Data Storage and Query Answering

Indexing and Hashing (1)

Introduction

- We have discussed the organization of records in secondary storage blocks.
- Records have an address, either logical or physical.
- But SQL queries reference attribute values, not record addresses.

  SELECT * FROM R WHERE a=10;
- How to find the records that have certain specified attribute values?

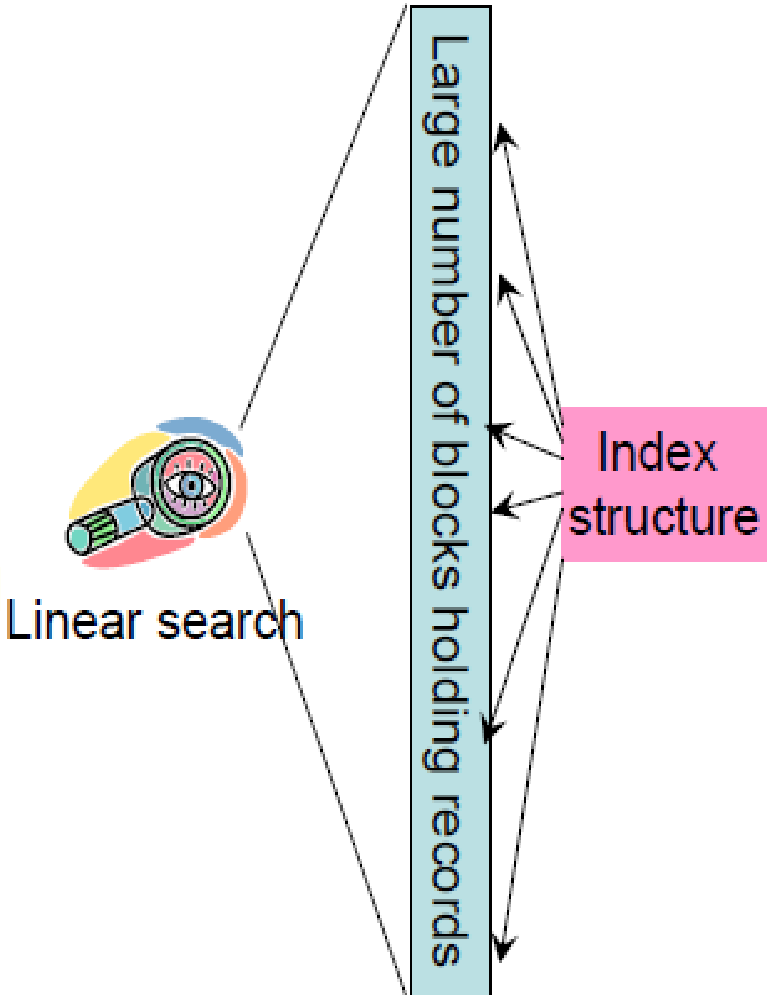
Why Index?

- Table Students (id, name, major, address)
- Query 1: find student with id 200830782
  - Scanning the table to find the student – O(n) time, can be costly
  - Sort all students on id into a sorted list, conduct a binary search on the sorted list – O(log n) query time, fast
    - What if the sorted list cannot be held into main memory?
- Query 2: find students majoring in computer science
  - Can the sorted list on id help?
  - Sort all students on major into a sorted list, conduct a binary search on the sorted list – O(n log n) construction time, O(log n) query time, O(n) space
- Some issues
  - An update to table Students has to be propagated to both sorted lists
  - Tradeoff between time and space – we cannot afford to construct a separate sorted list for each query
  - Queries may be raised ad hoc – we may not gain to construct a separate sorted list on the fly for each query

What is Index?

- An index is an (efficient) data structure that can facilitate answering a set of queries
  - General – can be used to answer a set of queries
  - Efficient – construction, query answering, space, and maintenance
- Issues in index construction
  - Query types – what kinds of queries that an index can support
  - Query answering time
  - Construction time and space cost
  - Maintenance cost
- Search key – an attribute or a set of attributes used to look up records in a file
  - An index is built to facilitate searching on a search key
  - A search key may not be unique – different from key in database design

An index structure provides links to target tuples with minor overheads



Indices Can Make Big Difference

```
SELECT *
FROM Table1 Table2
WHERE P1 AND P2
```

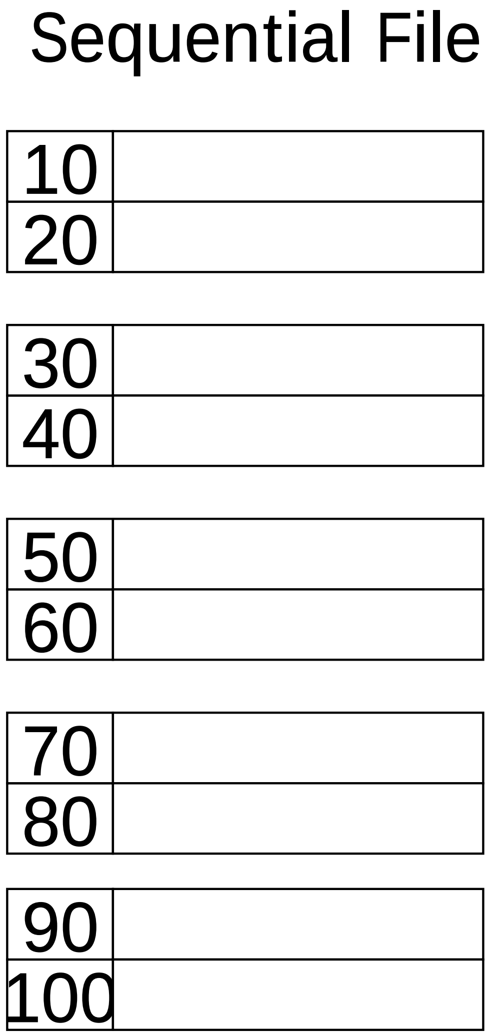
– P1 and P2 are on Table1 and Table2, respectively

- Table1 and Table2 contain 1 million tuples each, P1(Table1) and P2(Table2) contain 100 tuples each
- Without index, 10^{12} tuples will be read!
- With index, only 10,000 tuples will be read!

Index Structures: Concepts

- Storage structures consist of files.
- *Data files* store, e.g., the records of a relation.
- *Search key*: one or more attributes for which we want to be able to search efficiently.
- *Index file* over a data file for some search key associates search key values with pointers to (recordID = rid) data file records that have this value.
- *Sequential file*: records sorted according to their primary key.

Sequential File



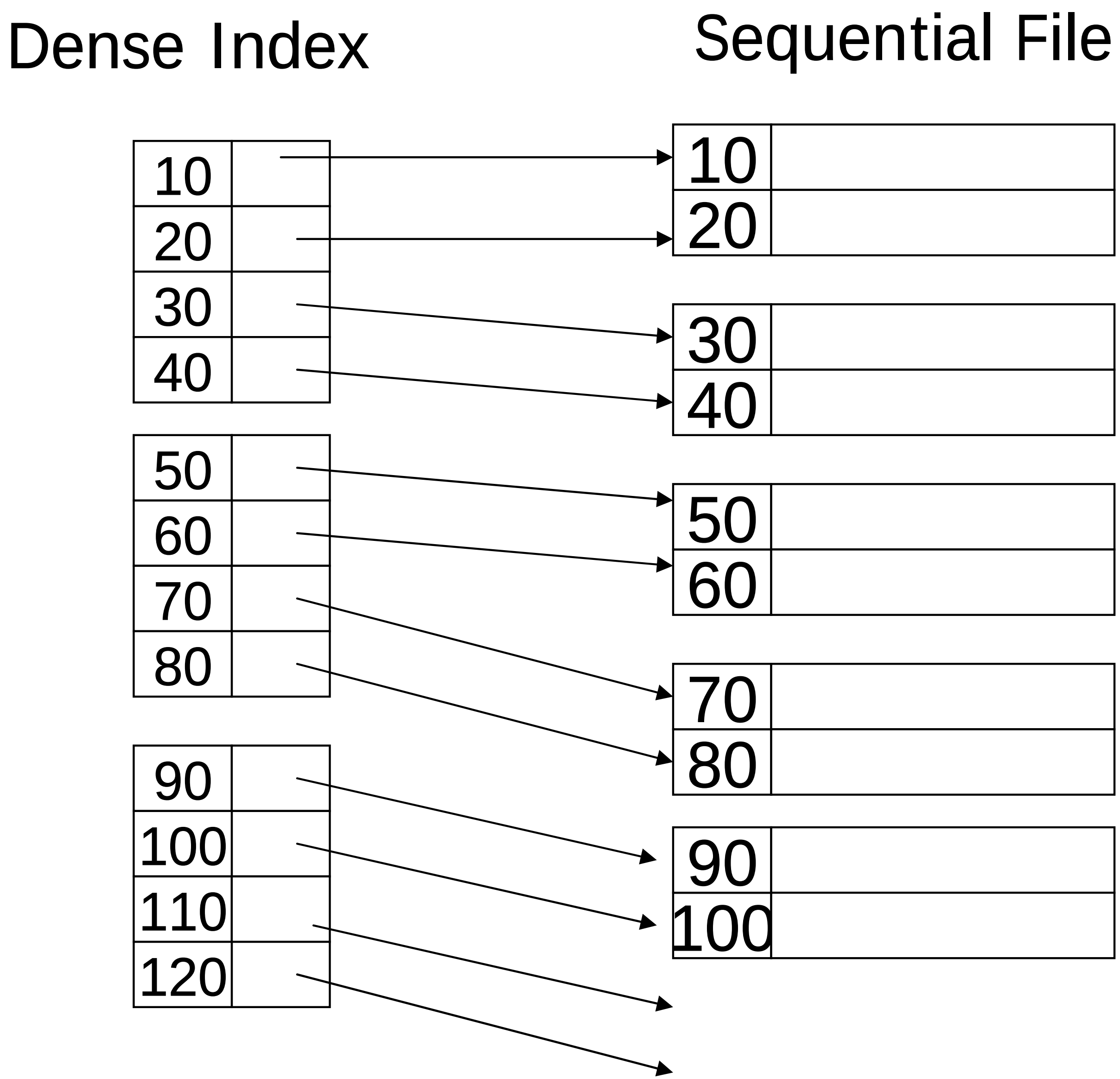
Data Entries

- Three alternatives for data entries k*:
  - record with key value k
  - <k, rid of record with search key value k>
  - <k, list of rids of records with search key k>
- Choice is orthogonal to the indexing technique used to locate entries **k***
- Two major indexing techniques:
  - tree-structures
  - hash tables.

