Intro to Index (on files)

- Suppose we have a relation R
  - → stored in a number of blocks on disk.
- User query:

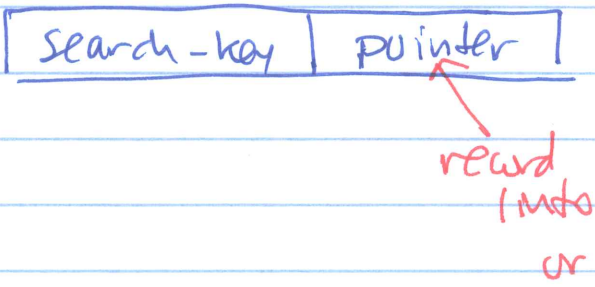
```
select *
from R
where a = 10
```

- Without further info., the ONLY way to process this query is:
  - (1) read all blocks of the relation R
  - (2) Look for tuples with attr. value $a = 10$.
- Index:
  - additional info. on specific attribute(s) to speed up lookup of tuples.

- **Definition:**

Search key = field(s) used to create the index.

index file = file that store records of this form:



(NB: size(index file) << sizeof(data file)).

(several orders)

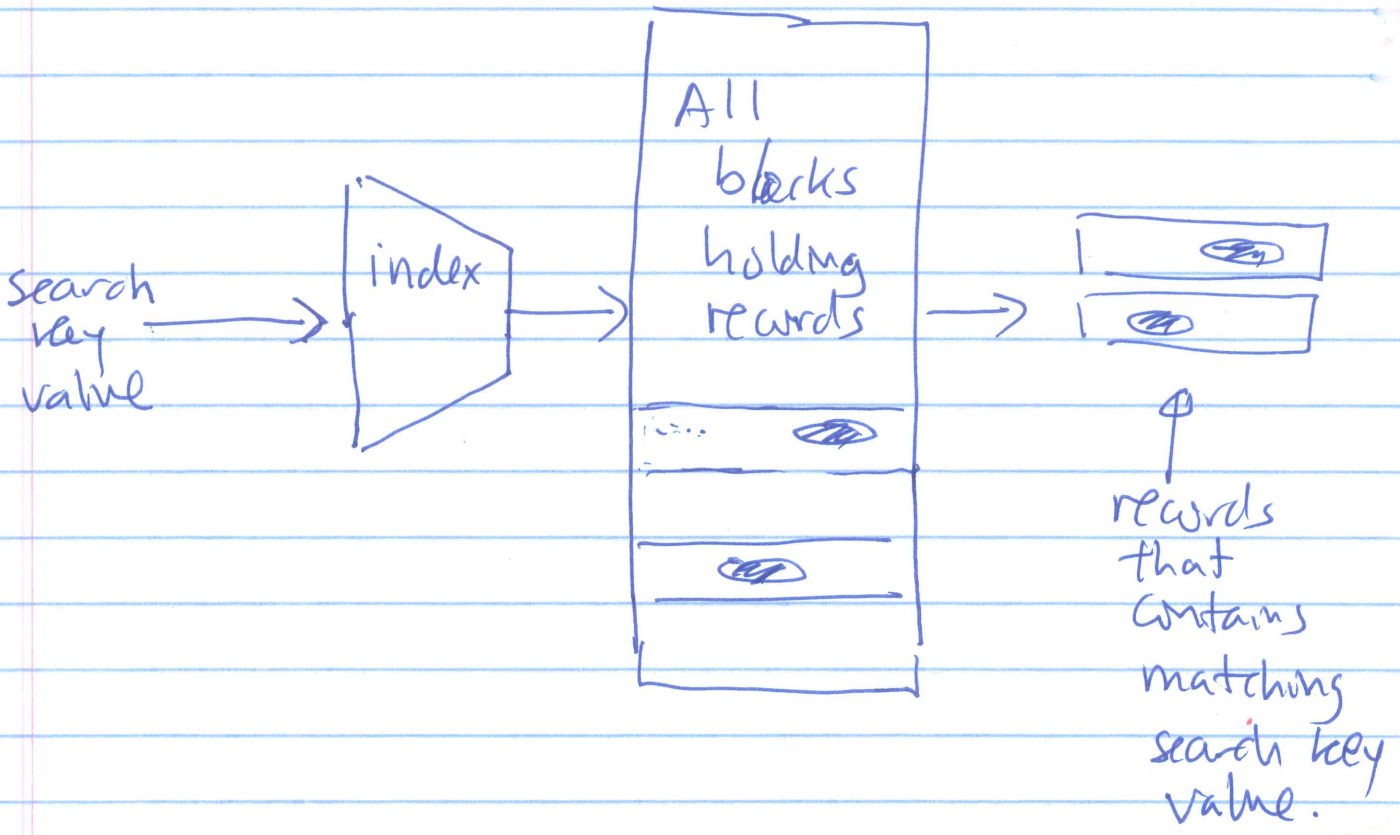
- **2 Basic kinds of indices:**

(1) Ordered indices (search keys stored in ~~ordered~~ sorted order)

(2) Hash indices. (search keys are hashed into buckets).

