Storage & Indexing

Abdu Alawini

University of Illinois at Urbana-Champaign

CS411: Database Systems

Thanks to Doris Xi for her help with preparing the slides.

Learning Objectives

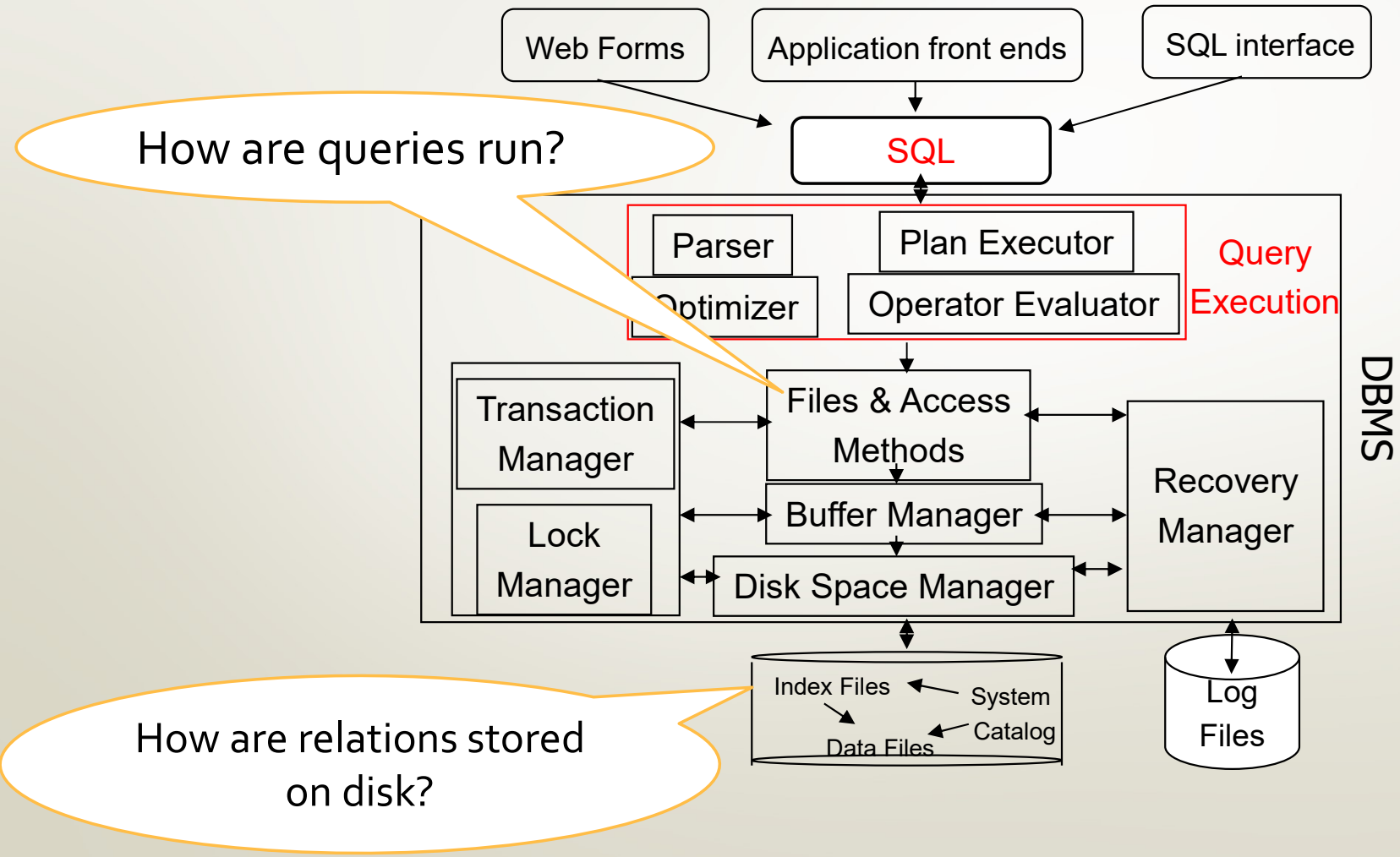
After this lecture, you should be able to:

- describe how relations are stored on disk
- describe how index structures speed up data access

CS411 Goals:
Two Perspectives of DBMS

- USER PERSPECTIVE
  - **how to use a database system?**
  - conceptual data modeling, the relational and other data models, database schema design, relational algebra, SQL and No-SQL query languages.
- SYSTEMS PERSPECTIVE
  - **how to design and implement a database system?**
  - data representation, indexing, query optimization and processing, transaction processing, and concurrency control.
  - NOT COMPLETE: high-level view of implementation; CS511

