
DATABASE

CSE3103 : Database
FALL 2020

Nazmus Sakib
Assistant Professor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Basic Concepts

- Indexing mechanisms used to speed up access to desired data.
  - E.g., author catalog in library
- **Search Key** - attribute to 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 indices:
  - **Ordered indices:** search keys are stored in sorted order
  - **Hash indices:** search keys are distributed uniformly across "buckets" using a "hash function".

Index Evaluation Metrics

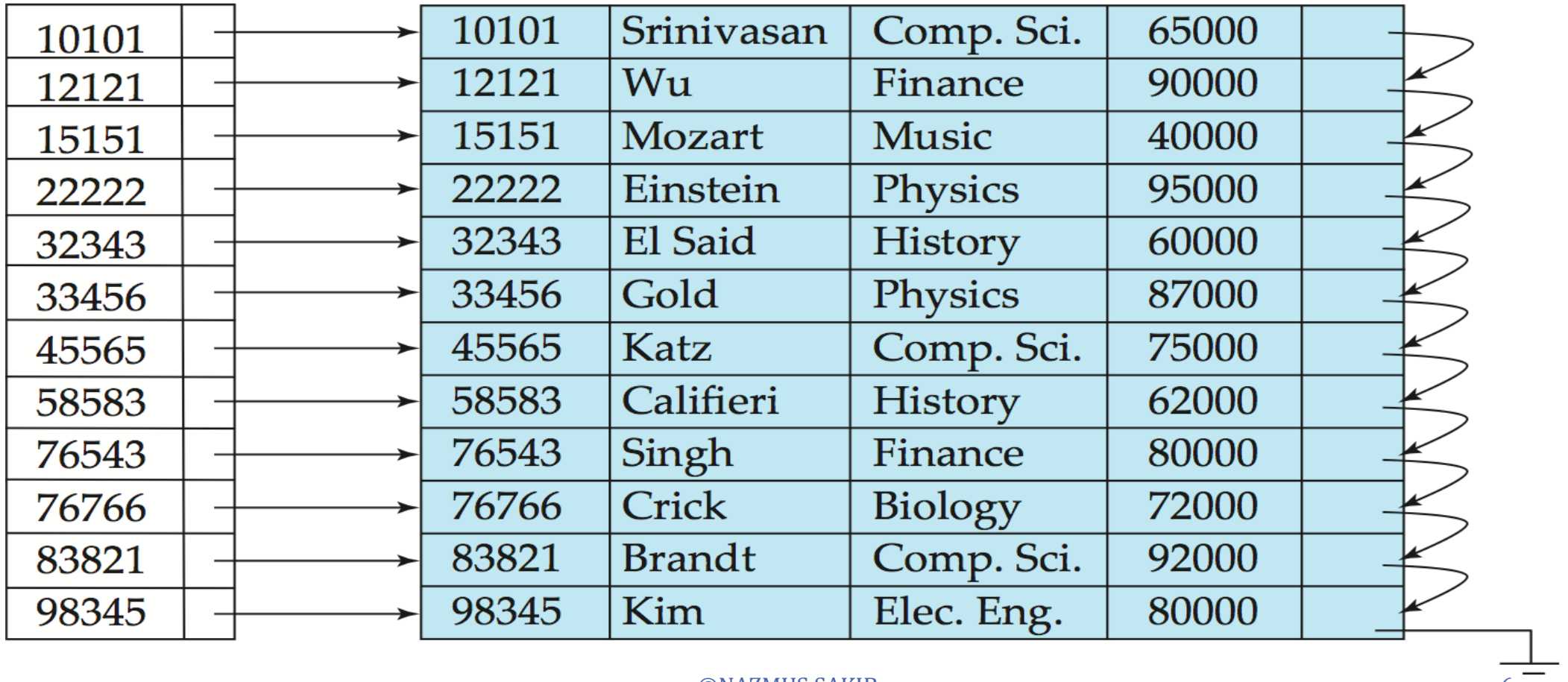
- Access types supported efficiently. E.g.,
  - records with a specified value in the attribute
  - or records with an attribute value falling in a specified range of values.
- Access time
- Insertion time
- Deletion time
- Space overhead

Ordered Indices

- In an **ordered index,** index entries are stored sorted on the search key value. E.g., author catalog in library.
- **Primary index:** in a sequentially ordered file, the index whose search key specifies the sequential order of the file.
  - Also called **clustering index**
  - The search key of a primary index is usually but not necessarily the primary key.
- **Secondary index**: an index whose search key specifies an order different from the sequential order of the file. Also called non-clustering index.
- Index-sequential file: ordered sequential file with a primary index.