(Hash Tables).

Operation / Effect of an index:



Index-structure Basics

Data File:

Usually: primary key.
↓ (sort key).

10		A	
20		D	

30		B	
40		A	

50		E	
60		B	

70		B	
80		E	

90		C	
100		F	

↑

sequential file

- Sequential file = a file containing records (tuples) that are sorted by some attribute(s) (usually the primary key).

  Tuples (records) are stored in the sorted order !!!

  (NB: common practice in sequential files
  (1) - leave room in a block for future tuples.
  (2) - use overflow blocks.

- Sort key = field(s) whose values are used to sort/order tuples in a sequential file.

- **Ordered Index** = an index structure where the index entries:

| search-key | pointer |

are stored SORTED (on the search-key)

in an index file.

(NB: take "sorted" with a gram of salt.
B-tree is an ordered index.
B-tree store the index-entries

non-sequentially
but still, there is an order
(sorted order !).

- **Primary Index** = is an ordered index.

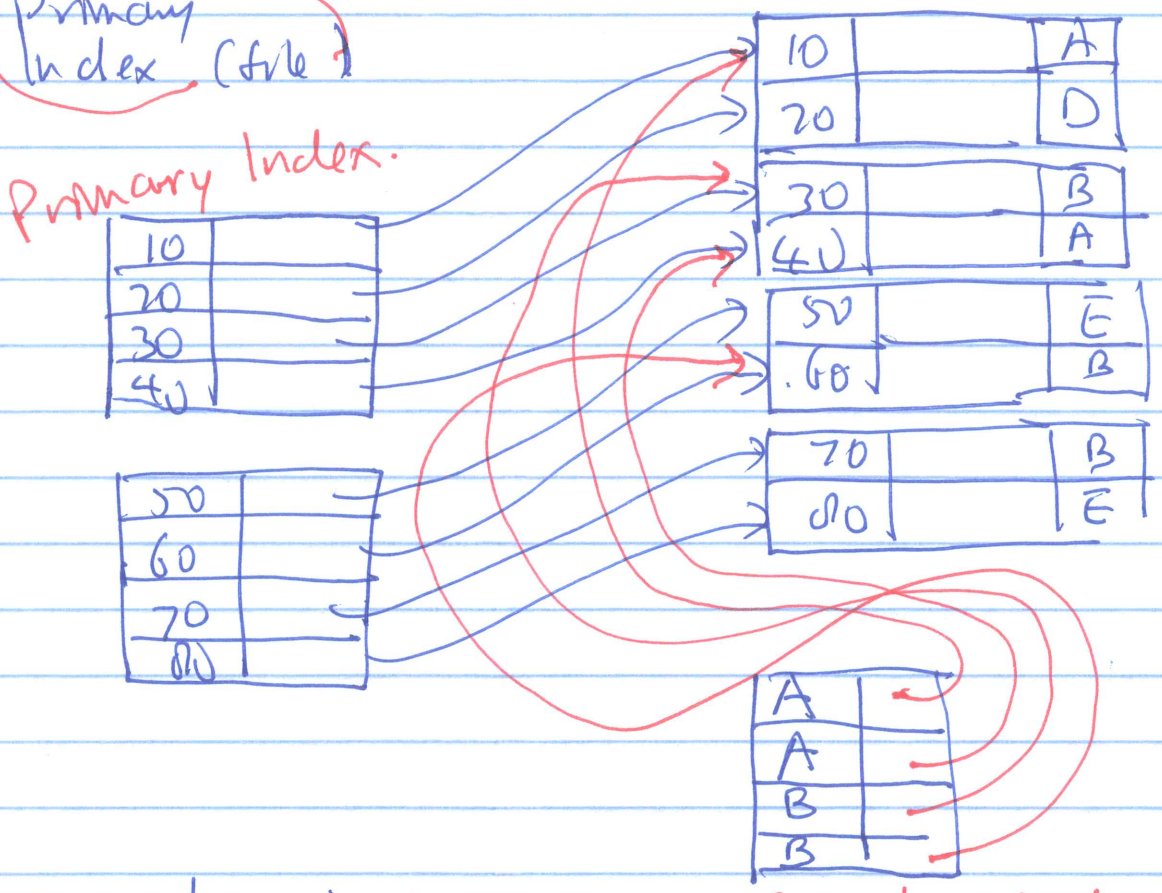
The search key in the index entries:

| search key | pointer |

of a primary index, is ALSO
the sort key used for
the Sequential file.

