Index Structures

What is an Index?

An index is a data structure that allows us to directly locate units of data based on certain values

- Not just used for databases: also books can contain an index

Indexes for databases are used to find records that have a particular value for the indexed attribute (the “search key”)

An index has to be created before it can be used

- creation often initiated by the database designer
- cost of maintenance

Different categories exist

- primary / secondary indexes
- dense / sparse indexes

Sequential Files

Records ordered by search key (may not be "key" in DB sense).

- facilitates queries on the search key

Blocks containing records therefore ordered.

- physically contiguous
- chained

On insert: put record in appropriate block if room.

- Good idea: initialize blocks to be less than full; reorganize periodically if file grows.

If no room in proper block:

- 1. Create new block; insert into proper order if possible.
- 2. If not possible, create overflow block, linked from original block.

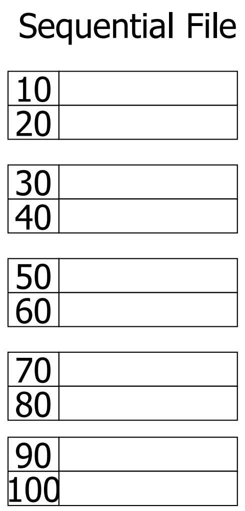
Indexes

Dense Indexes: Pointer to every record of file, ordered by search key.

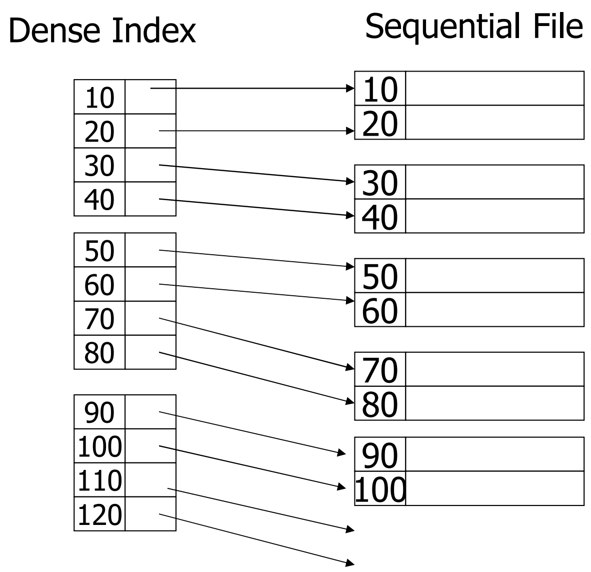
- Can make sense because records may be much bigger than key-pointer pairs.
  - If index requires fewer blocks faster search through index than data file
  - Index might fit in memory, even if data file does not
- Test existence of record without going to data file.

Sparse Indexes: Keypointer pairs for only a subset of records, typically first in each block.

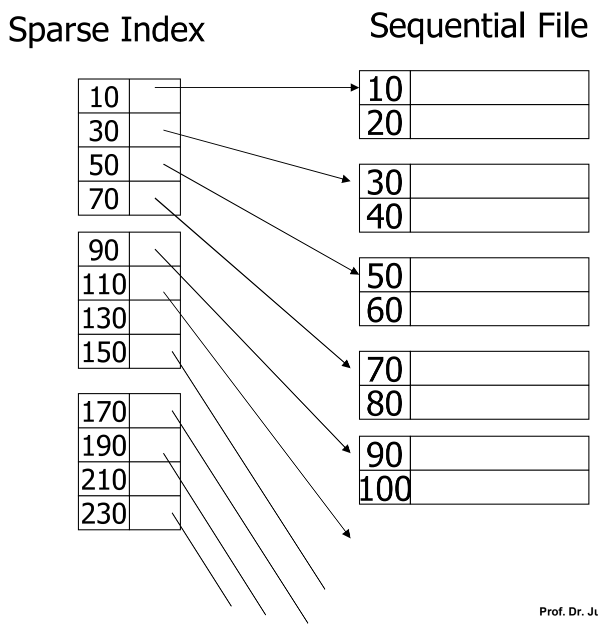
Example: Sequential File



Example: Dense Index



Example: Sparse Index



Sparse vs. Dense Index

Sparse: Less index space per record can keep more of index in memory

Dense: Can tell if any record exists without accessing file

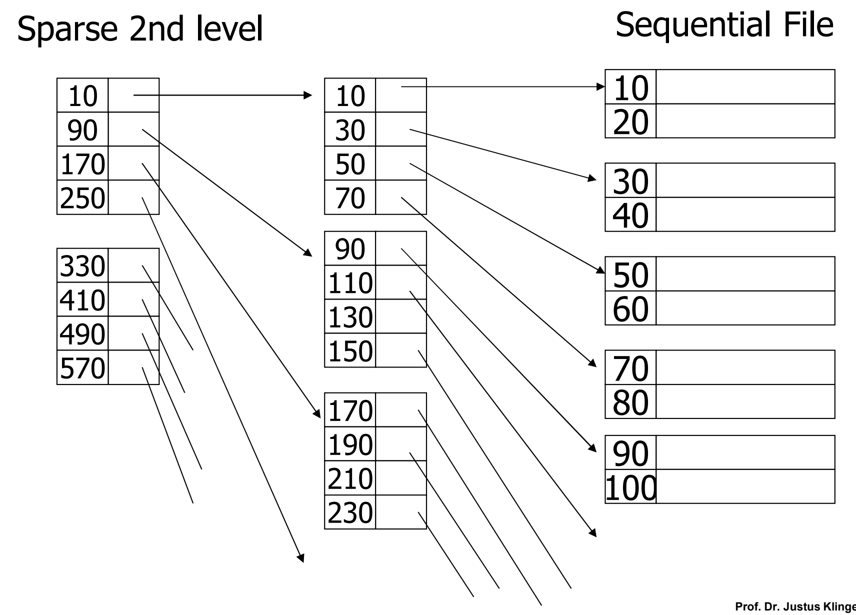
Multiple Levels of Index

A sparse index on a (sparse or dense) index is an option.

Good chance that 2nd or higher level indexes can be housed in main memory, so no additional disk I/O's.

Dense higher level indexes make no sense;

Example: Second Level Index



DB Modifications

When we insert or delete on the data file, here are the primitive actions we might take:

1. Create or destroy an overflow block.
2. Create or destroy an empty block in the sequence of blocks belonging to the sequential file.
3. Insert a record into a block that has room.
4. Delete a record.
5. Slide a record to an adjacent block.

Effect of Primitive Actions on Index File

Action	Dense	Sparse
Create/destroy empty overflow block	none	none
Create empty seq. block	none	insert
Destroy empty seq. block	none	delete
Insert record	insert	update(?)
Delete record	delete	update(?)
Slide record	update	update(?)