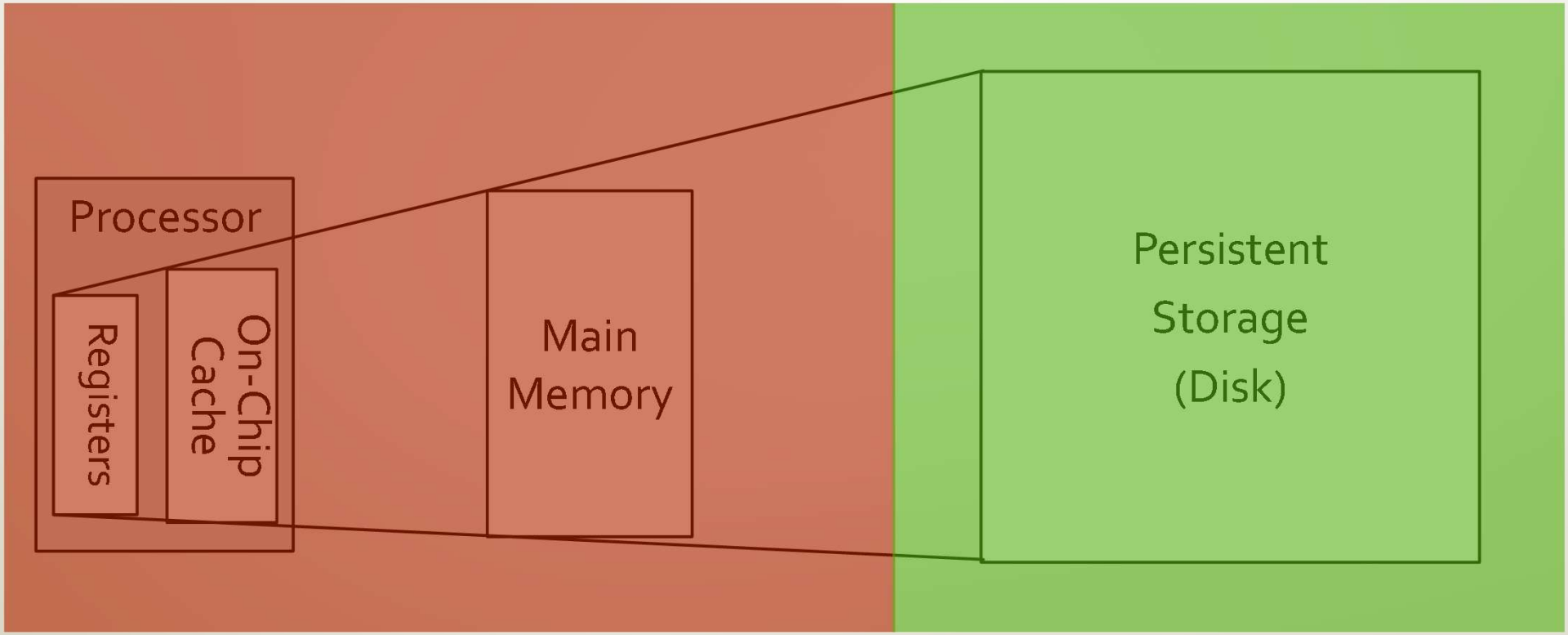
DBMS Architecture



Today's lecture

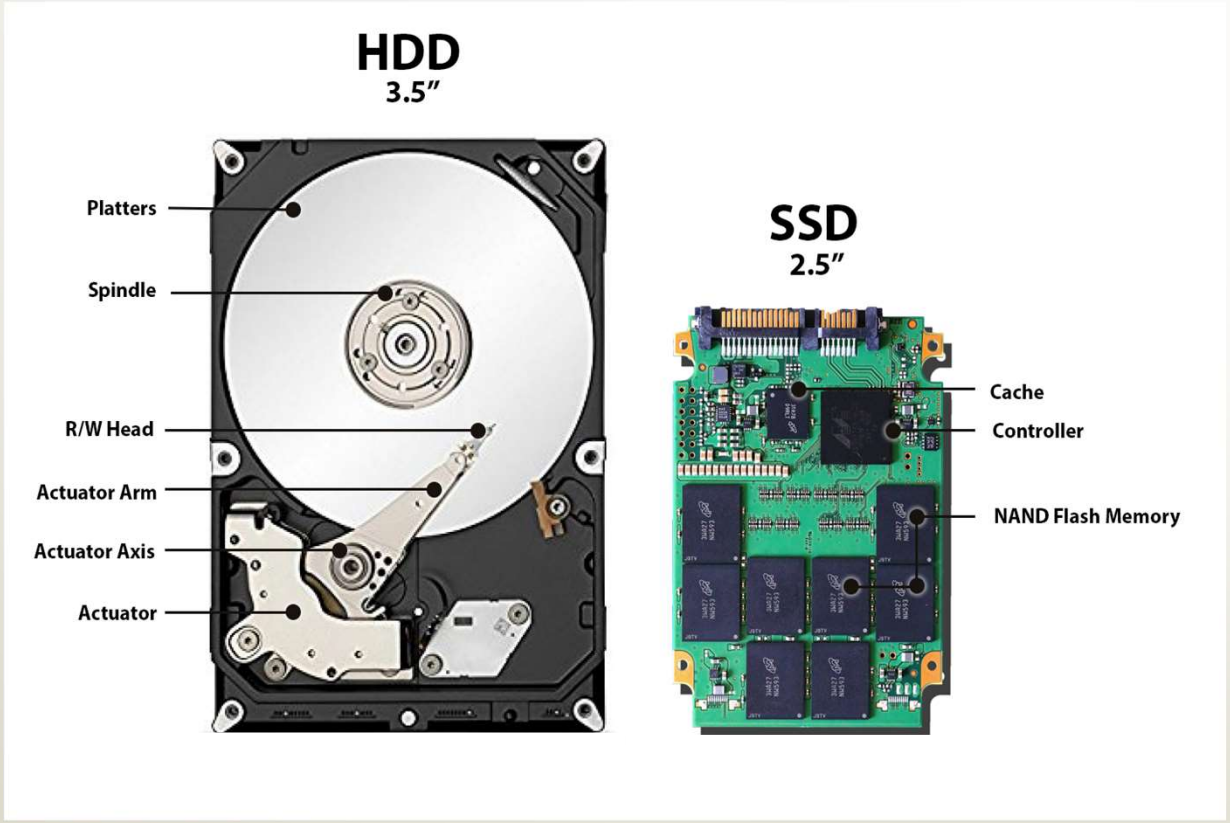
- Storage
- Indexing
  - What is an index? Why do we need it?