Example:

Primary Index (file)



- Secondary Index:

The (search key) in the index entries

search key	pointer

of a secondary index, is

NOT a sort key

Index-sequental file =

a sequental file that

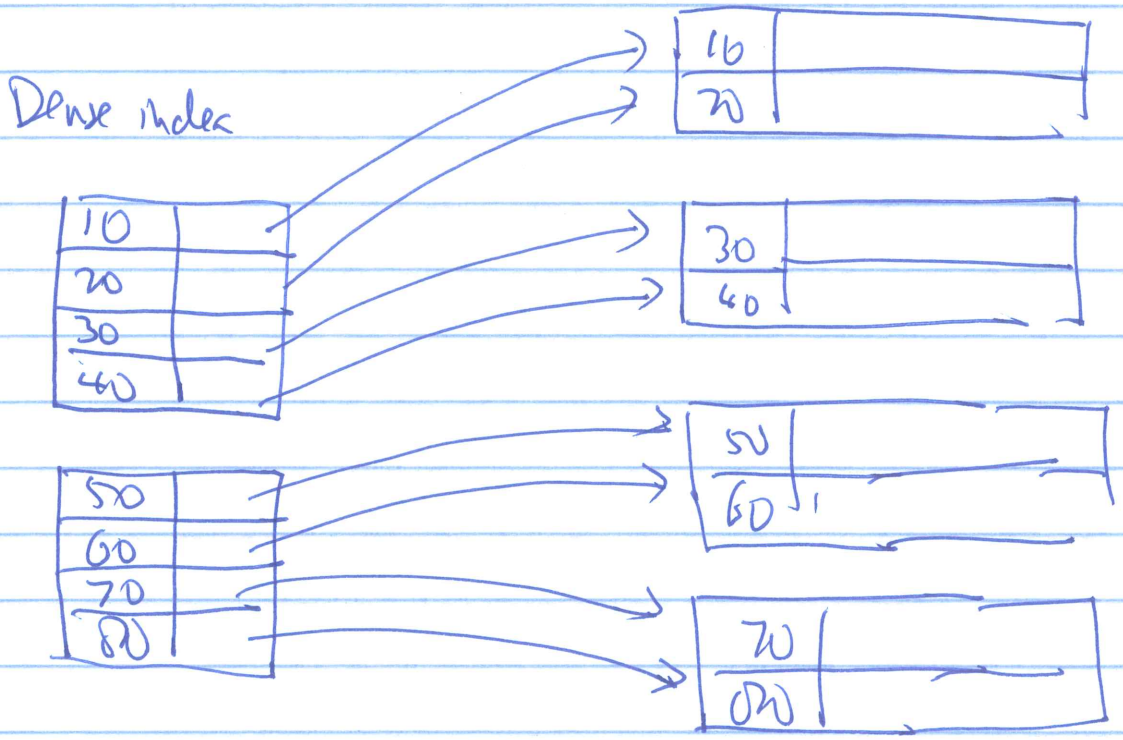
has a primary index

(made pupular by IBM).

- **Dense Index** = The index file contains an index entry:

for every value of the search key in the data file.

eg: "dense index"



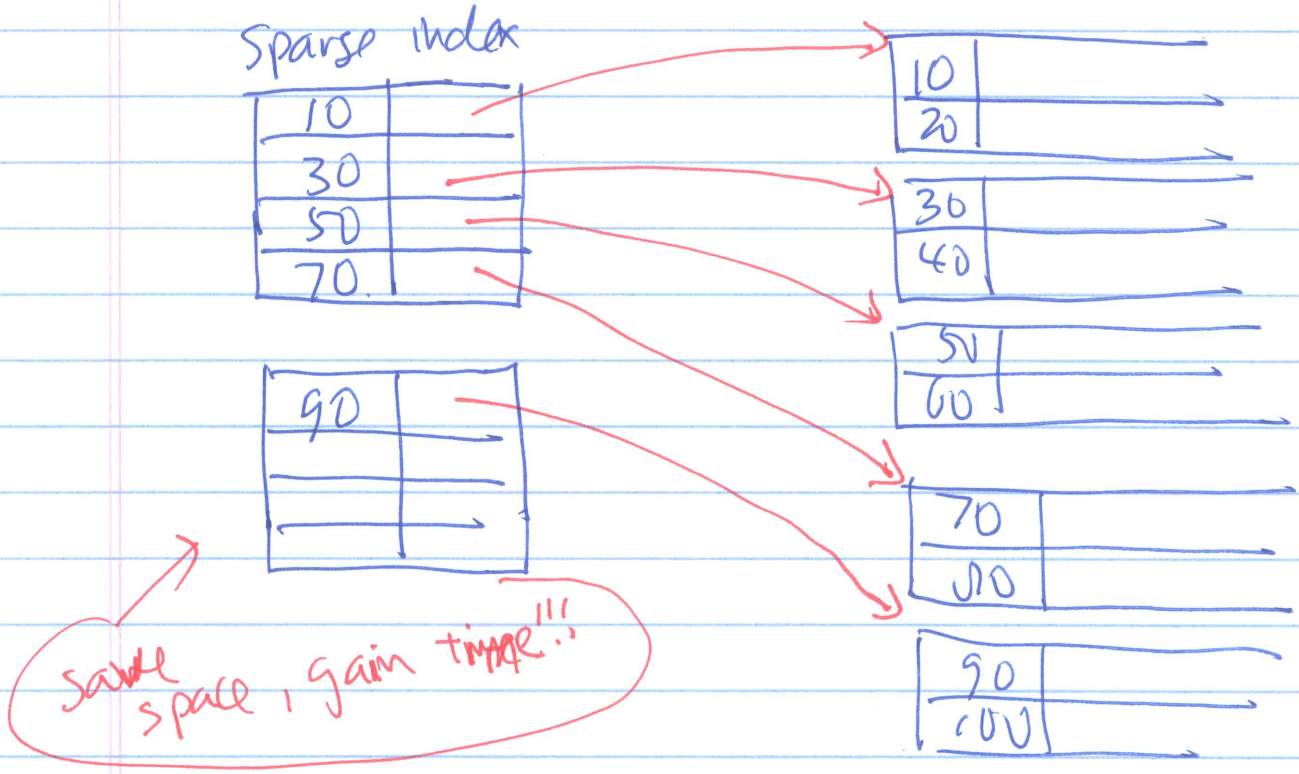
- Note: a secondary index is ALWAYS dense

