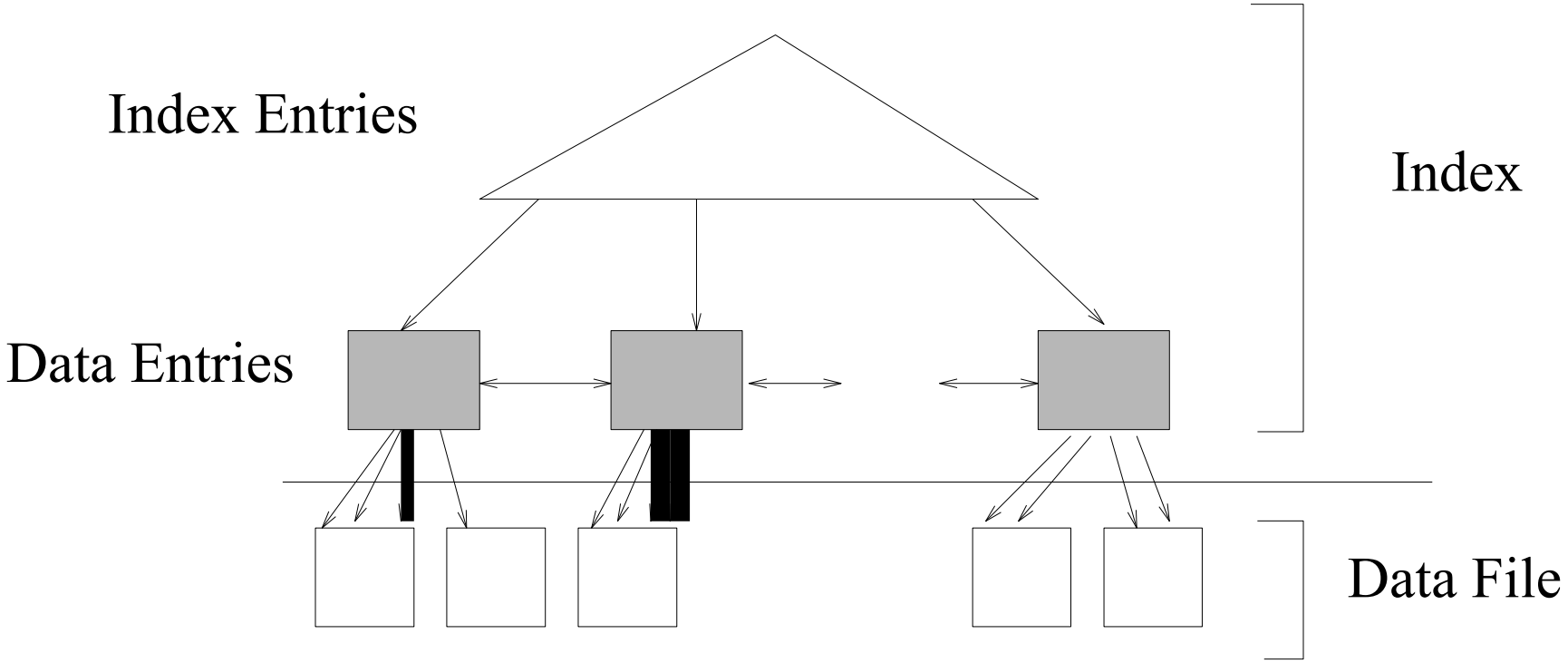
Clustered Index

- Clustered: Sort and store the data records in the table or view based on the search keys of the index.
- Typically, the search key of file is the same as the search key of index.

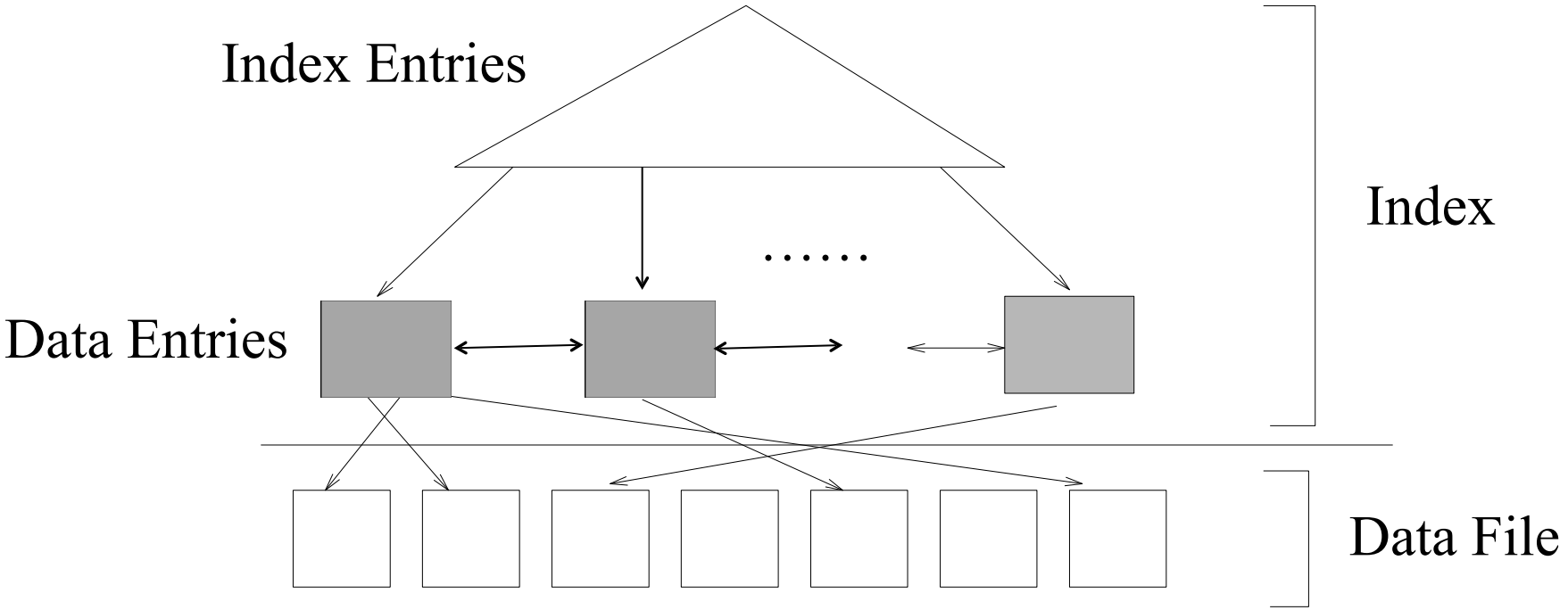


Clustered Index

- Clustered indexes are relatively expensive to maintain.
- A data file can be clustered (indexed) on **at most one** search key.



Unclustered Index

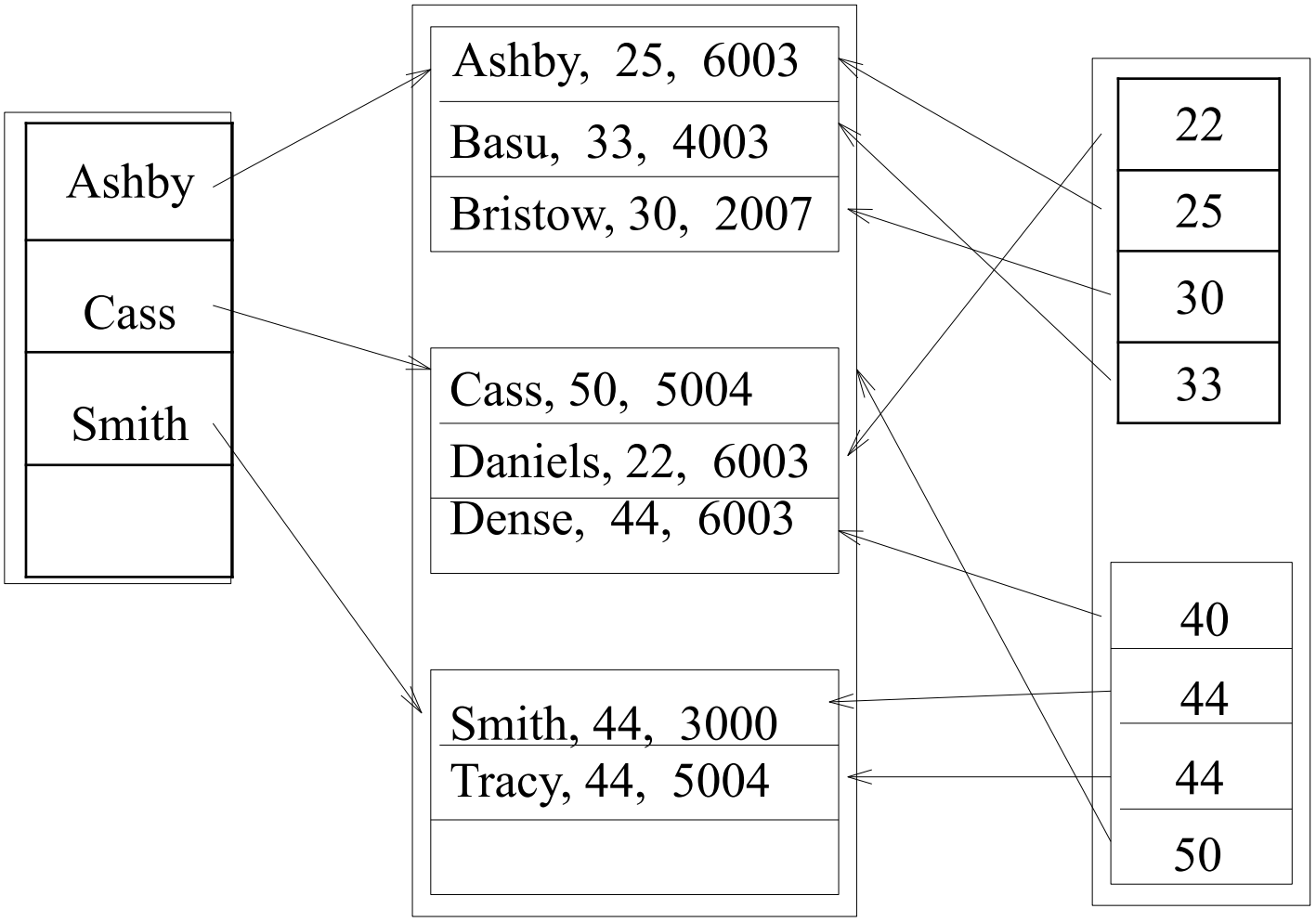


Dense VS Sparse Indexes

- Dense Index and Sparse Index
  - Dense Index contains (**at least**) one data entry for every search key value.
  - Sparse Index: one search key can point to a set of data entries

Q: Can we build a sparse index that is not clustered?

Dense VS Sparse Indexes



Sparse Index

Dense Index

Primary and Secondary Indexes

- Primary: Indexing fields include **primary key**.
  - There can be at most one primary index for a table

- Secondary: otherwise.
  - Composite search keys: search key contains several fields.