Simplified Computer Architecture



Speed (ns)	**10 ns**	**10^2 ns**	**10^7 ns**
Size	**KB**	**GB**	**TB**

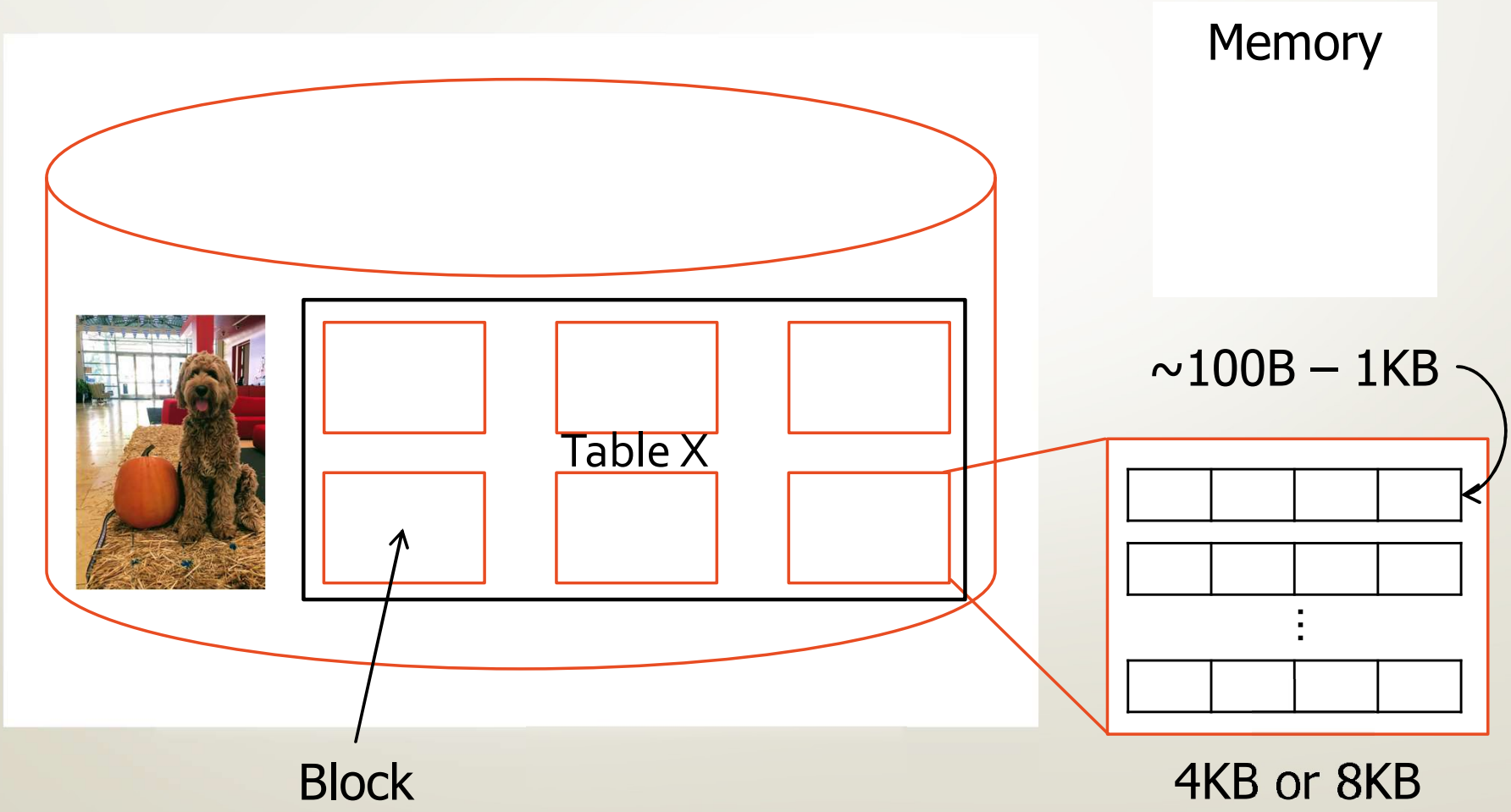
Cost of Accessing Data on Disk



Main Memory

Speed	10^7 ns	10^5 ns	10^2 ns

Block size vs. record size

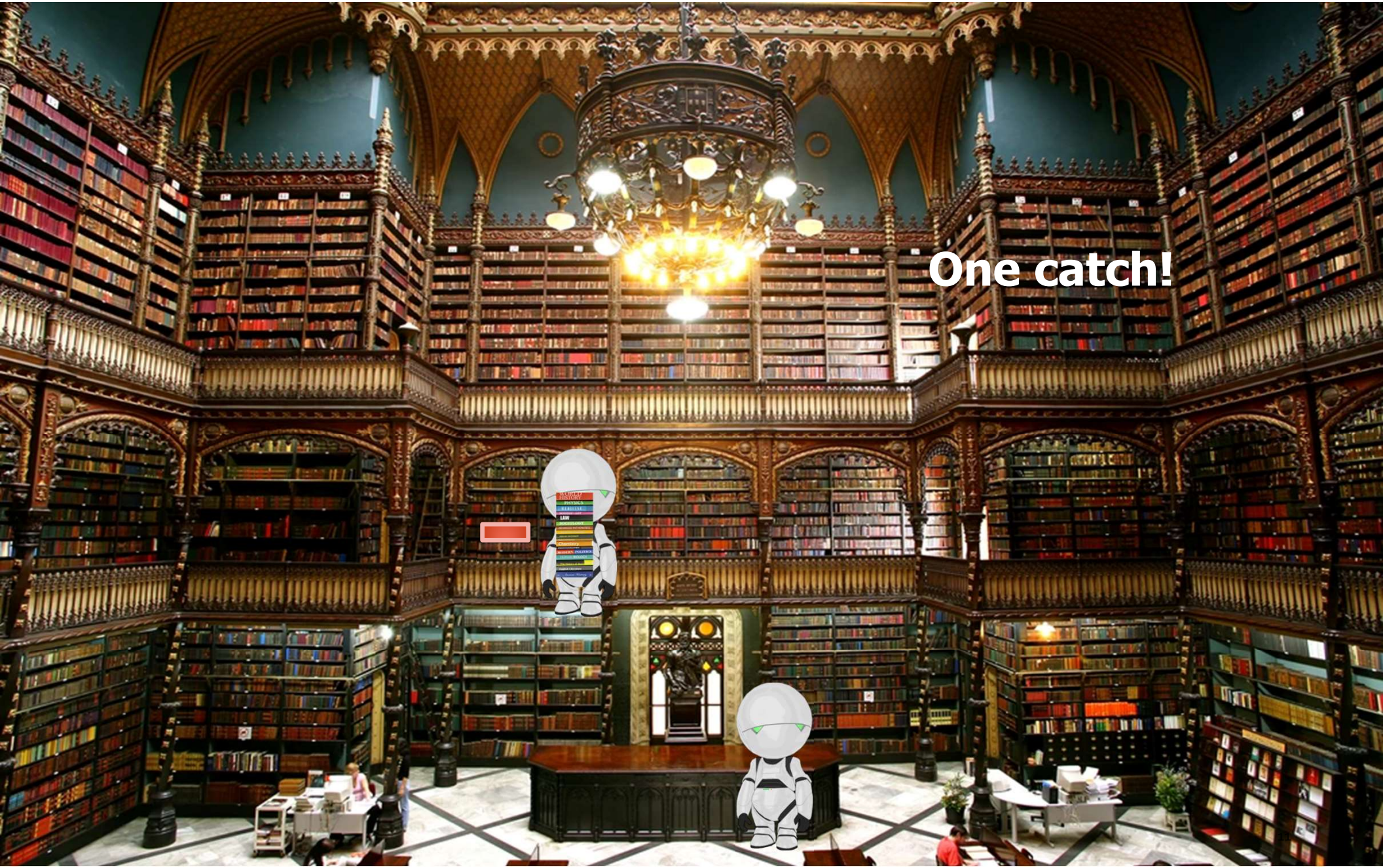


OK. So how do we do simple operations?

Lookups. Insertions. Deletions

Outline

- ✓ Storage
- Indexing
  - What is an index? Why do we need it?


One catch!

Indexes in databases