- Sparse index = The index file's index entry

search-key	pointer

does NOT contain every value of the search key in the data file.

eg:



- NOTE: ONLY the primary index can be Sparse !!!

NOTE: generally slower than dense index for writing records.

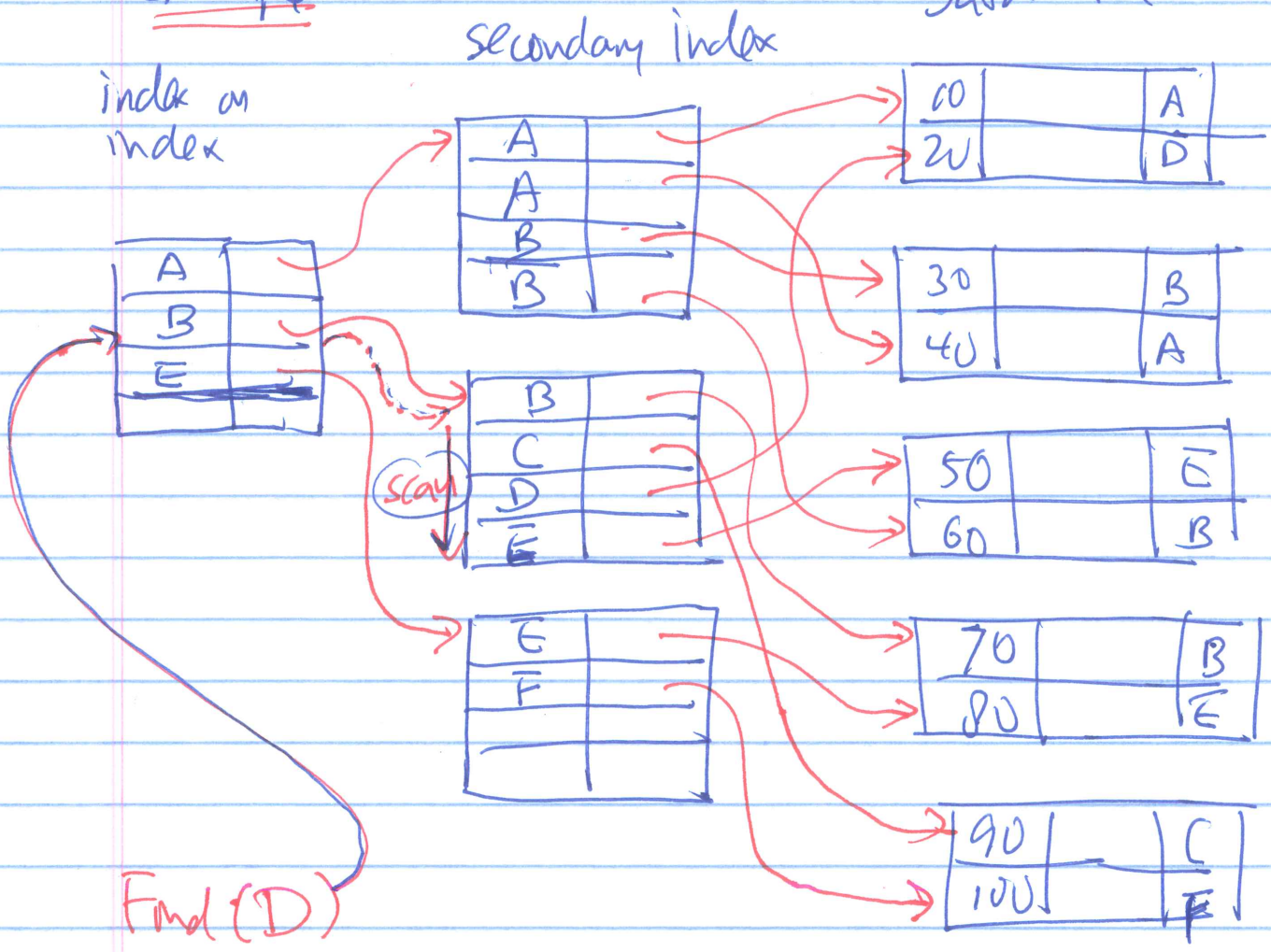
Multi-level Index.