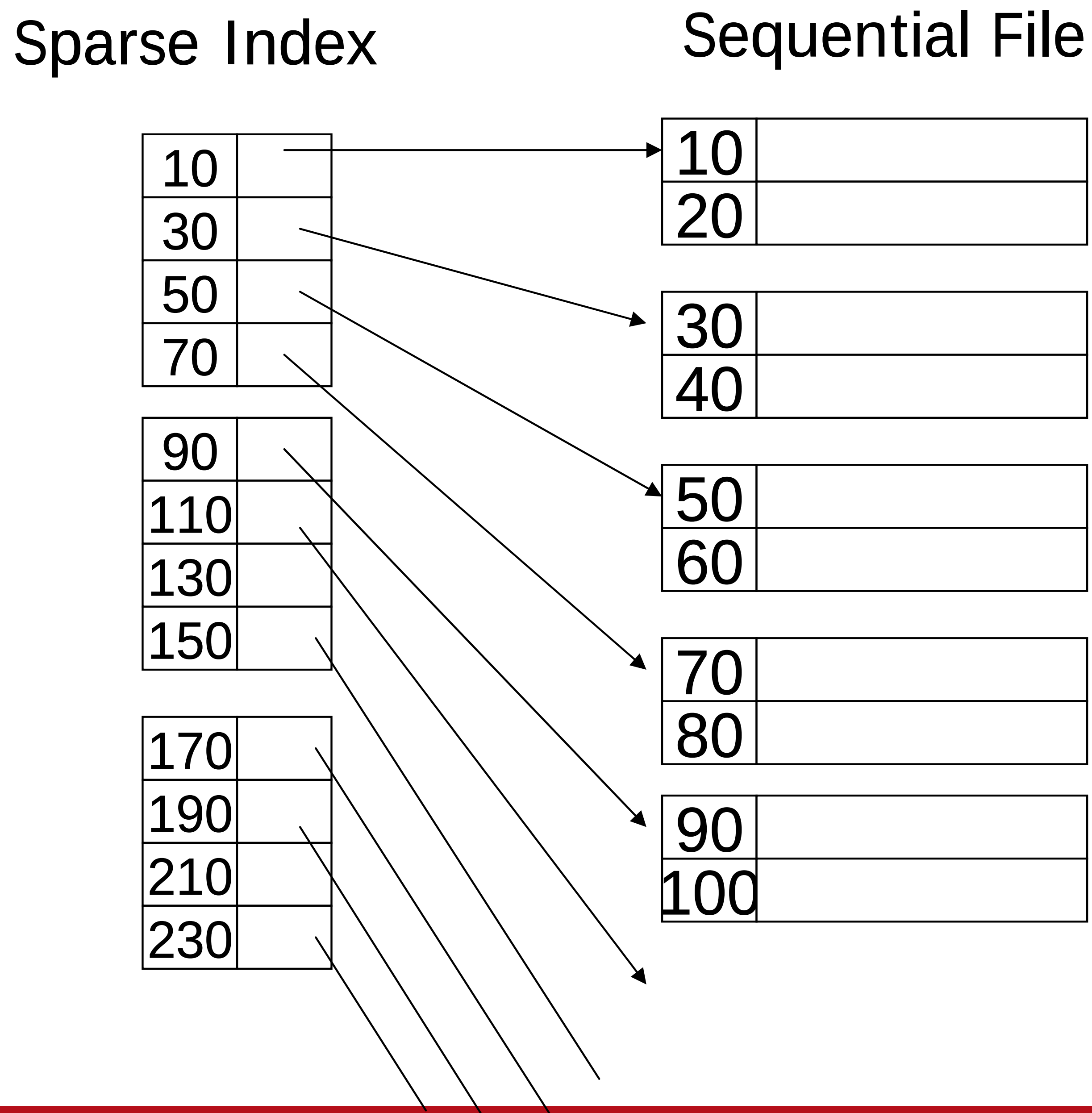
Index-Structure Basics

- *Dense index*: one index entry for every record in the data file.
- *Sparse index*: index entries only for some of the record in the data file. Typically, one entry per block of the data file.
- *Primary index*: determines the location of data file records, i.e. order of index entries same as order of data records.
- *Secondary index* does not determine data location.
- Can only have one primary index, but multiple secondary indexes.

Index-Structure Basics

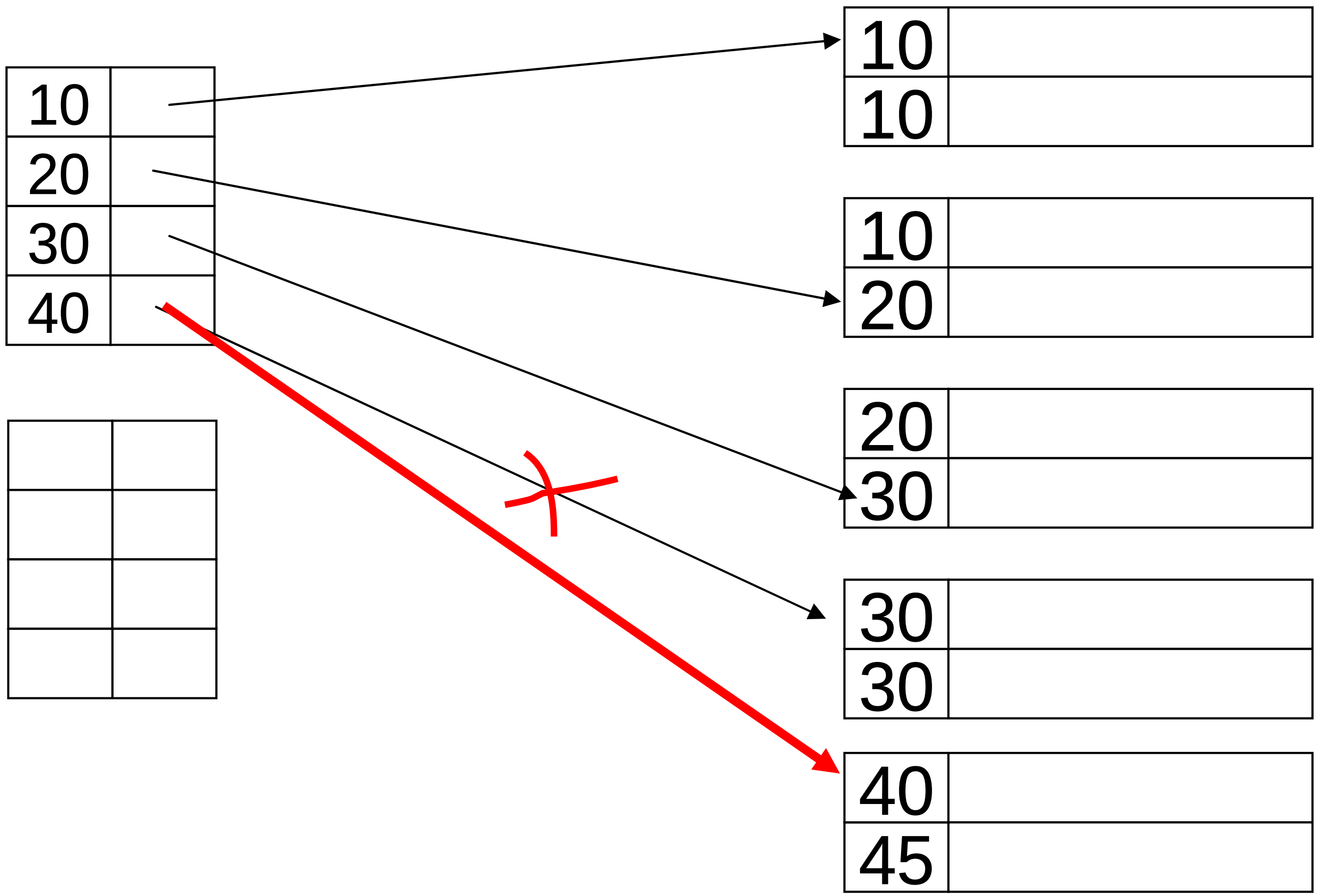


Index-Structure Basics



Index-Structure Basics

Duplicate key values
– sparse index
– index may point to first instance of each value only