- An *index* speeds up selections on the *search key field(s)*
- Search key = any subset of the fields of a relation
  - *Search key* is not necessarily the same as a *key*
- Entries in an index: (k, r), where:
  - k = the search key
  - r = the record OR record id OR record ids OR pointers

Some terminology

- ***Data file***: has the data corresponding to a relation
- ***Index file***: has the index
- File consists of smaller units called **blocks** (e.g. of size 4 KB or 8 KB)
- # index blocks < # data blocks. Index may even fit into main memory.

An Index is a Function!

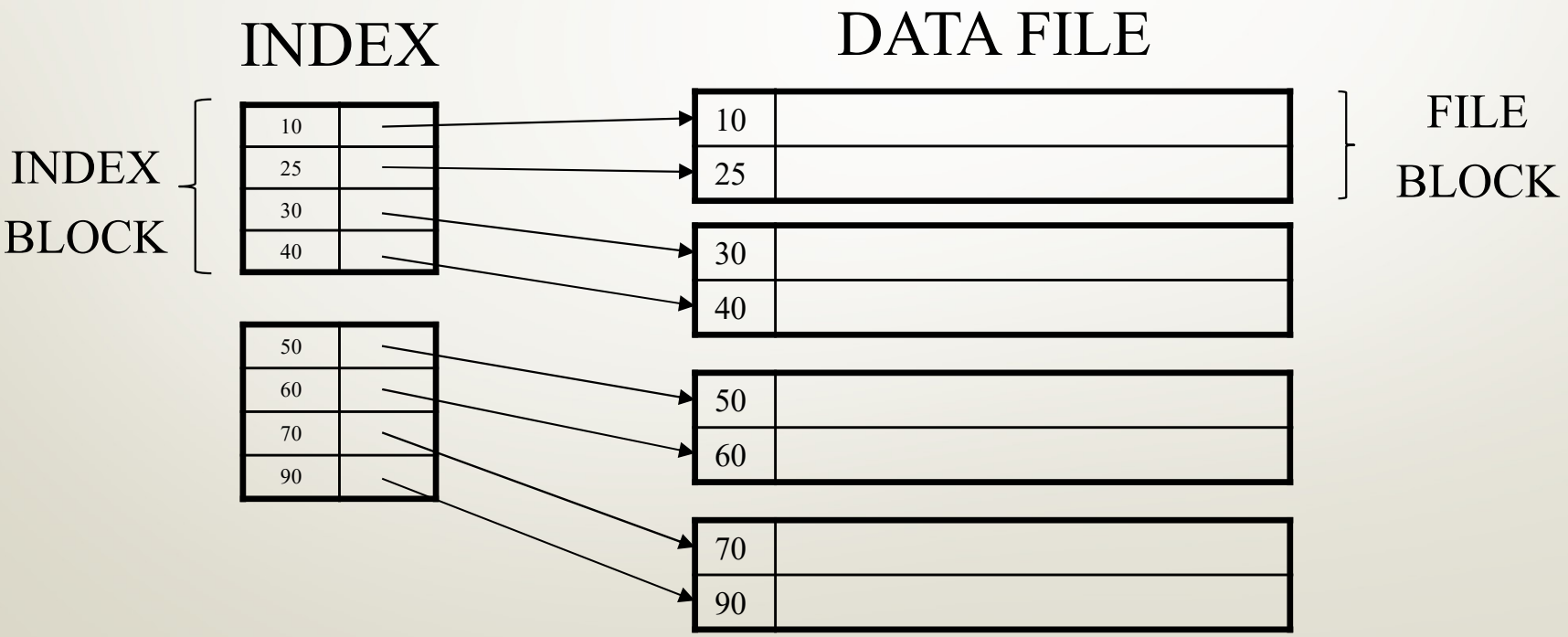
f(what: key) = where: file block

Characteristics of Indexes

- Clustered/unclustered
  - Clustered: records sorted in the search key order
  - Unclustered: records are NOT sorted in the search key order
- Dense/sparse
  - Dense = each record has an entry in the index
  - Sparse = only some records have
- Primary/secondary
  - Primary = on the primary key
  - Secondary = on any attribute

