# COL 362 & COL 632

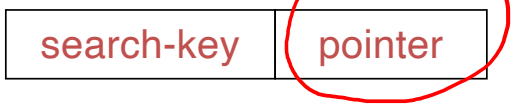
Indexing

22 Feb 2023

# Basic Concepts

file ≡ relation

- Indexing mechanisms speed up access to desired data.
- Search Key – set of attributes used to look up records in a file.
- An index file consists of records (called index entries) of the form




- Index files are typically much smaller than the original file
- Two basic kinds of indexes
  - **Ordered indexes:** search keys are stored in sorted order
  - **Hash indexes:** search keys are distributed uniformly across buckets using a hash function

Secondary memory Indexes

# Dimensions for Evaluating Indexes

- **Access type**
  - Tuple with a specified value in the attribute
  - Tuples with an attribute value falling in a specified range of values.
- **Access time**
  - Find one or more items – or to confirm searched item does not exist
- **Insertion time**
- **Deletion time**
- **Space overhead**

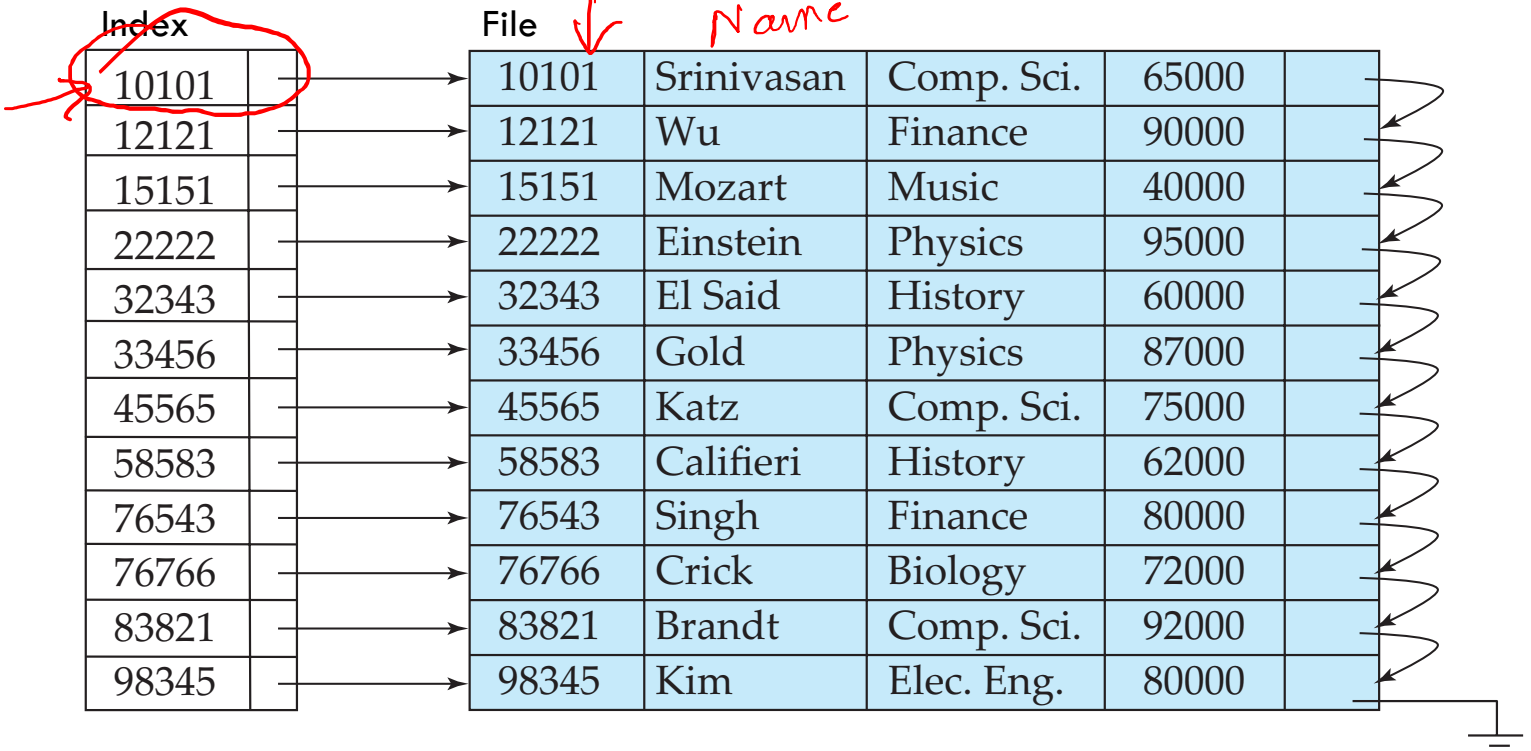
More than one index for a relation

# Ordered Indexes

- Index entries are sorted on the **search key** value
  - Search key can be **one or more** attributes
- Primary or clustering index
  - Search key is the same as the corresponding sequentially ordered file
  - The search key of a primary index is usually *but not necessarily* the primary key
- Secondary or non-clustering index
  - Search key different from corresponding sequentially ordered file

# Dense Indexes (1/2)

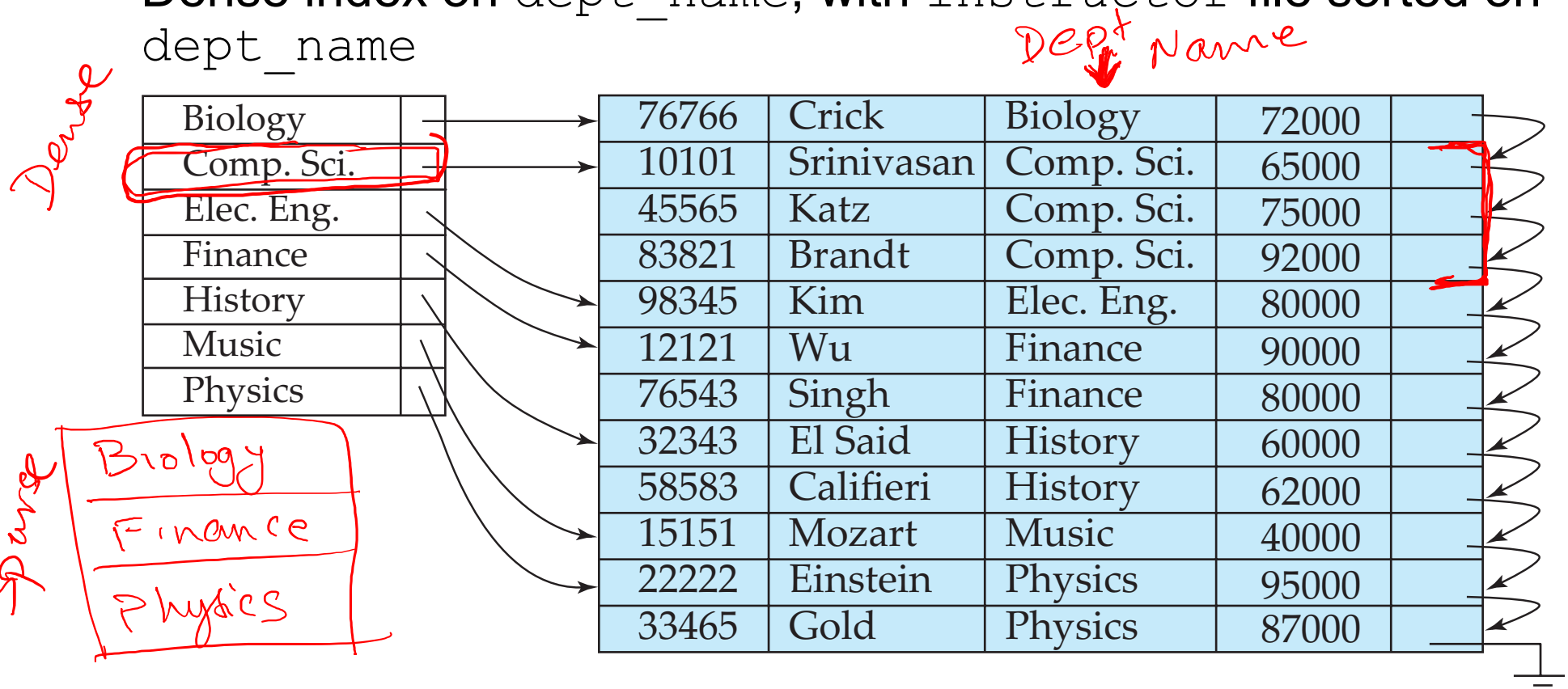
- Index record appears for every search-key value in the file.
- index on `ID` attribute of `instructor` relation



# Dense Indexes (2/2)

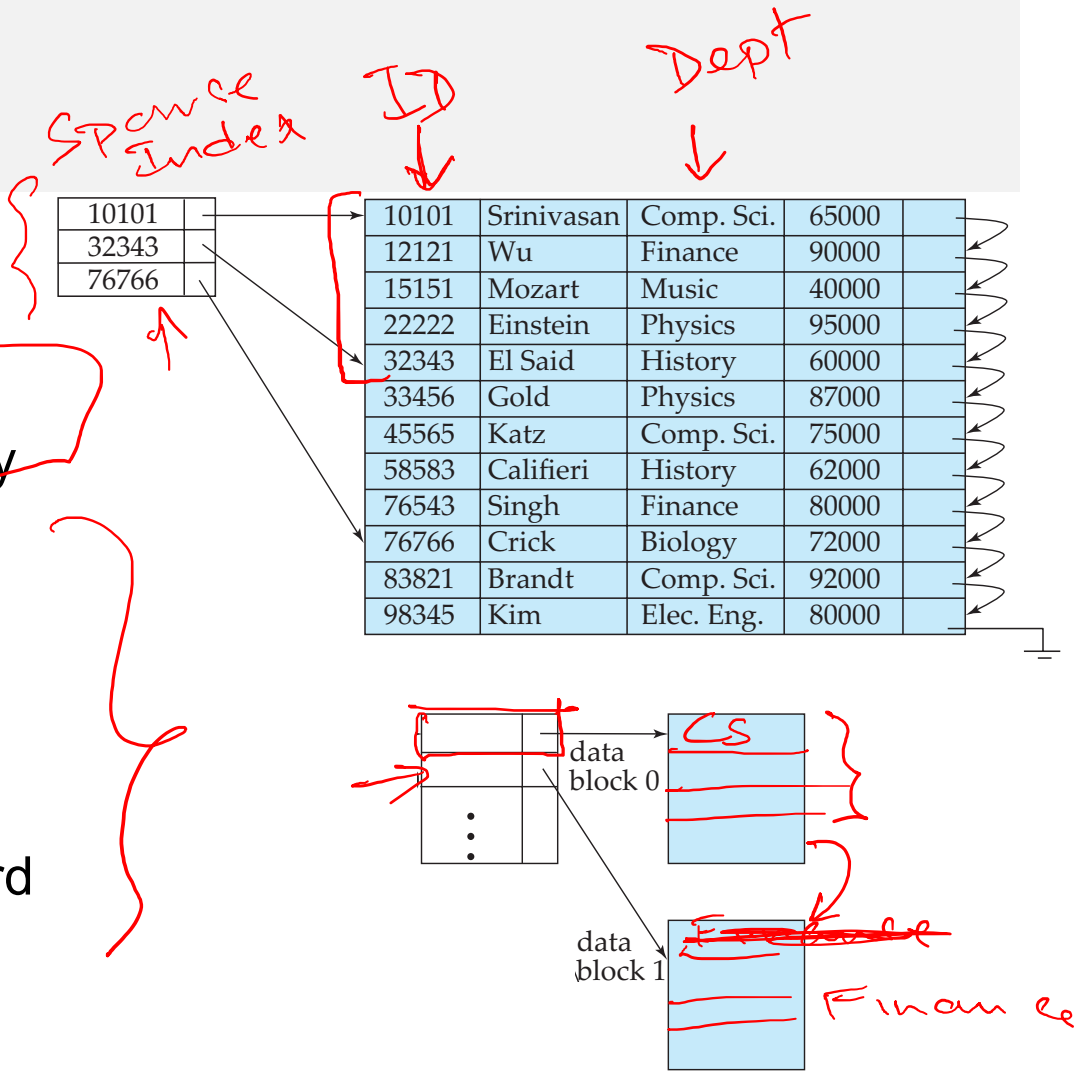
Dense clustered Index
Dense non-clustered Index.