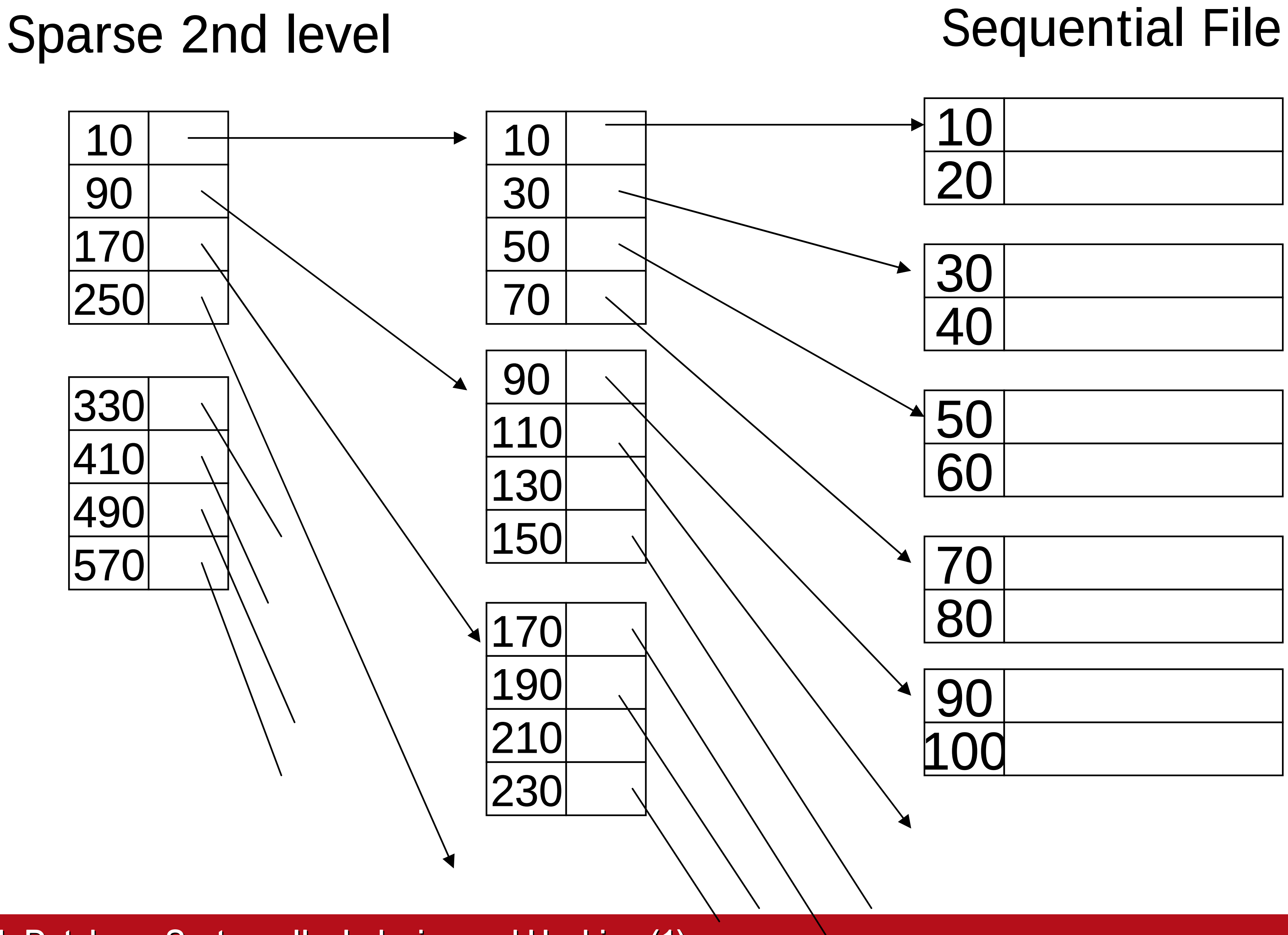
Index-Structure Basics

- Sparse index:
  - requires less index space per record,
  - can keep more of index in memory,
  - needed for secondary indexes.
- Dense index:
  - can tell if any record exists without accessing data file,
  - better for insertions.

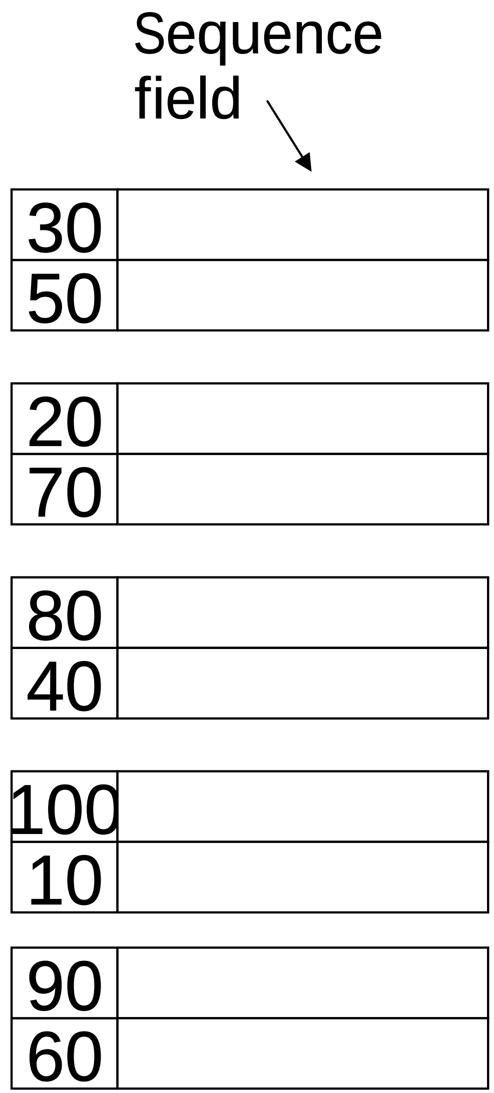
Index-Structure Basics

- Index file can become very large, e.g. at least one tenth of data file size for records with ten attributes of same length.
- To speed-up index access, add a second index level on top of the first index level, a third level on top of the second one, . . .
- First level can be dense, other levels are sparse.
  - Why? Can we build a dense, 2nd level index for a dense index?