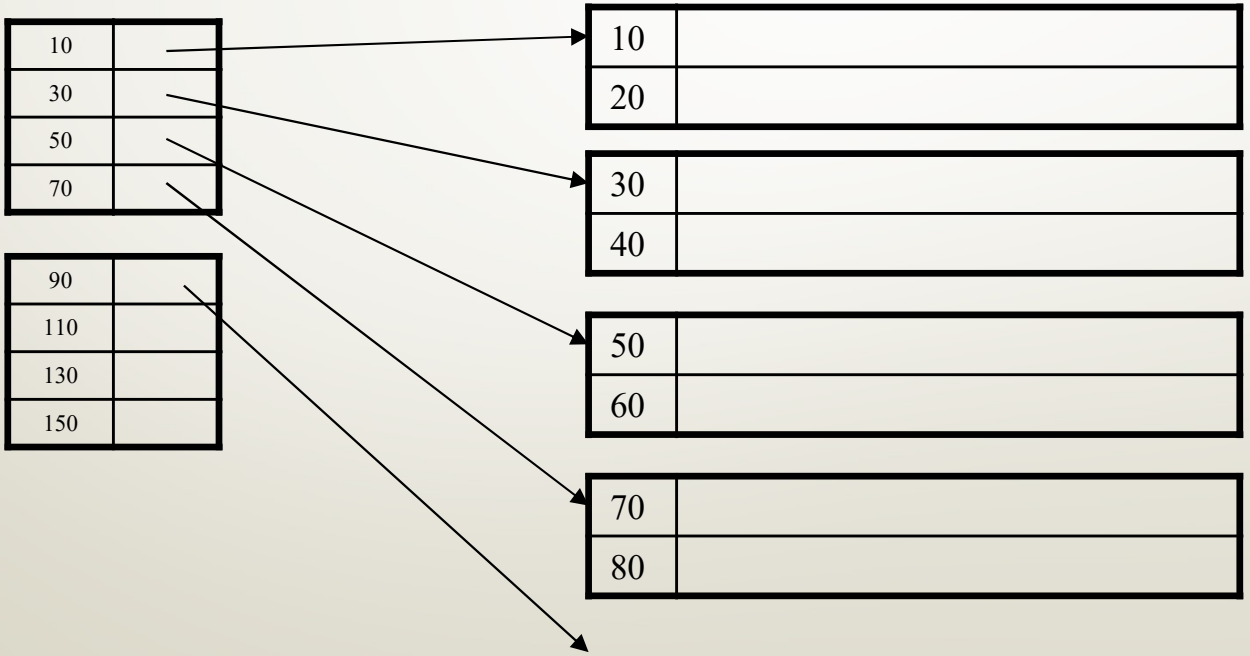
Ex: Clustered, Dense Index

- Clustered: File is sorted on the index attribute
- *Dense*: sequence of (key,pointer) pairs



Clustered, Sparse Index

- *Sparse* index: one key per data block, corresponding to the lowest search key in that block