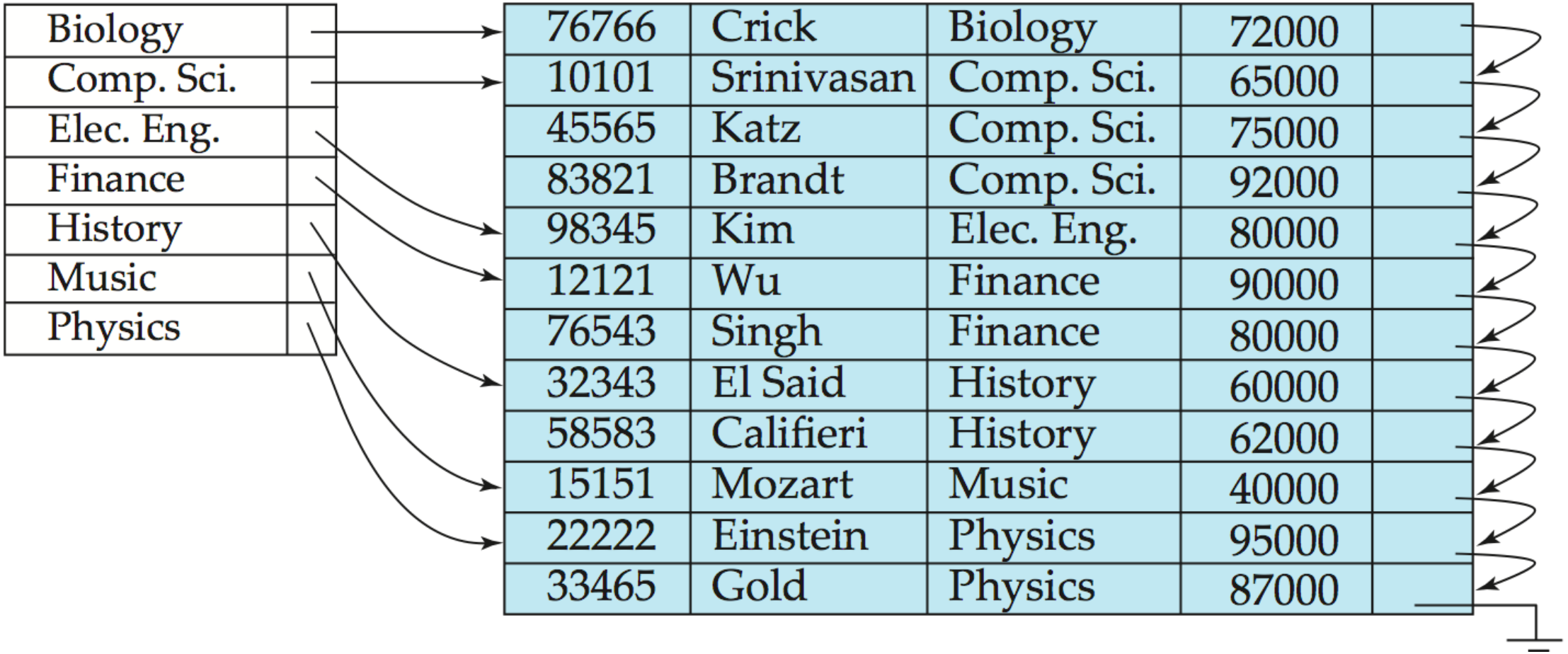
Dense Index Files

- **Dense index** — Index record appears for every search-key value in the file.
- E.g. index on *ID* attribute of *instructor* relation