- Fact: An ordered index is a file

  So: We can put a (primary) index on this file

  (to speed up ~~search~~ searching for a search key !!!)

- Example:



Note: Later, we will study a

dynamic multi-level index

data structure

$\Rightarrow$ B-tree.

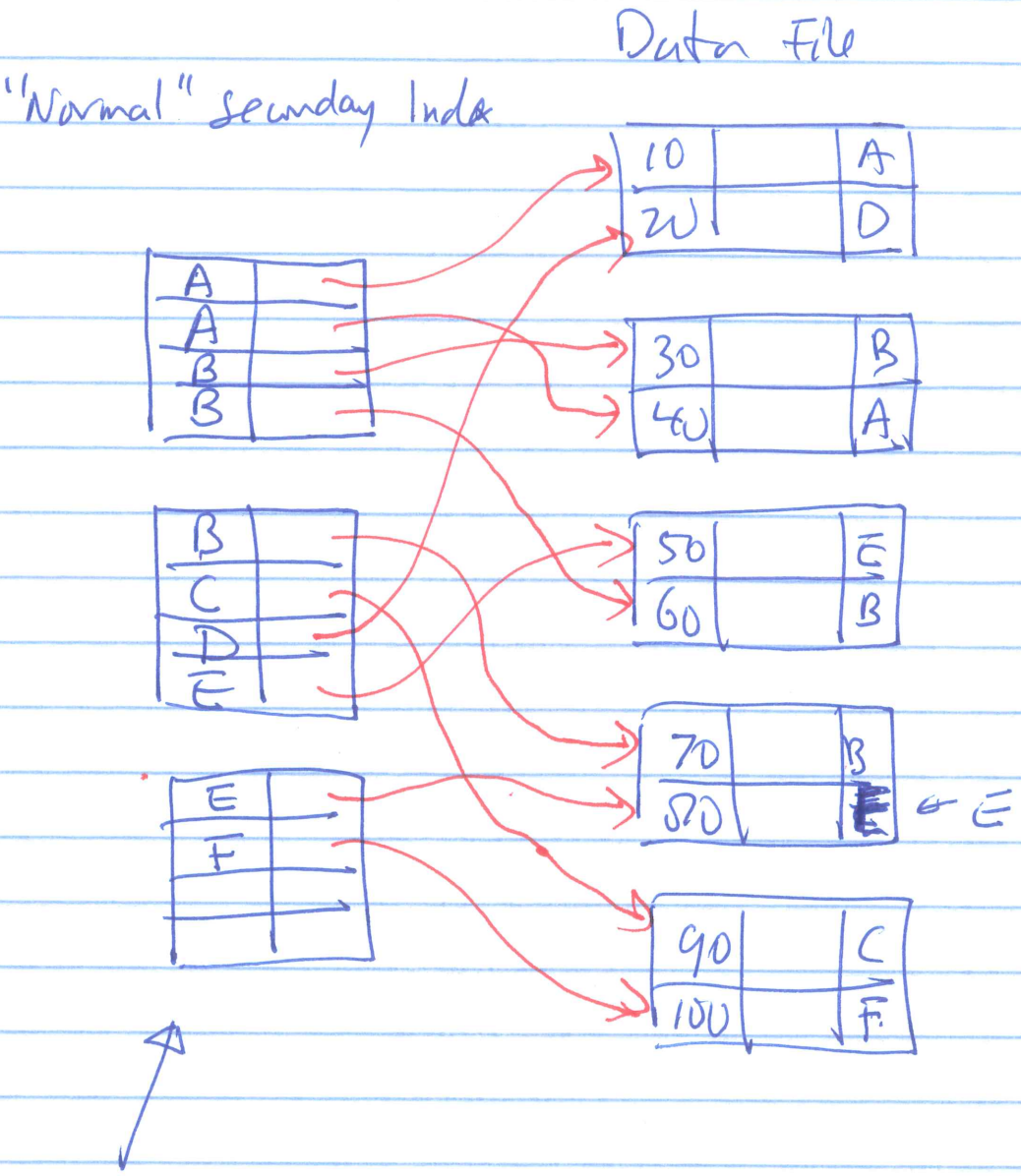
↓

very popular (Most popular)

ordered index structure !!!