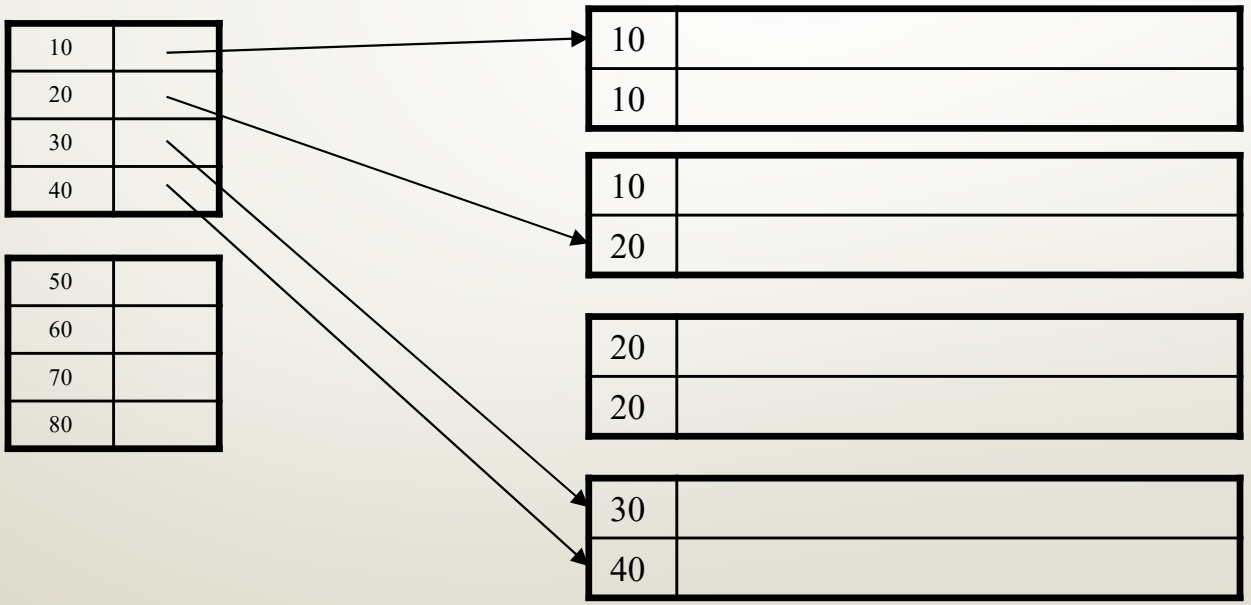


What if there are duplicate keys?

Clustered Index with Duplicate Keys

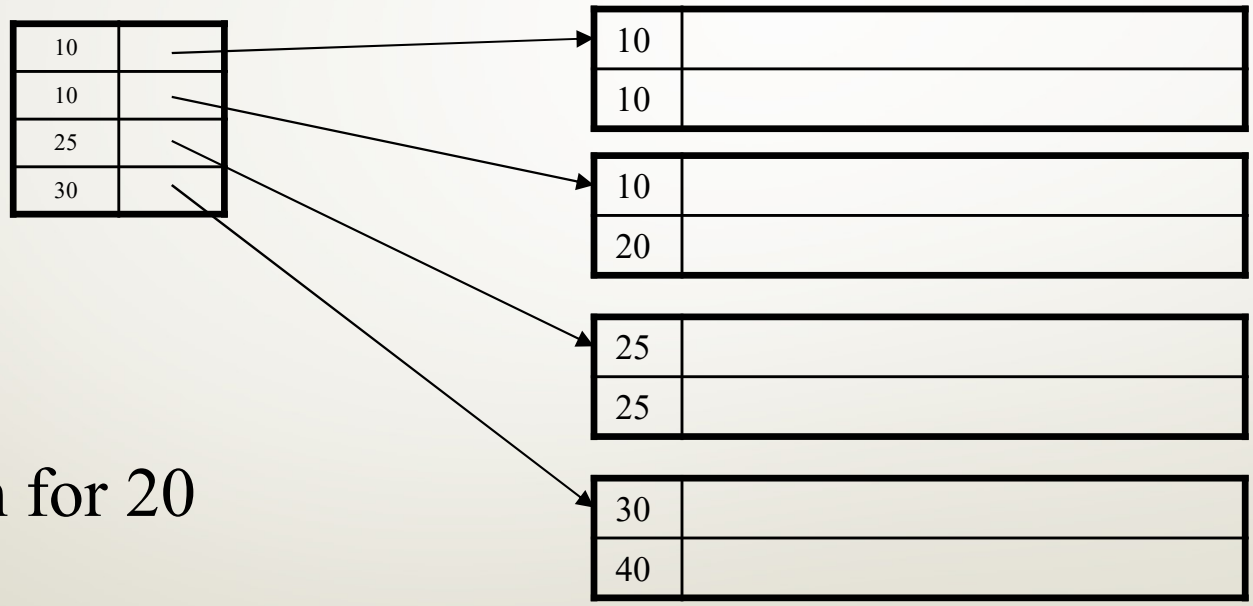
Dense index: point to the first record with that key

(must have a pointer for each new key)



Clustered Index with Duplicate Keys

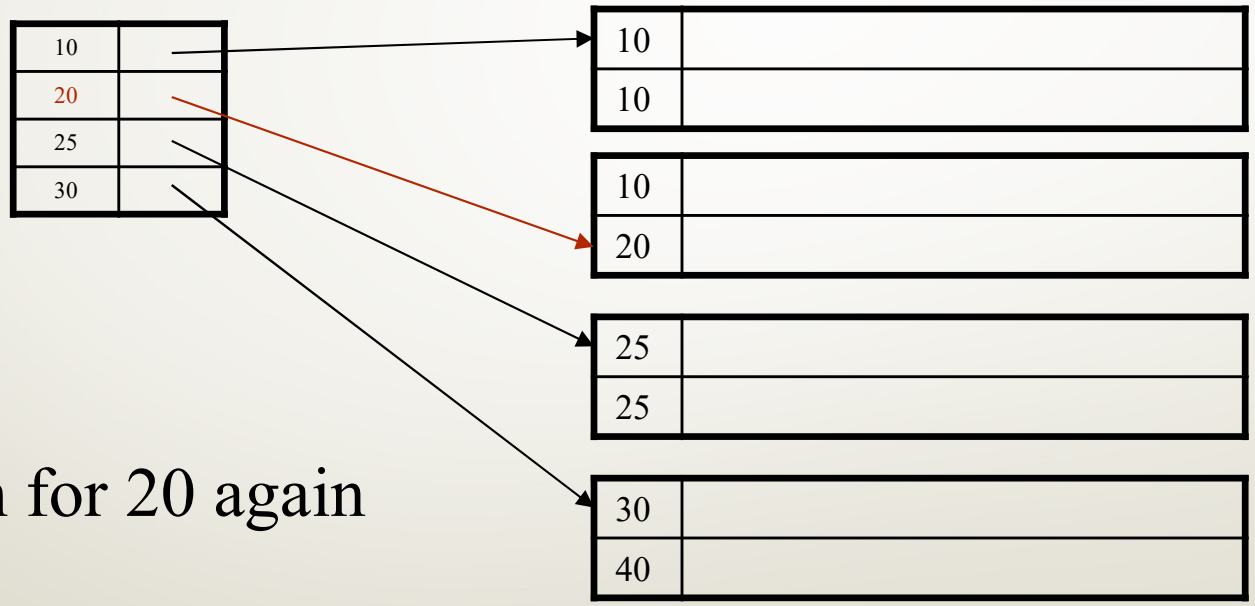
- Sparse index: pointer to lowest search key in each block



Search for 20

Clustered Index with Duplicate Keys

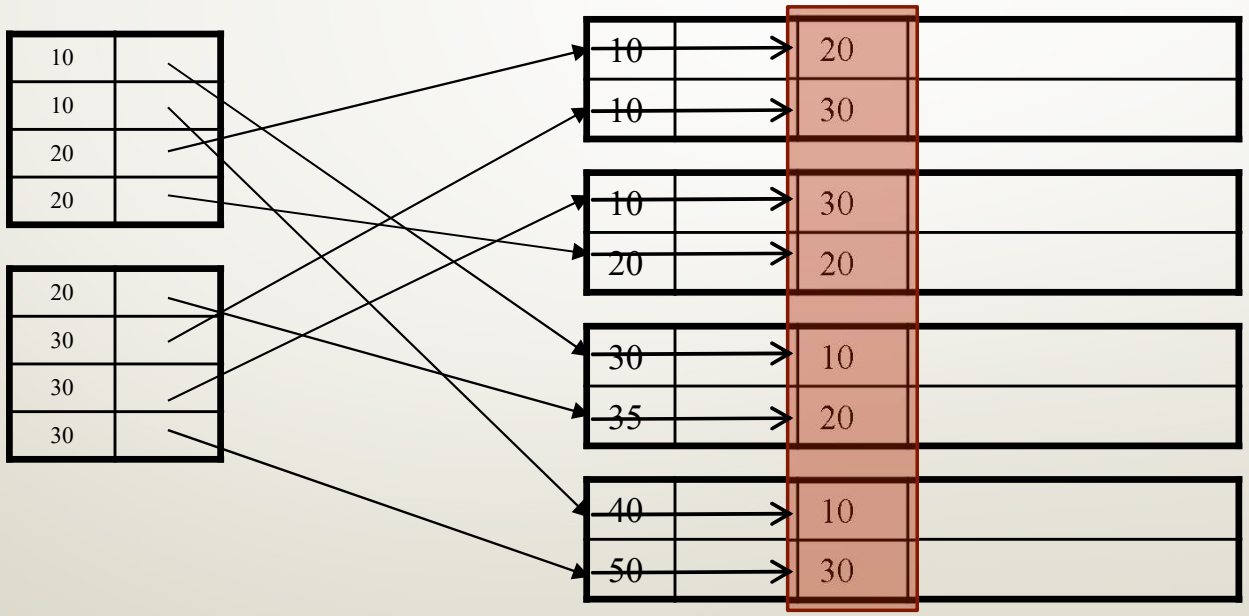
- *Better: pointer to lowest new search key in each block*



Search for 20 again

Unclustered Indexes

- Often for indexing other attributes than primary key
- Can it be sparse?



Summary Clustered vs. Unclustered Index