Index-Structure Basics

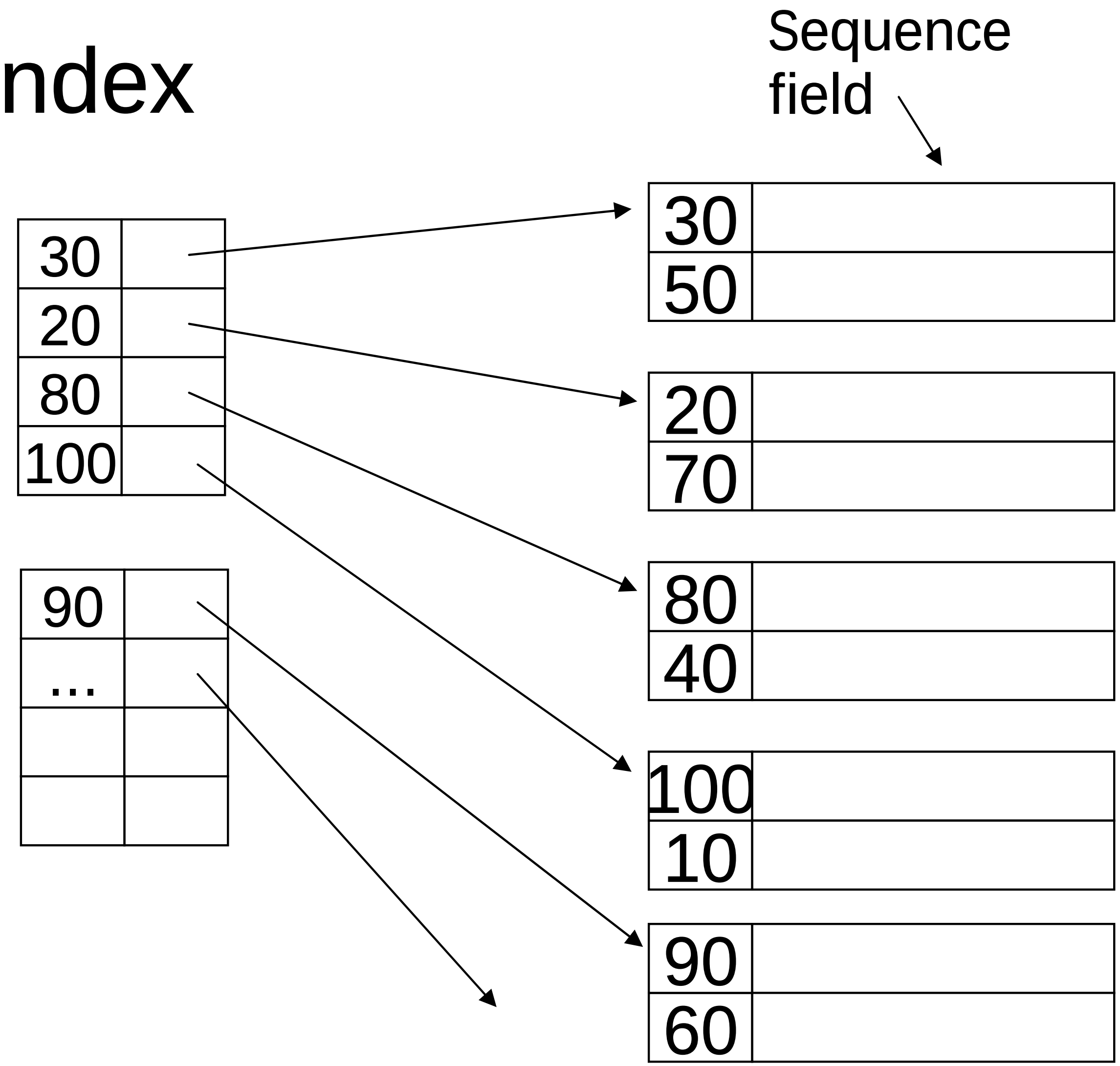


Secondary Indexes



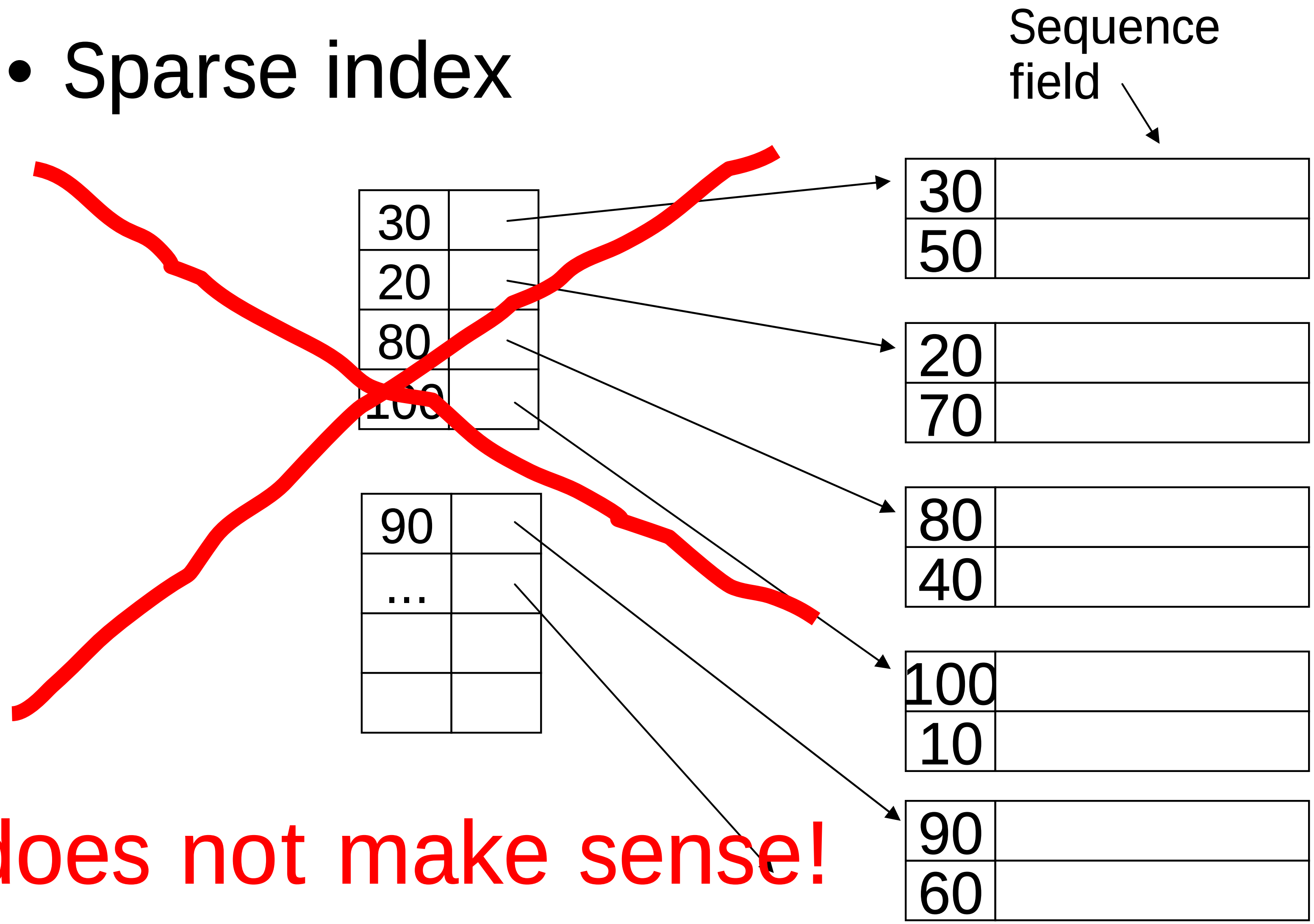
Secondary Indexes

- Sparse index



Secondary Indexes

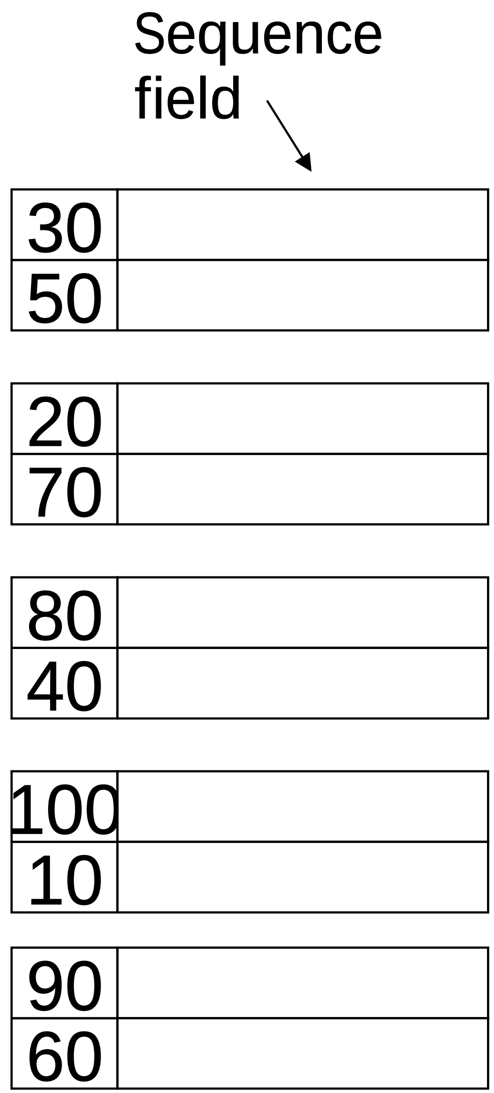
- Sparse index



does not make sense!