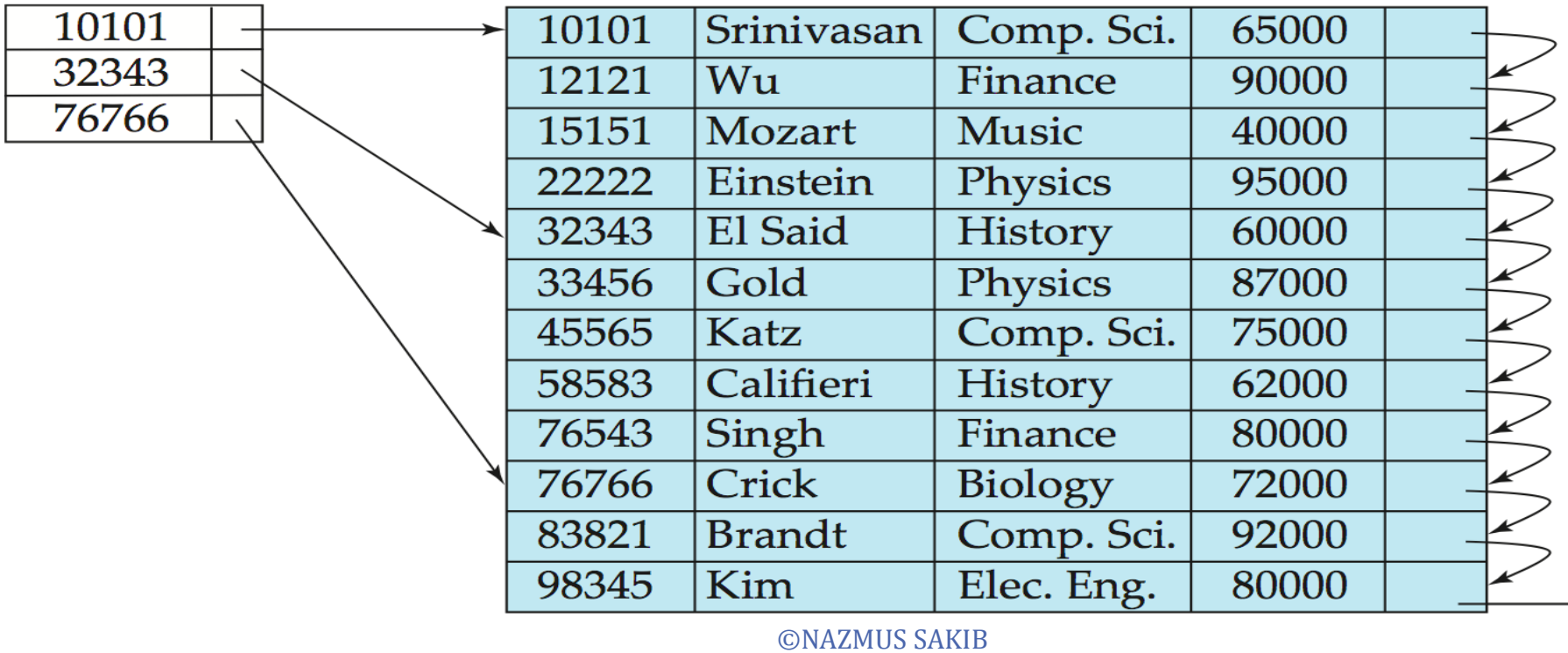
Dense Index Files (Cont.)

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



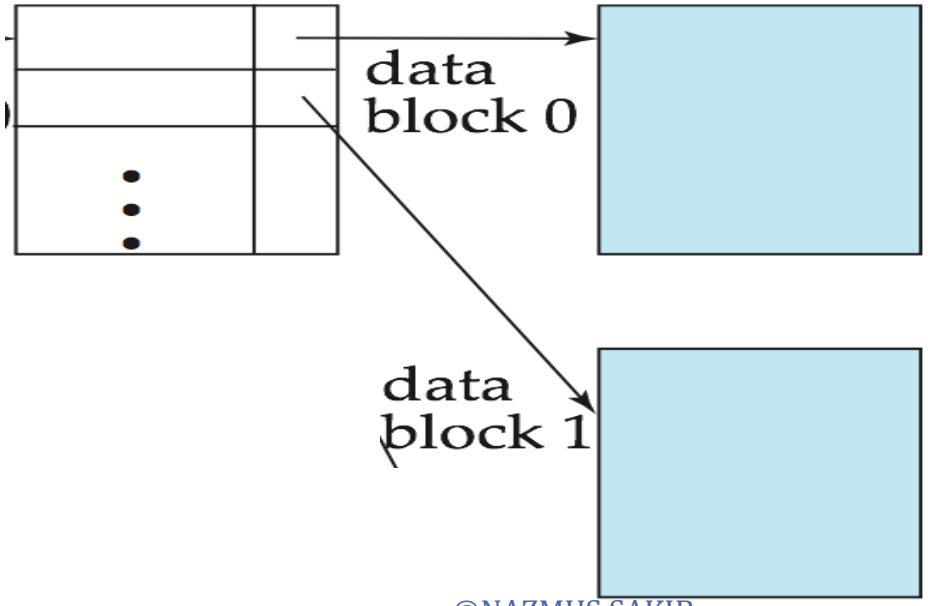
Sparse Index Files

- **Sparse Index**: 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



Sparse Index Files (Cont.)

- Compared to dense indices:
  - Less space and less maintenance overhead for insertions and deletions.
  - Generally slower than dense index for locating records.
- **Good tradeoff**: sparse index with an index entry for every block in file, corresponding to least search-key value in the block.




DBMS
data
database
distributed
parallel
shared
system
Distributed
DDBMS
DBMSs
memory
architecture
systems
network
processing
performance
processors
access
centralized
multiple
sites
Parallel
processor
may
typically
fragments
major
require
physically
users
site
applications
computers
Database
nothing
Shared
application
management
transparent
bottleneck
interconnection
Figure
organizational
administered
mode
technology
computer
disk
stored
logically
number
user
dis
local
large
decentralized
directly
provide
replicated
sometimes
computing
single
one
split
disks
islands
based
make
also
many
often
distinction
logical
control
scalability
approach
improve
definition
development
nodes
databases
capable
least
used
partitioned
consists
global
associated
incompatible
structure
collection
provides
way
Management
makes
per
within
machines
like
reliability
across
eliminates
fact
communication
distribution
referred
operational
architectures
scalable
order
resources
remote
defined
controlled
Date
com
must
System
share
discuss