- Dense index on `dept_name`, with `instructor` file sorted on `dept_name`



# Sparse Indexes

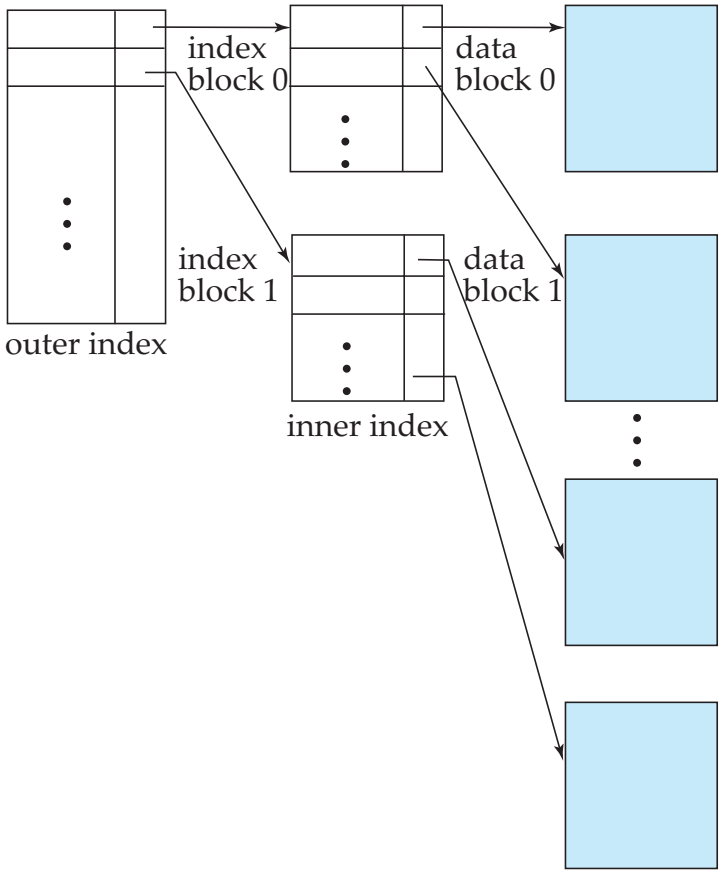
- Contains index records for only some search-key values
  - Applicable when records are sequentially ordered on search-key
- To locate a record with search-key value *K* we:
  - Find index record with largest search-key value < *K*
  - Search file sequentially starting at the record to which the index record points



# Multilevel Indexes

- If primary index does not fit in memory, access becomes expensive.
- Construct a sparse index on primary index – 2-level index
- Extend the idea to multiple levels