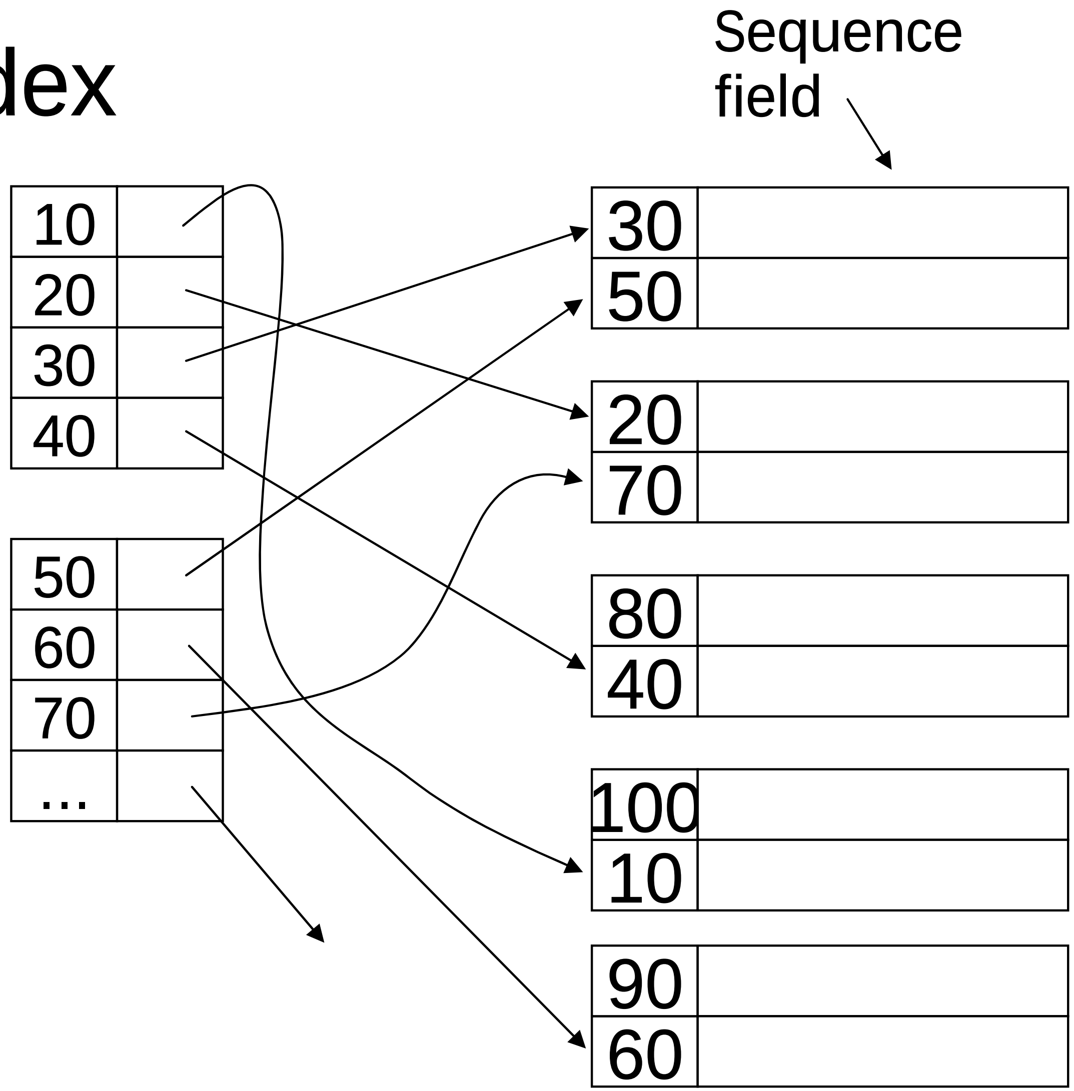
Secondary Indexes

- Dense index



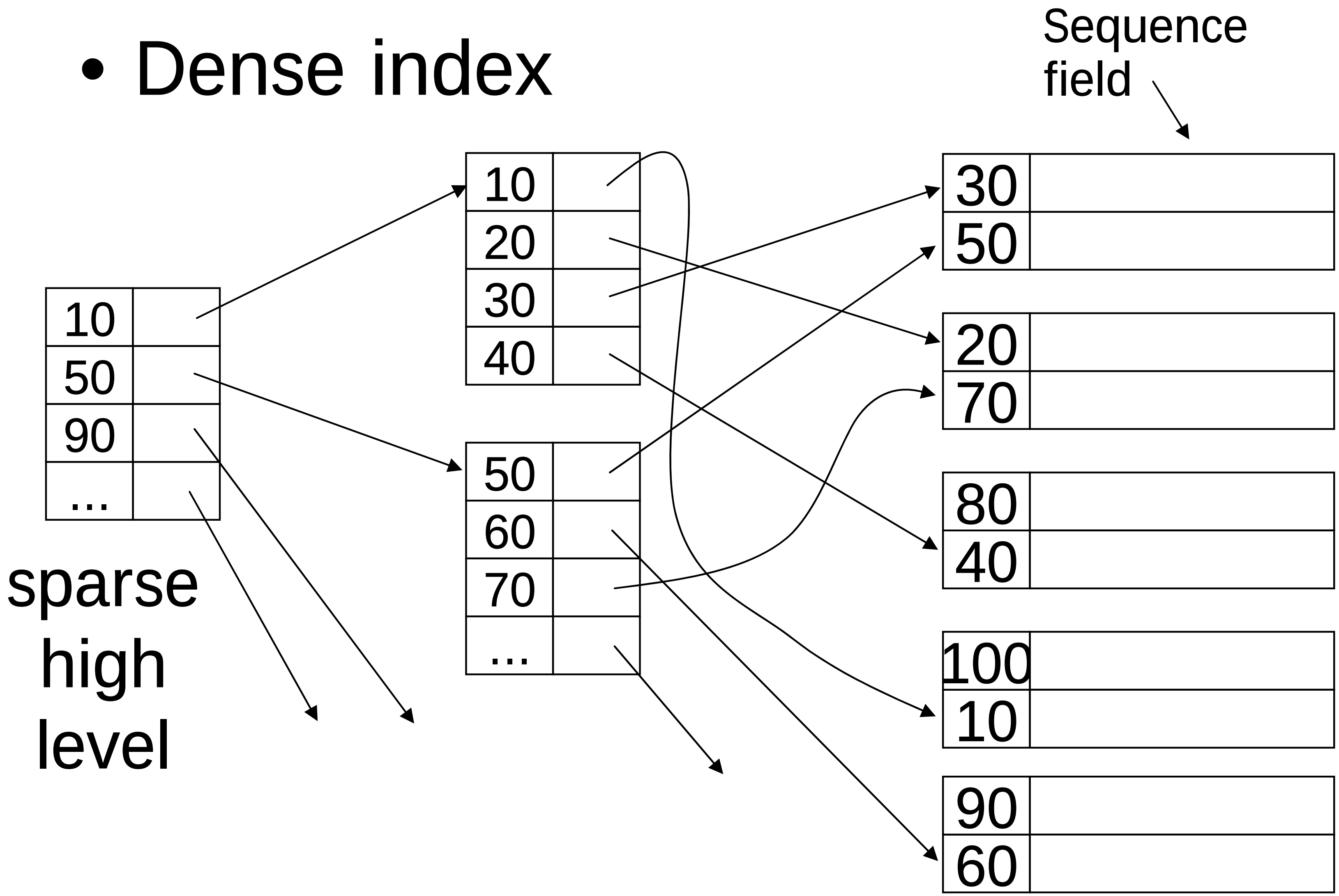
Secondary Indexes

- Dense index



Secondary Indexes

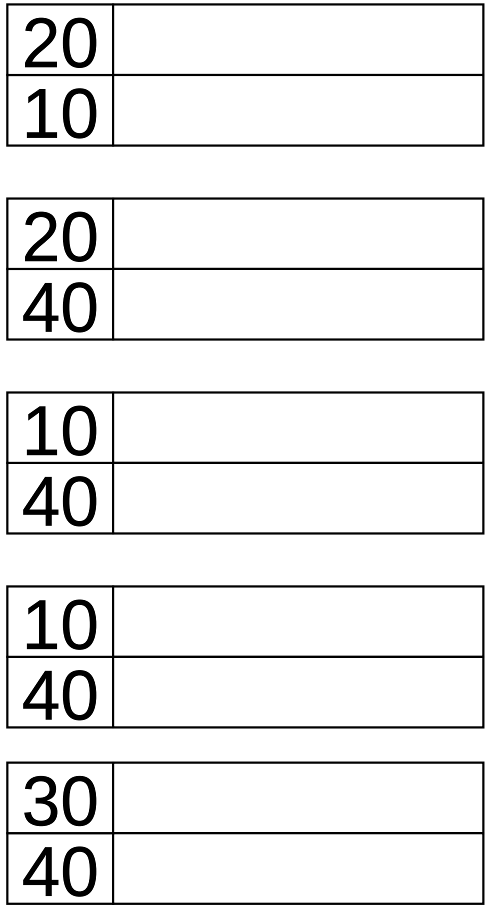
- Dense index



With Secondary Indexes:

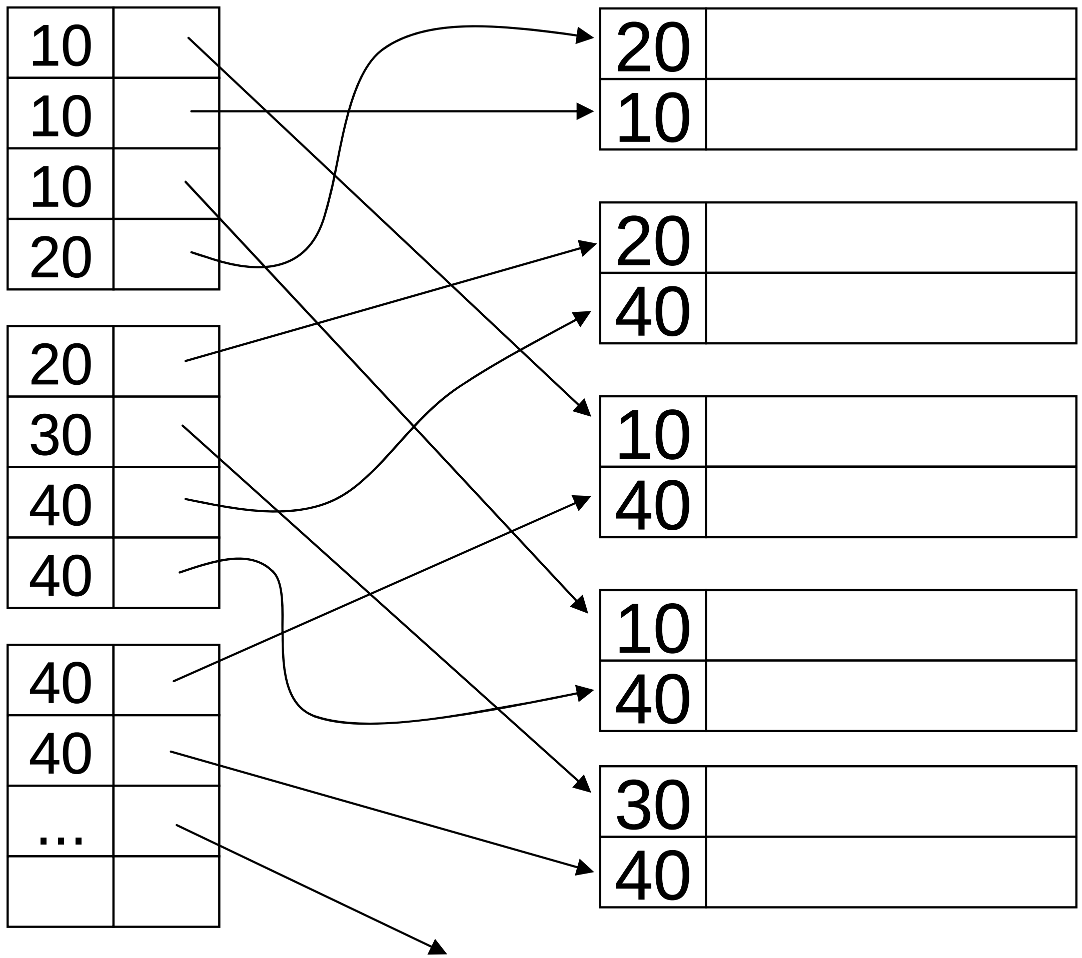
- Lowest level is dense
- Other levels are sparse

Duplicate values & secondary indexes



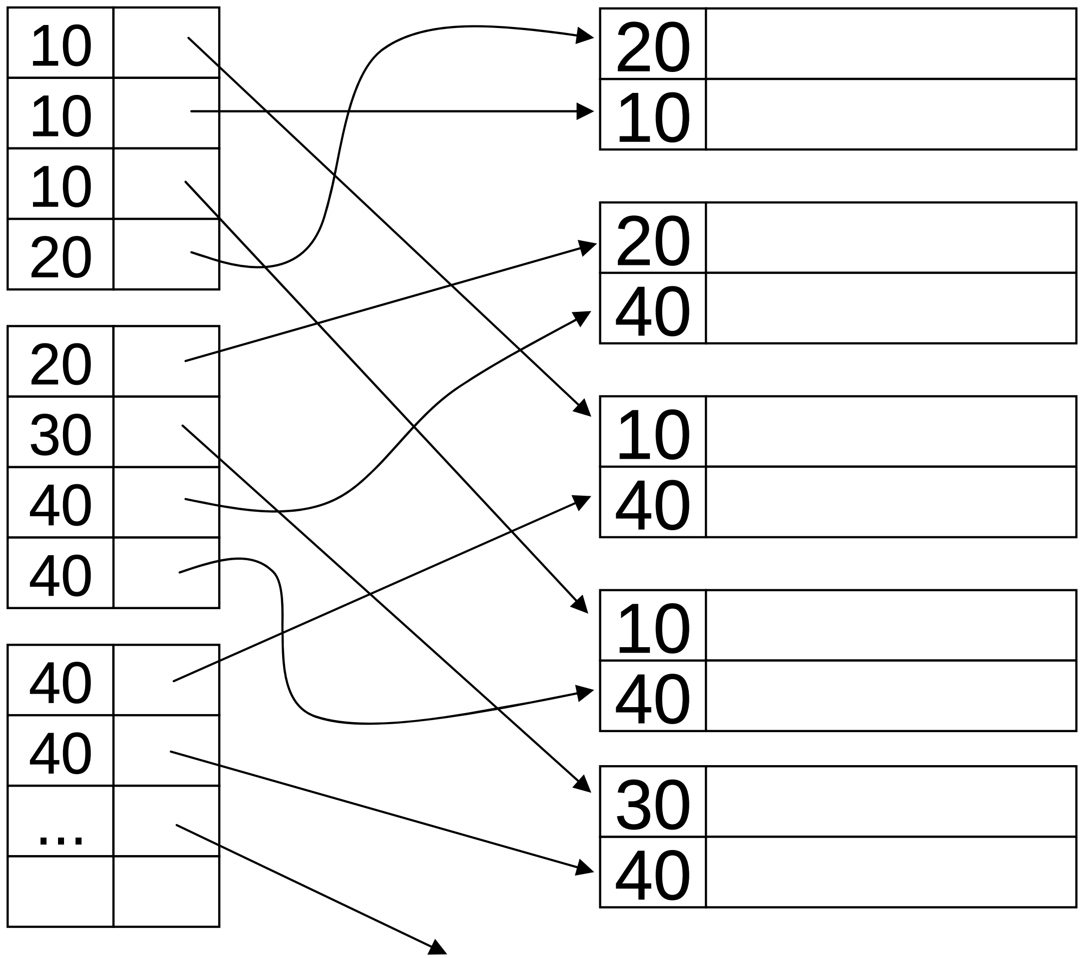
Duplicate values & secondary indexes

one option...