Indirection in Secondary Indexes.

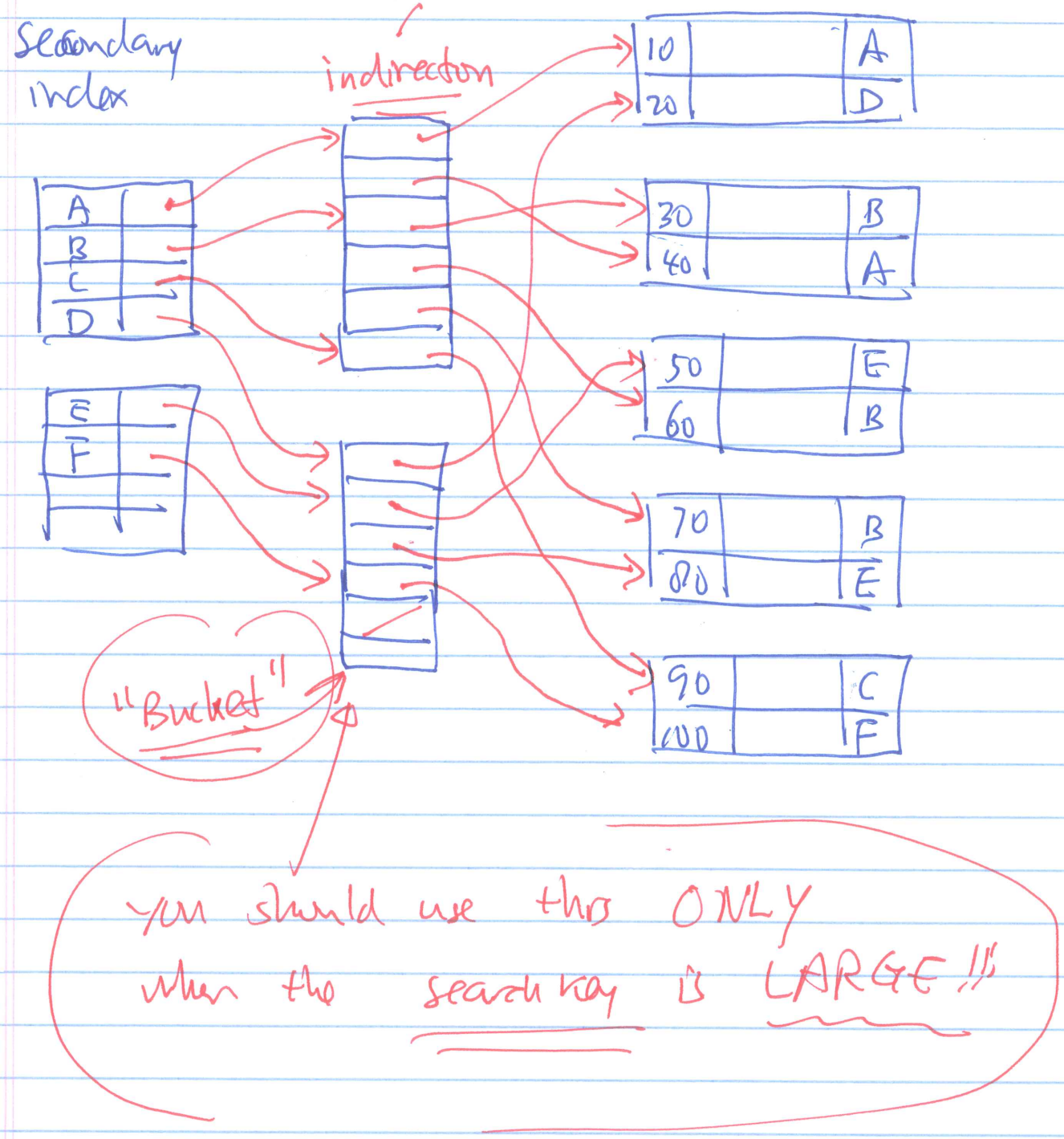
- "Normal" secondary index:



A search key that appears n times in the Data file will appear n times in the index file.

Bucket

Secondary index with "indirection":



you should use this ONLY when the search key is LARGE!!