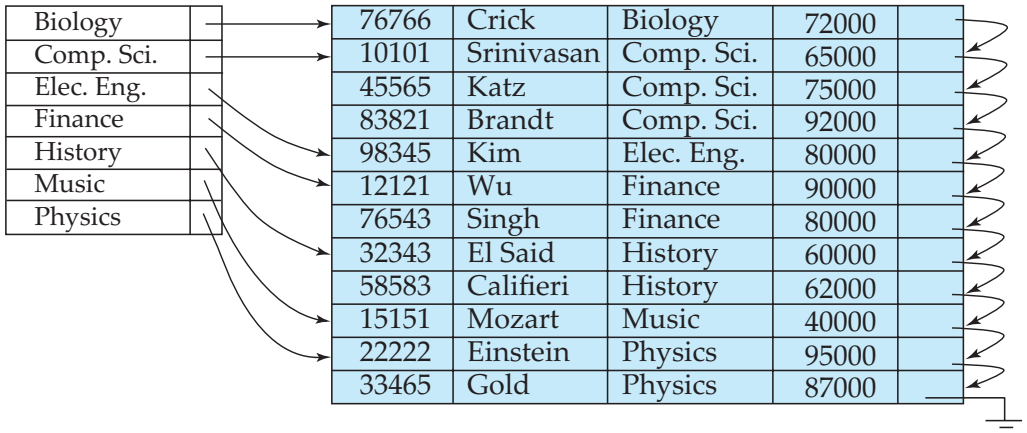
B+ - tree

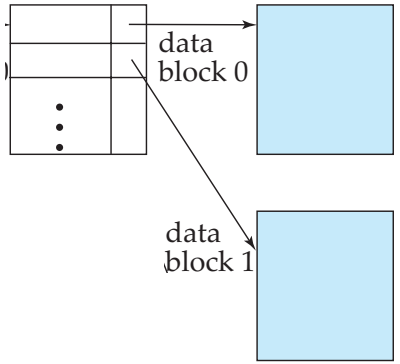


# Insertion

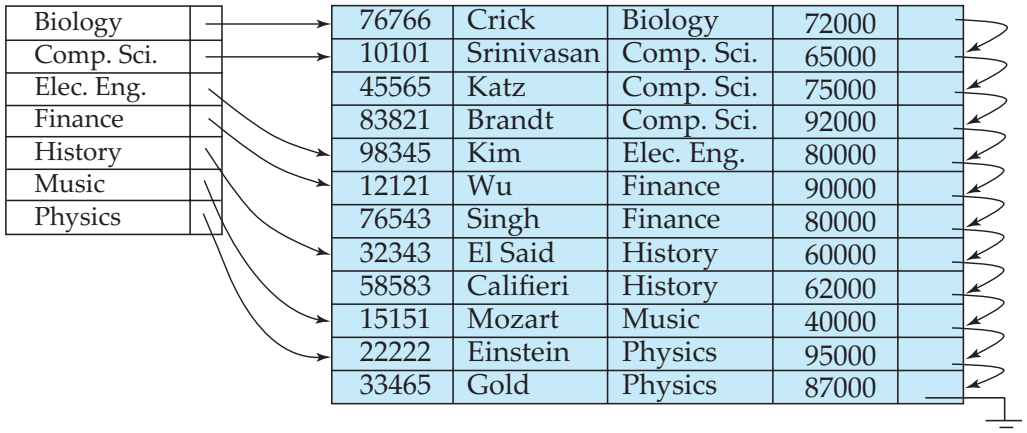
Dense

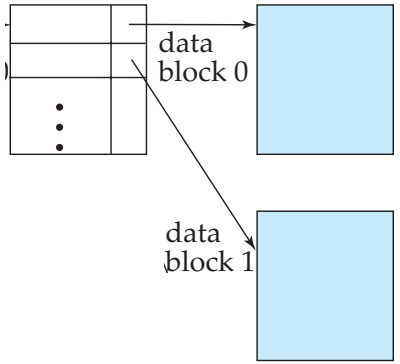
Sparse



# Deletion


Biology
Comp. Sci.
Elec. Eng.
Finance
History
Music
Physics
76766 Crick Biology 72000
10101 Srinivasan Comp. Sci. 65000
45565 Katz Comp. Sci. 75000
83821 Brandt Comp. Sci. 92000
98345 Kim Elec. Eng. 80000
12121 Wu Finance 90000
76543 Singh Finance 80000
32343 El Said History 60000
58583 Califieri History 62000
15151 Mozart Music 40000
22222 Einstein Physics 95000
33465 Gold Physics 87000


data
block 0
data
block 1

# Secondary Indexes

- Index record points to a bucket that contains pointers to all the actual records with that particular search-key value.
- Secondary indexes have to be dense