Duplicate values & secondary indexes

one option...



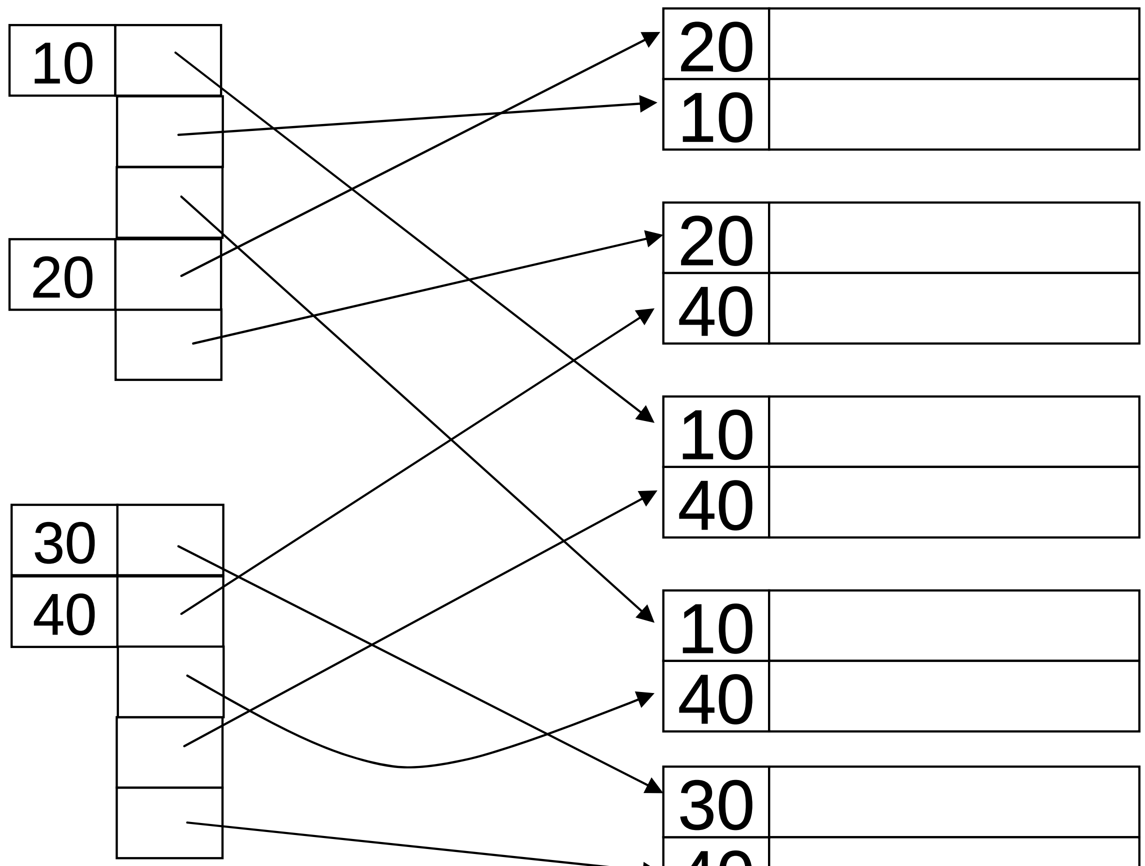
Problem:

excess overhead!

- disk space
- search time

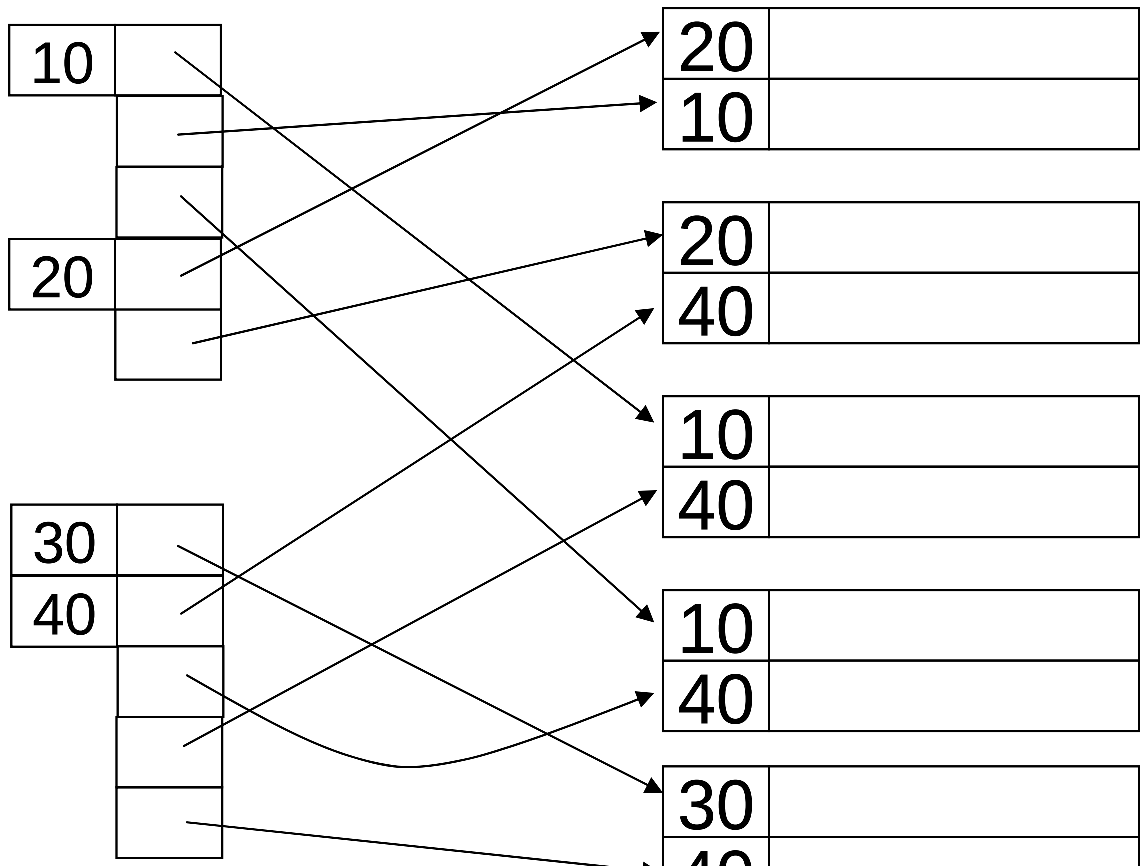
Duplicate values & secondary indexes

another option...



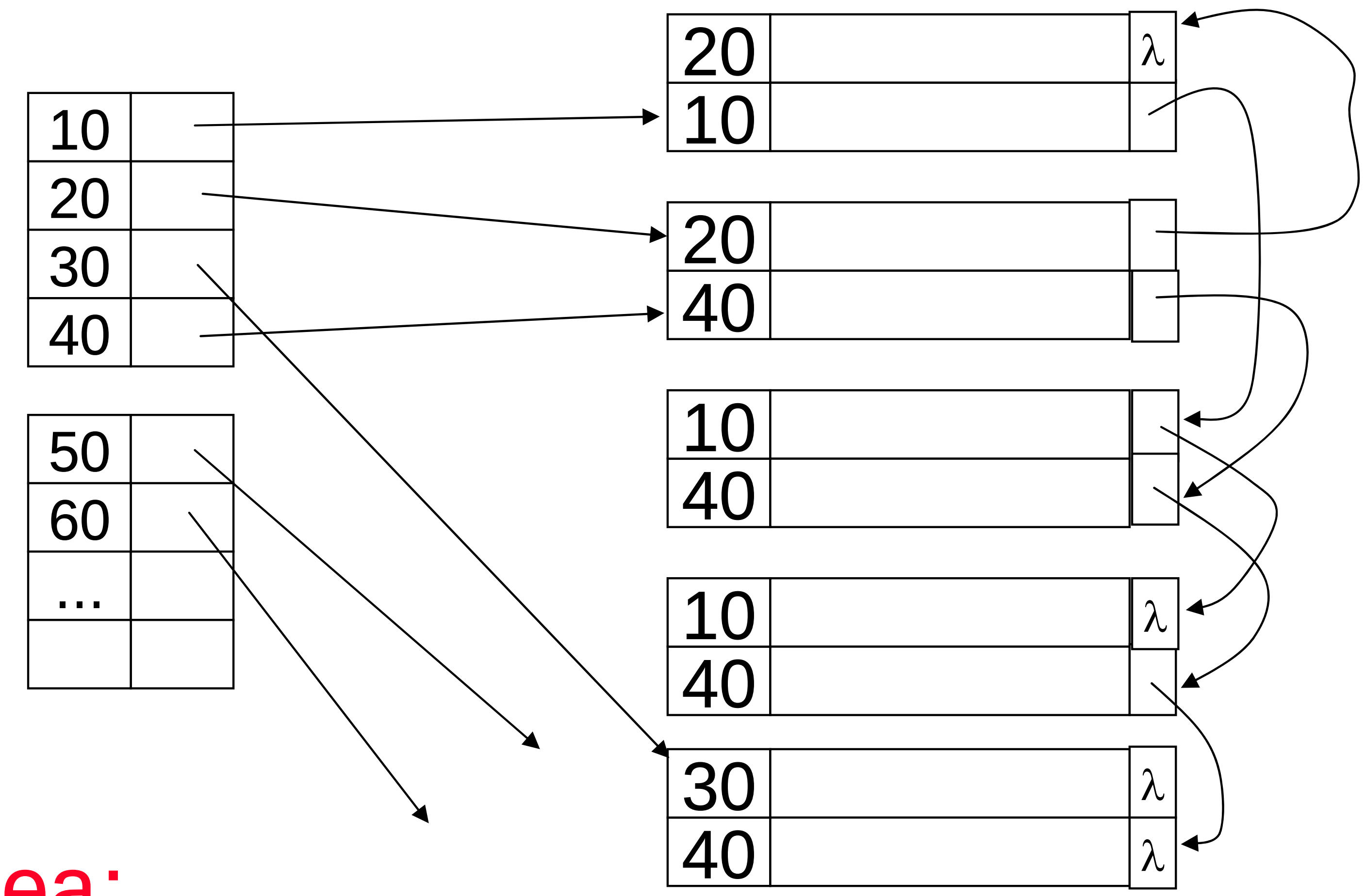
Duplicate values & secondary indexes

another option...



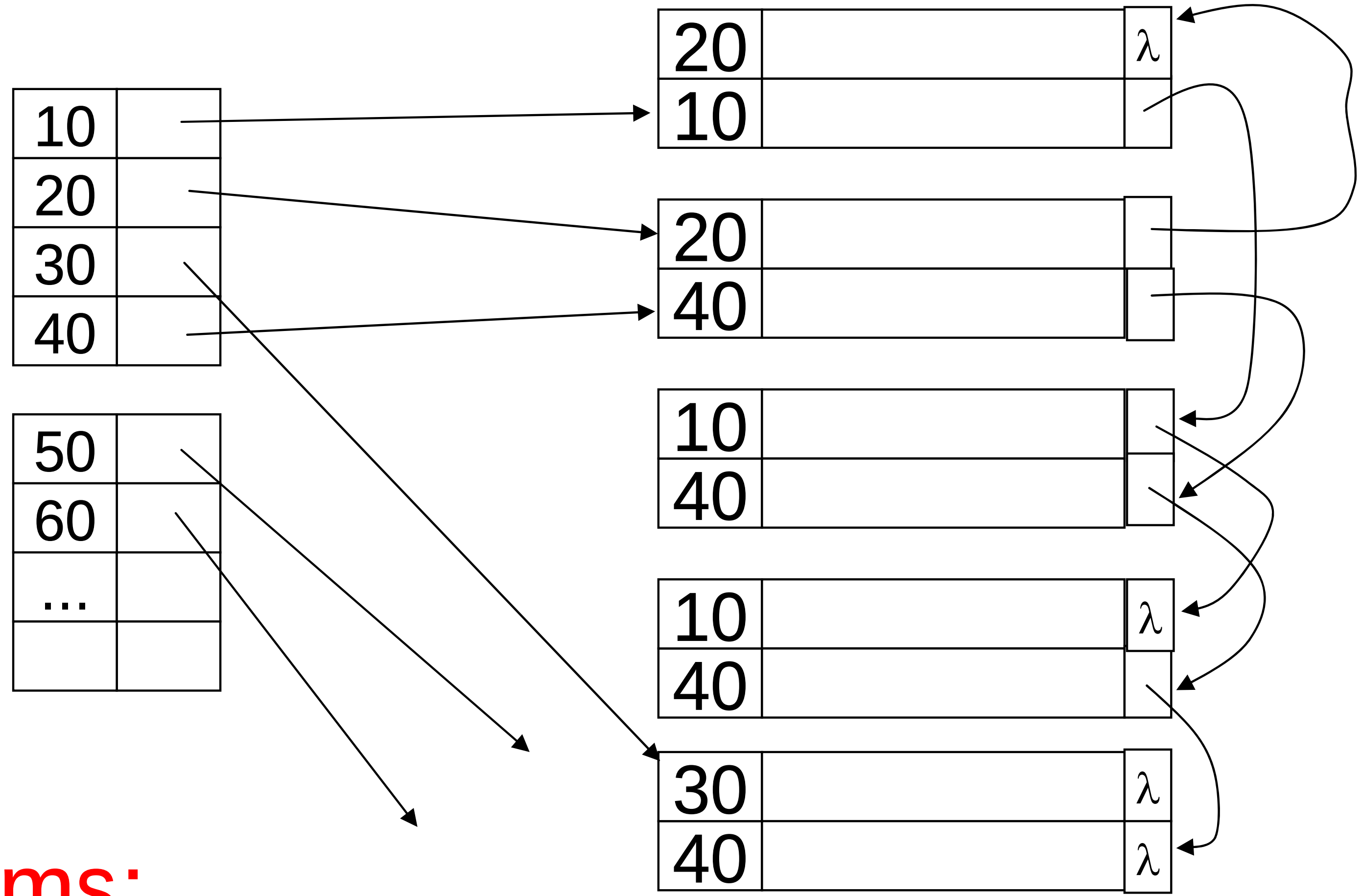
<u>Problem:</u> variable size records in index!

Duplicate values & secondary indexes



Another idea:
Chain records with same key?

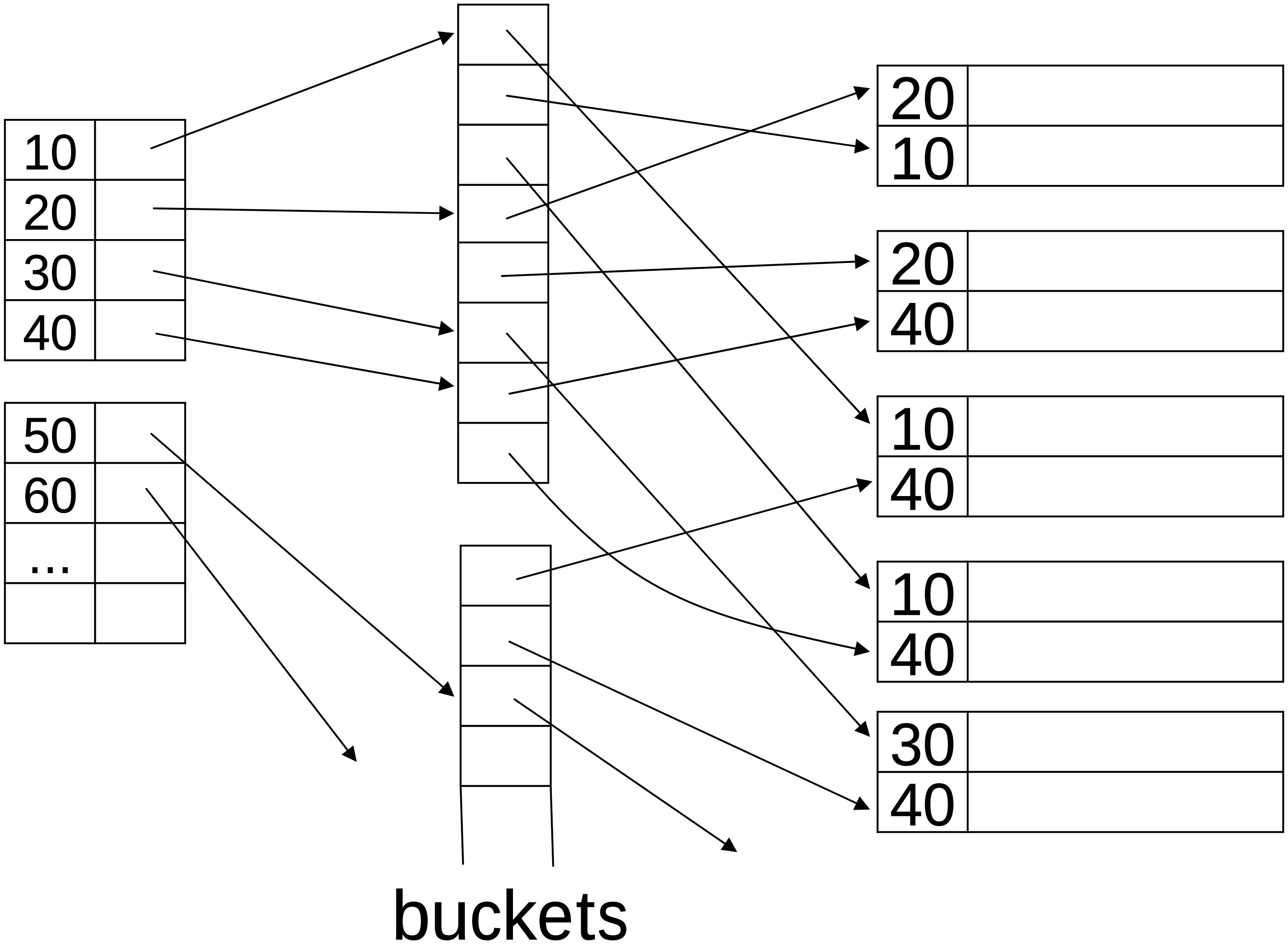
Duplicate values & secondary indexes



Problems:

- Need to add fields to records
- Need to follow chain to know records

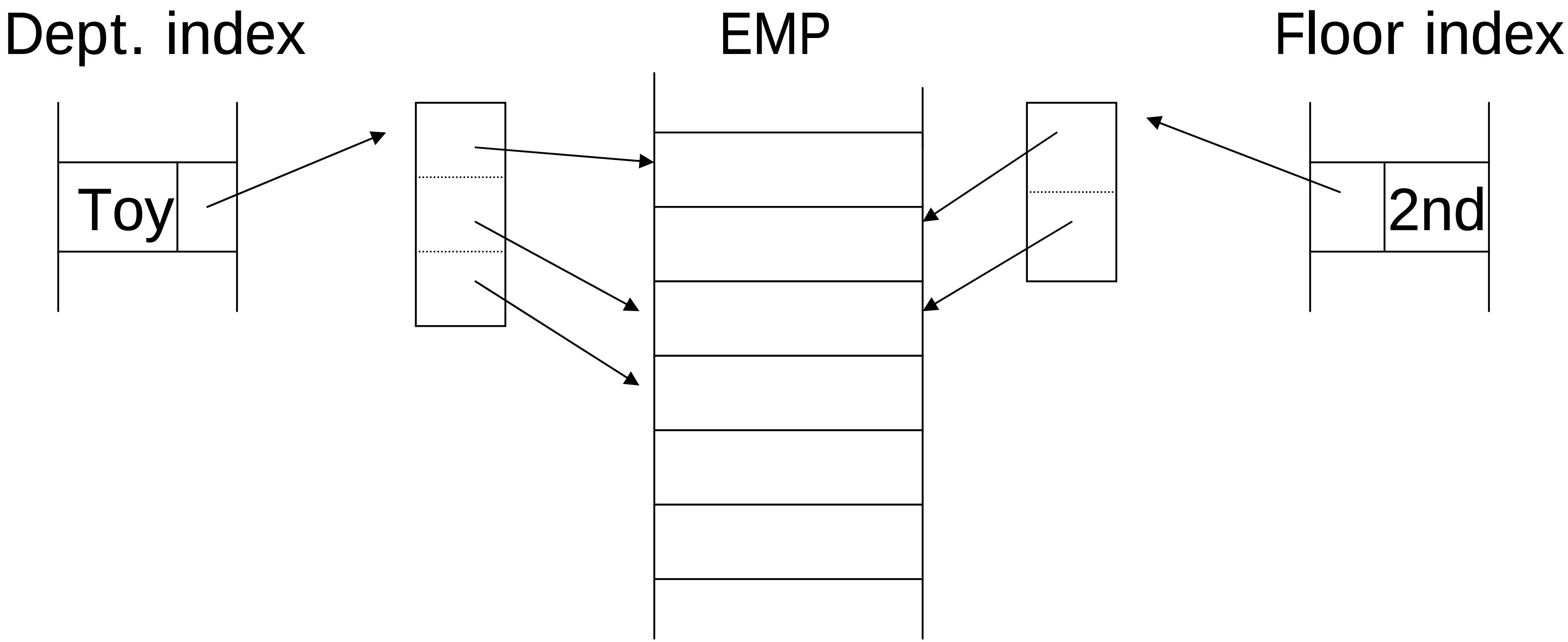
Duplicate values & secondary indexes



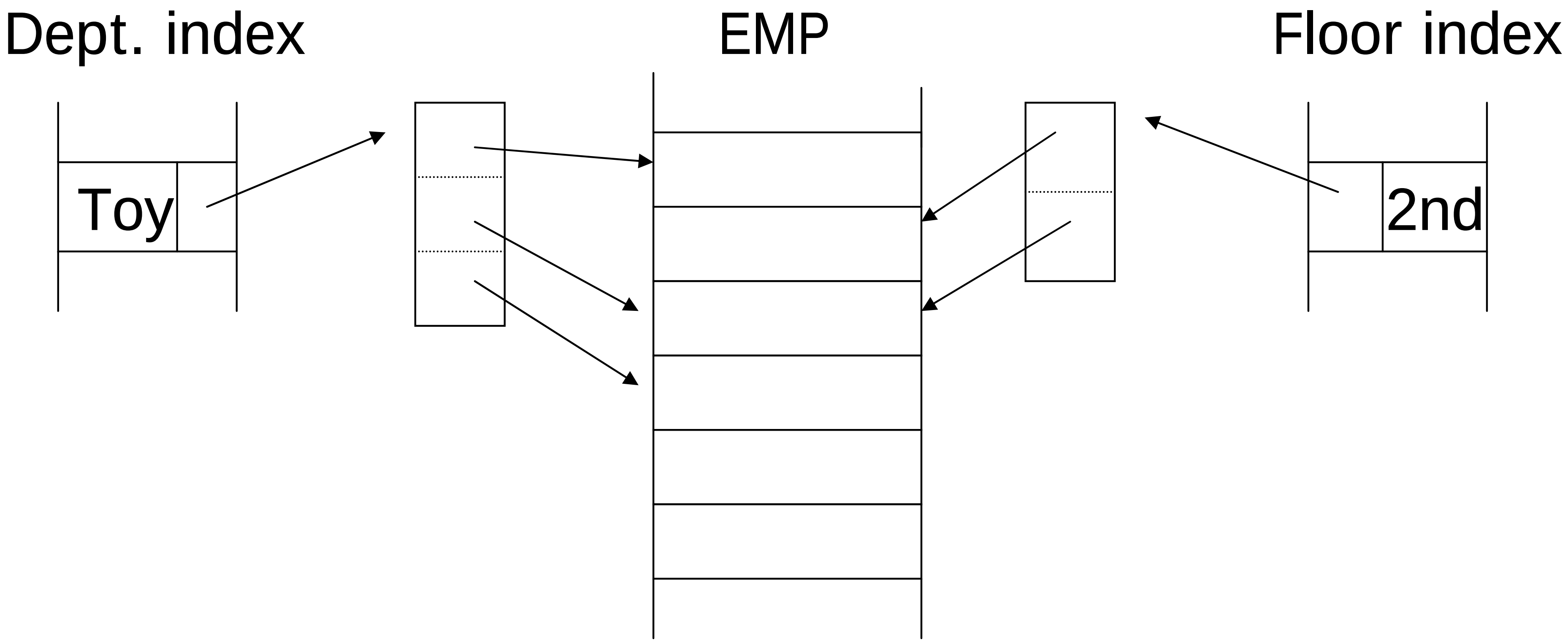
Why "bucket" idea is useful

Indexes	Records
Name: primary	EMP (name,dept,floor,...)
Dept: secondary	
Floor: secondary	

Query: Get employees in (Toy Dept) ∧ (2nd floor)



Query: Get employees in (Toy Dept) ∧ (2nd floor)



→ Intersect toy bucket and 2nd Floor bucket to get set of matching EMP's

This idea used in text information retrieval

Documents

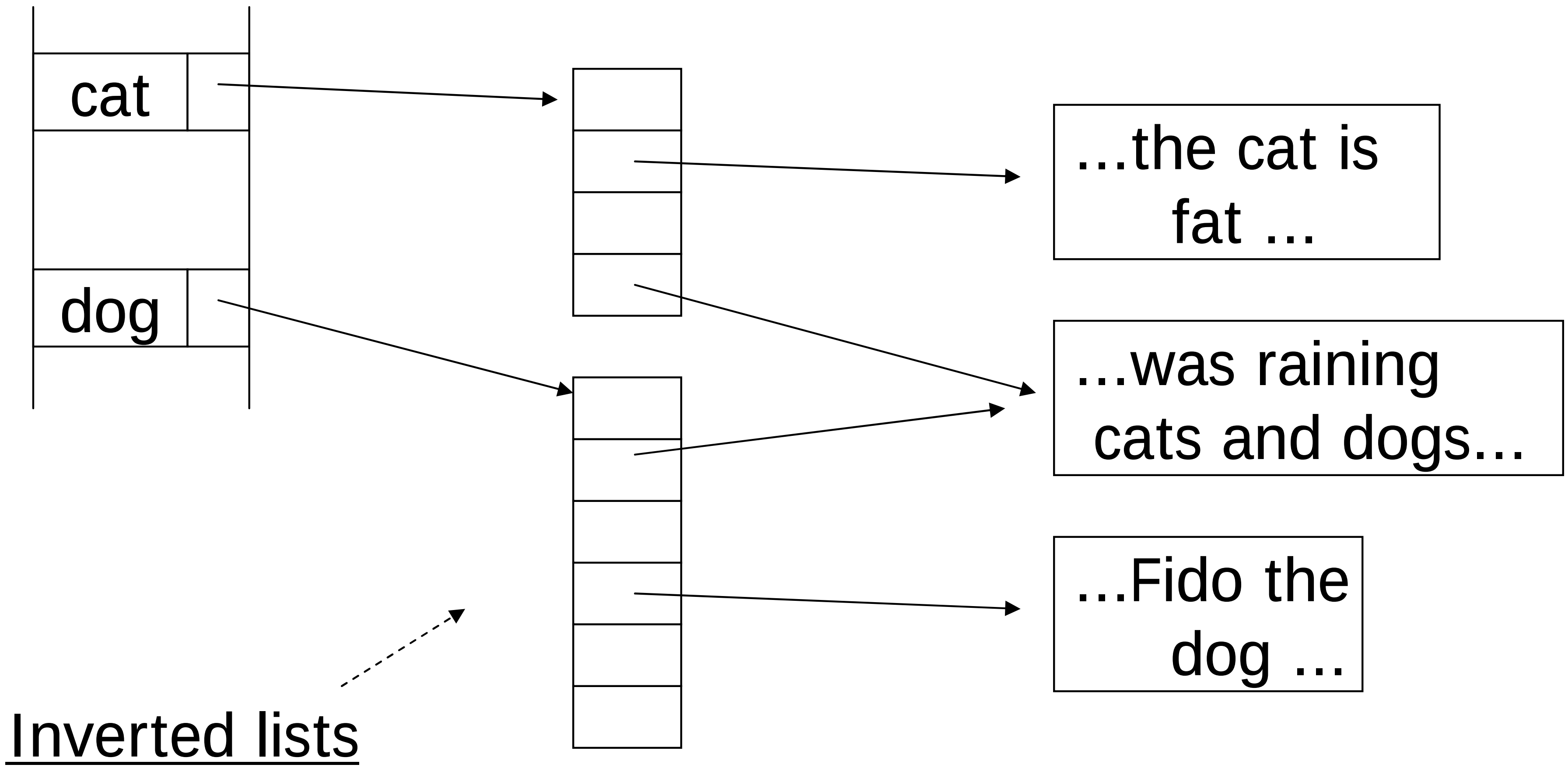
...the cat is fat ...

...was raining cats and dogs...

...Fido the dog ...

This idea used in text information retrieval

Documents



IR QUERIES

- Find articles with “cat” and “dog”
- Find articles with “cat” or “dog”
- Find articles with “cat” and not “dog”