Application

Quickly

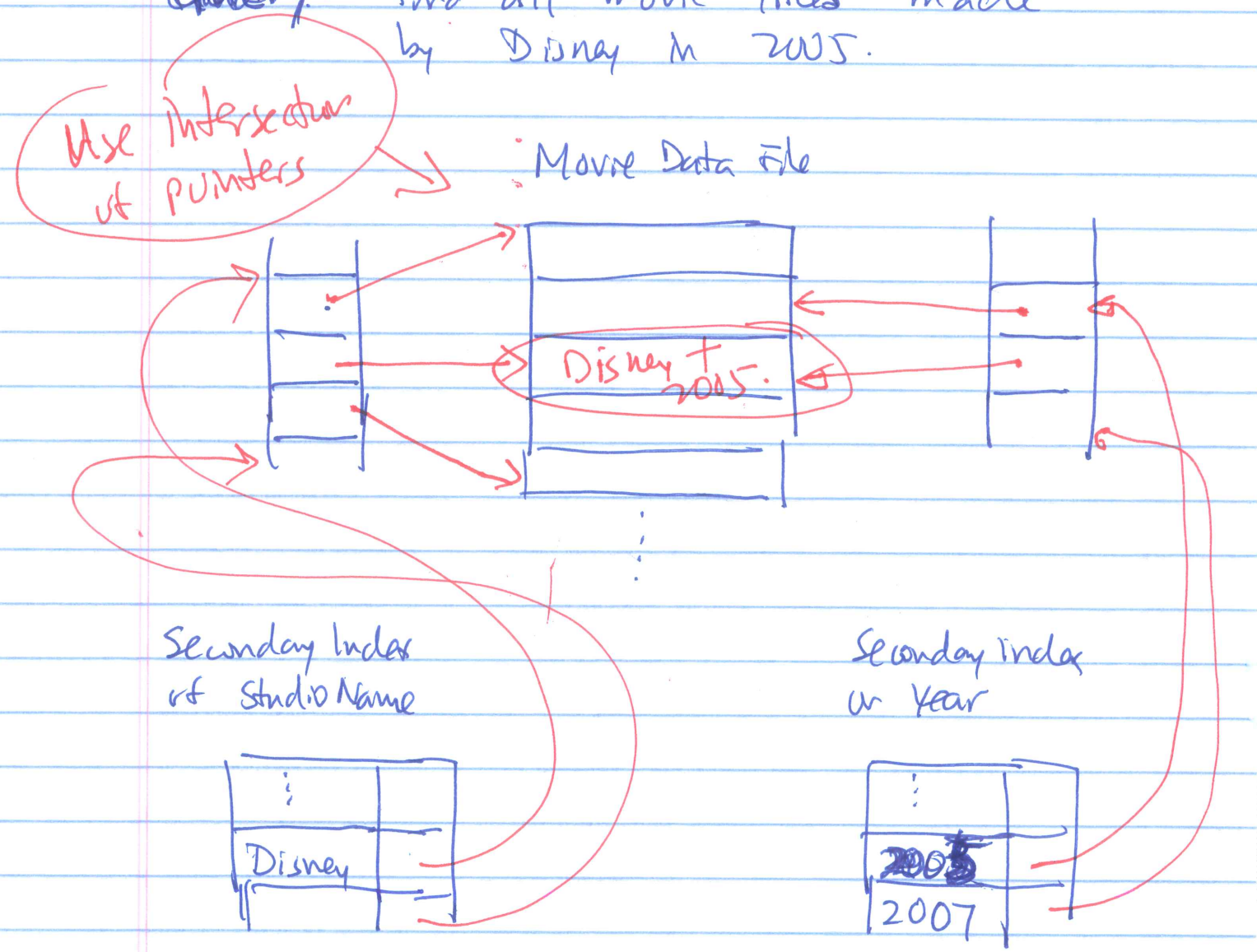
Answering queries using indexes.

- Suppose we have a movie relation:

  Movie (title, year, length, genre, studioName, producer).

- Query: find all movie titles made by Disney in 2005.

Use intersection of pointers



☆ Similar Application: Document retrieval.

- Q: How to find documents (quickly) that contain certain key words.

- Visualize a document as a relation:

  Doc (hasCat, hasDog, ...)

