File Organizations and Indexes

COST MODEL

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.

3 File Organizations:

Heap Files.

Sorted Files.

Hashed Files.

Operations to be investigated

Scan: fetch all records in a file.

Search with equality selection. (SWES) (“Find the students record with sid = 23”)

Search with Range Selection. (SWRS)

(“Find all student after ‘Smith’”)

Insert: Insert a given record into the

Delete: Delete a record with given rid.

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

Heap Files

Scan:

B(D + RC) where

- B is the number of pages, and
- R is the a records in a page (blo

SWES:

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

Heap Files

SWRS: B(D + RC).

Insert: 2D + C . (Always insert to the end of the file)

Delete:

- Only one rec

  - The average cost is 0.5 D if rid is not given;
  - otherwise (D + C) + D.

- Several records are involved. Expensive.

Sorted Files

Sorted on a search key - a combination of one or more fields.

If the following query is made against the search key, then:

1. Scan: B(D +
2. SWES:
   - O(D $\log_2$ B + C $\log_2$ R) if s .
   - O(D $\log_2$ B + C $\log_2$ R + #

3. SWRS: O (D $\log_2$ B + C $\log_2$ R + #matches).
4. Insert: expensive.
   - Search cost plus $2 * (0.5B(D + RC))$.
5. 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 dat d on the hash key.

- Scan: 1.25B(D + RC).

- SWES: H + D + 0.5RC if each _ et 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	B D		# mat		Search + BD
Hashed	1.25 BD	D	1.25 BD	2 D	Search + BD

A Comparison of I/O Costs

Indexes

Basic idea behind index is as for books.

INDEXES

aardvark 25,36 . . . . . lion 18

bat . . . . . 12 llama 17,21,22

cat 18

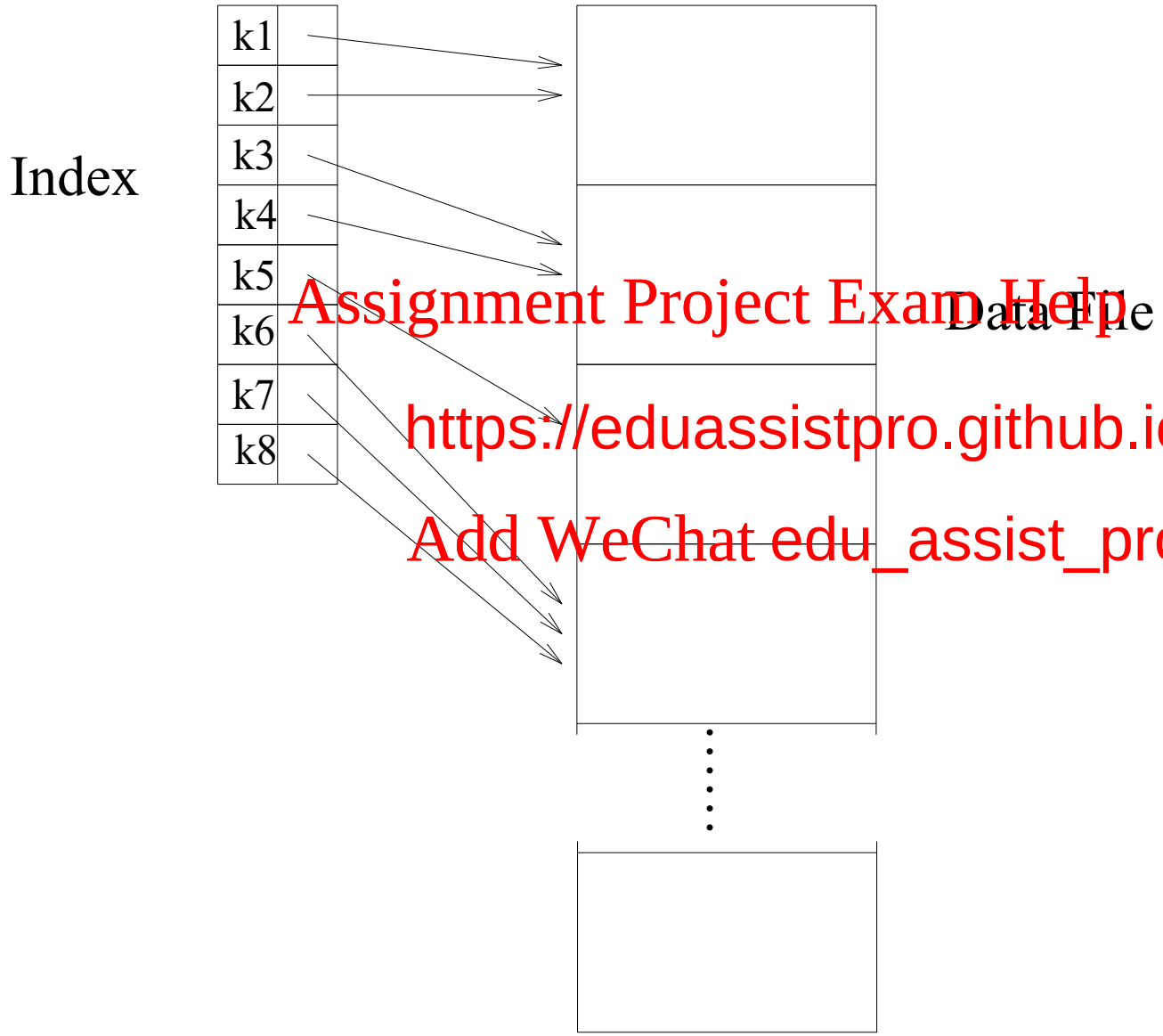
dog . . . . . 3 tig 18

elephant . . 17 wo 7

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 values.

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 or order to support efficient retrieval of data entries with a given search key value?
- Exactly what is stored as a data entry?

Alternatives for Data Entries in an Index

- A data entry k∗ is an actual data record (with search key value k).
- A data entry is (k, rid) pair (rid is the record id of a data record with search key value k).
- A data entry is (k, rid-list) pair (rid-list is the list of record ids of data records with search key value k).

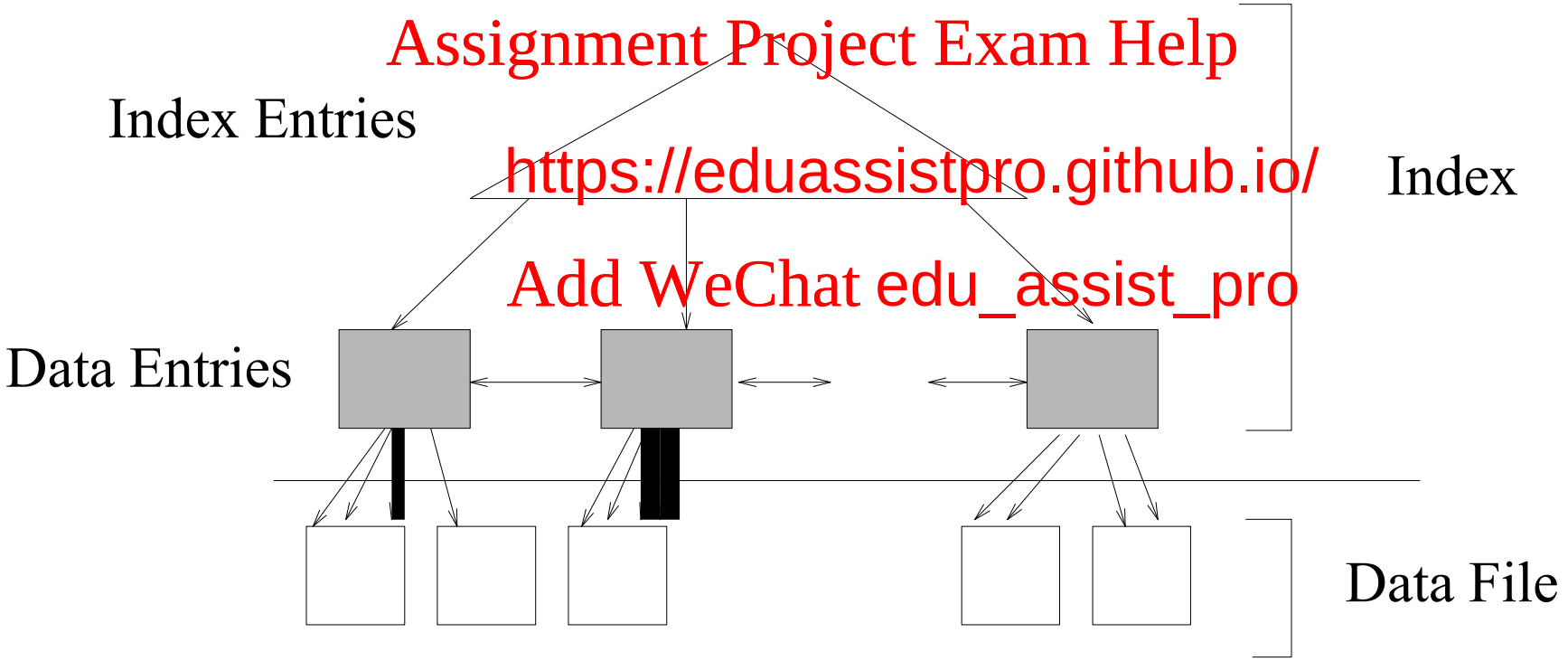
Example: (Xuemin Lin, page 12), (Xuemin Lin, page 100)

VS

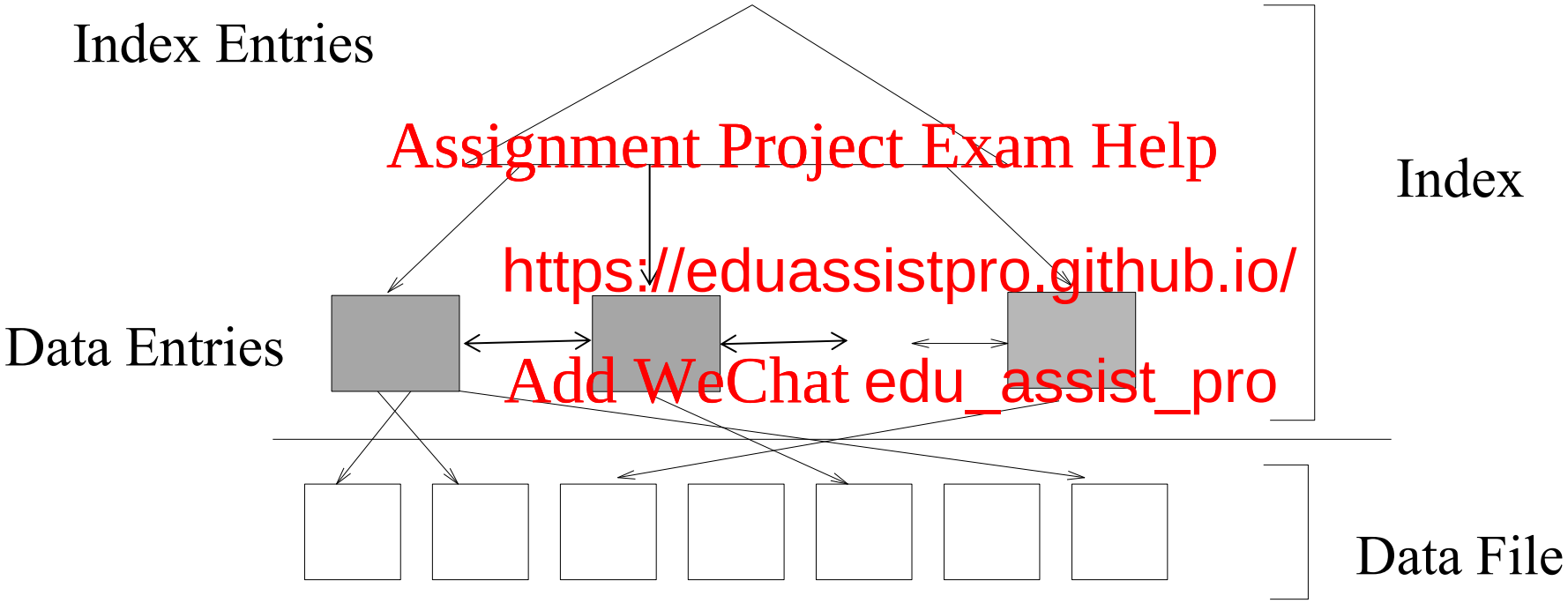
(Xuemin Lin, page 12, page 100)

Clustered Index

- Clustered: a file is organized of data records is the same as or close to the ordering of data entries in some index.
- Typically, the search key of file is the same as the search key of index.



Unclustered Index



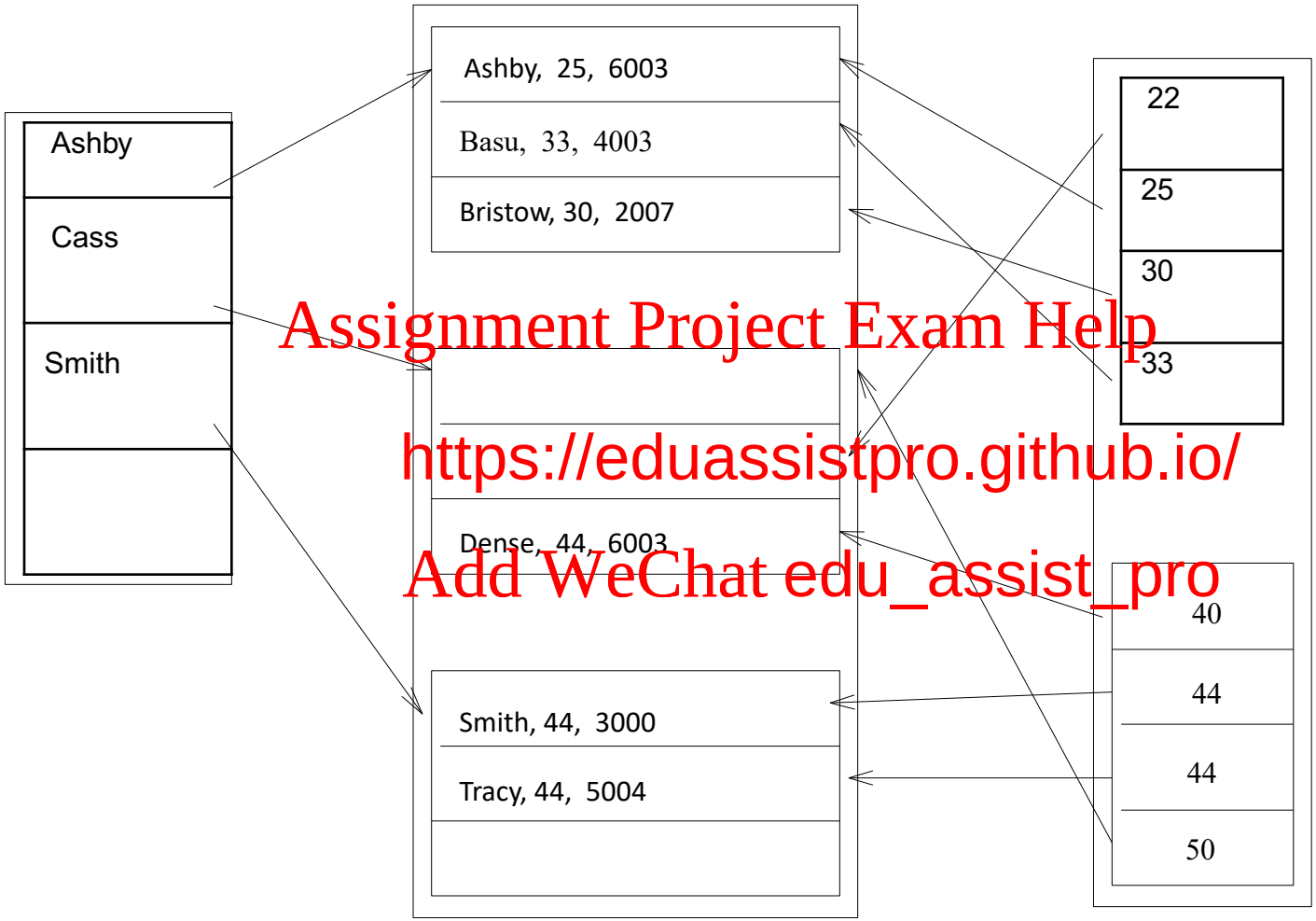
- Clustered indexes are relatively expensive to maintain.
- A data file can be clustered on at most one search key.

Dense VS Sparse Indexes

- Dense: it contains (at least) one data entry for every search key value.

- Sparse: ot

Q: Can we build a sparse at is not clustered?



Sparse Index VS Dense Index

Primary and Secondary Indexes

- Primary: Indexing fields include primary key.
- Secondary: otherwise.

There may b ary index for a file.

Composite search keys: search key contains several fields.