

## 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 (what if blocks are consecutive around a track for efficiency?).
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

Sequential File

10	
20	
30	
40	
50	
60	
70	
80	
90	
100	

## Example: Dense Index

Dense Index

10		→	10	
20		→	20	
30		→	30	
40		→	40	
50		→	50	
60		→	60	
70		→	70	
80		→	80	
90		→	90	
100		→	100	
110		→	100	
120		→	100	

## Example: Sparse Index

Sparse Index

Sequential File

10		→	10	
30		→	20	
50		→	30	
70		→	40	
90		→	50	
110		→	60	
130		→	70	
150		→	80	
170		→	90	
190		→	100	
210		→	100	
230		→	100	

## 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

Sparse 2nd level

Sequential File