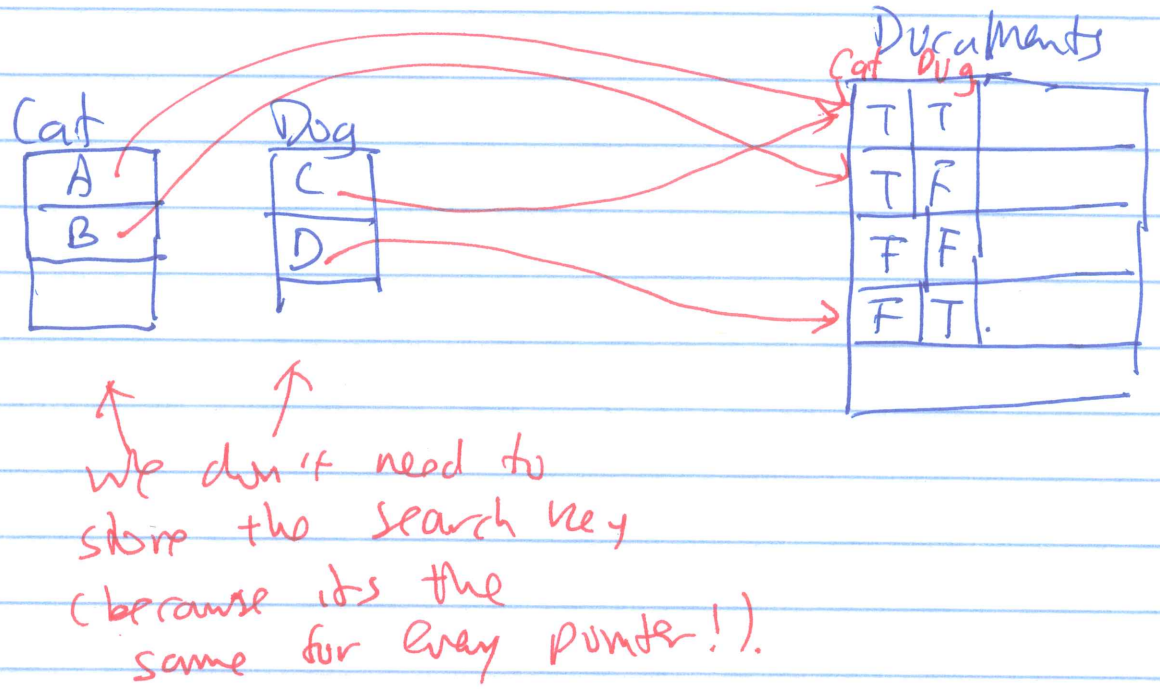
  ↑ all key words your want to index.

Example:

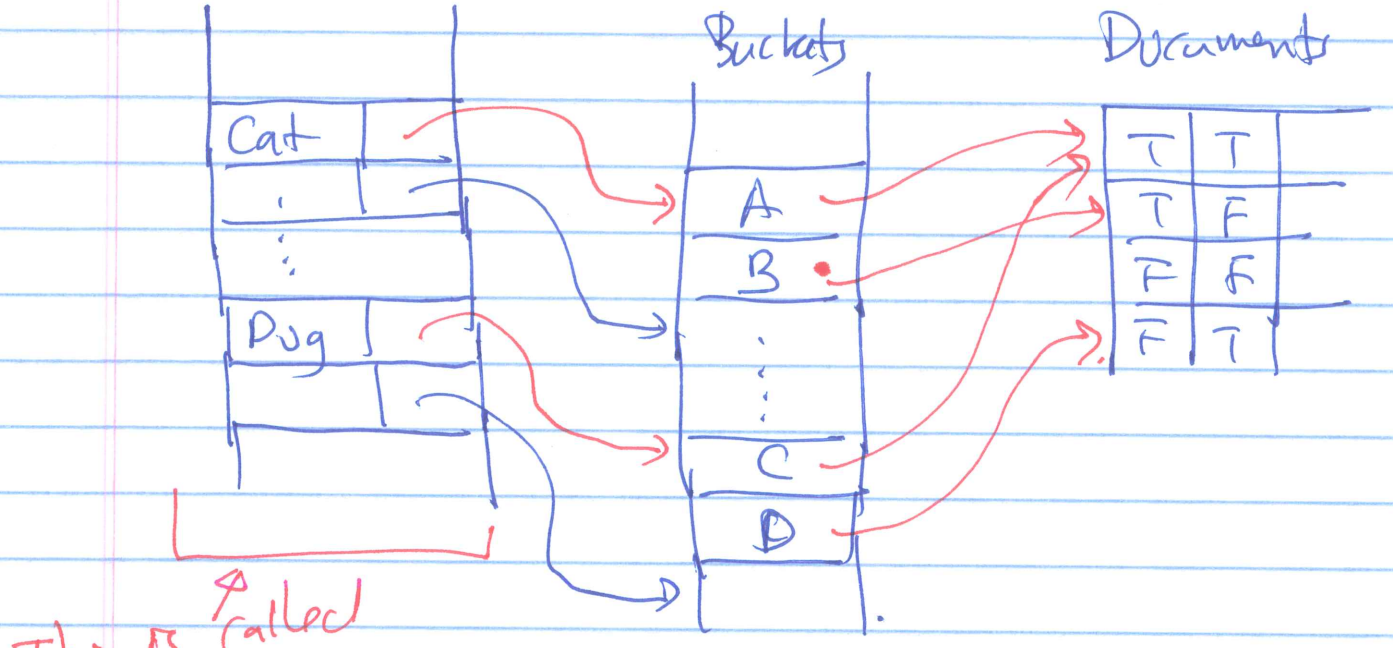
	cat	Dog	...
doc1	T	T	...
doc2	T	F	
doc3	F	F	...

Key: we DO NOT index tuples that contains the FALSE value.

First Iteration: one index file for each attr.



Better: combine indices with a second level.



More informative Bucket file

- When indexing HTML/XML documents, the occurrence of the ~~word~~ word has a context associate with it.

- This context (tag) can be stored in the Bucket file: