Chapter 11: Indexing

Edited by Radhika Sukapuram

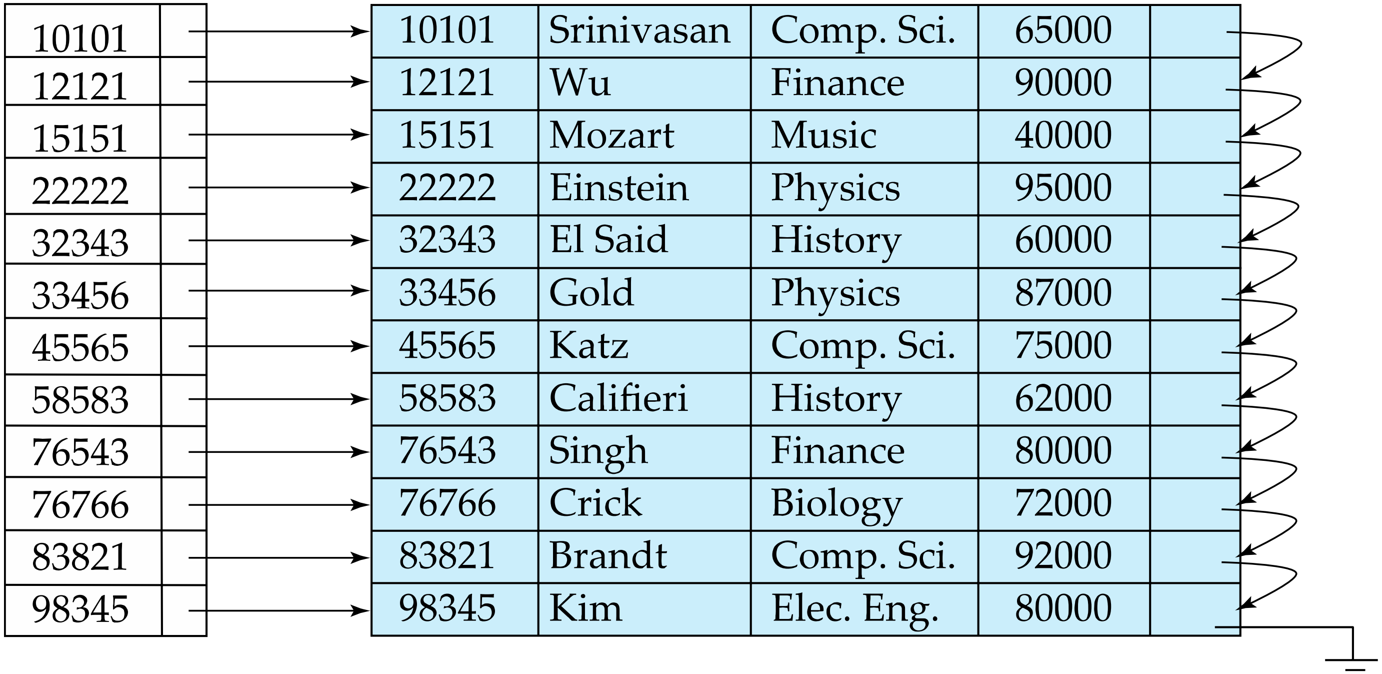
Database System Concepts, 7th Ed.

Ordered Indices

- **Ordered index**: index entries are stored based on a sorted ordering of the search key values.
  - E.g., author catalog in library.
  - **Clustering index:** in a sequentially ordered file, the index whose search key specifies the sequential order of the file.
    - Also called **primary index**
    - The search key of a clustering index is usually (but not necessarily) the primary key.
  - **Non-clustering index**: an index whose search key specifies an order different from the sequential order of the file. Also called **secondary index.**
- Two types of ordered indices: **dense** and **sparse**.

Dense Index Files

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

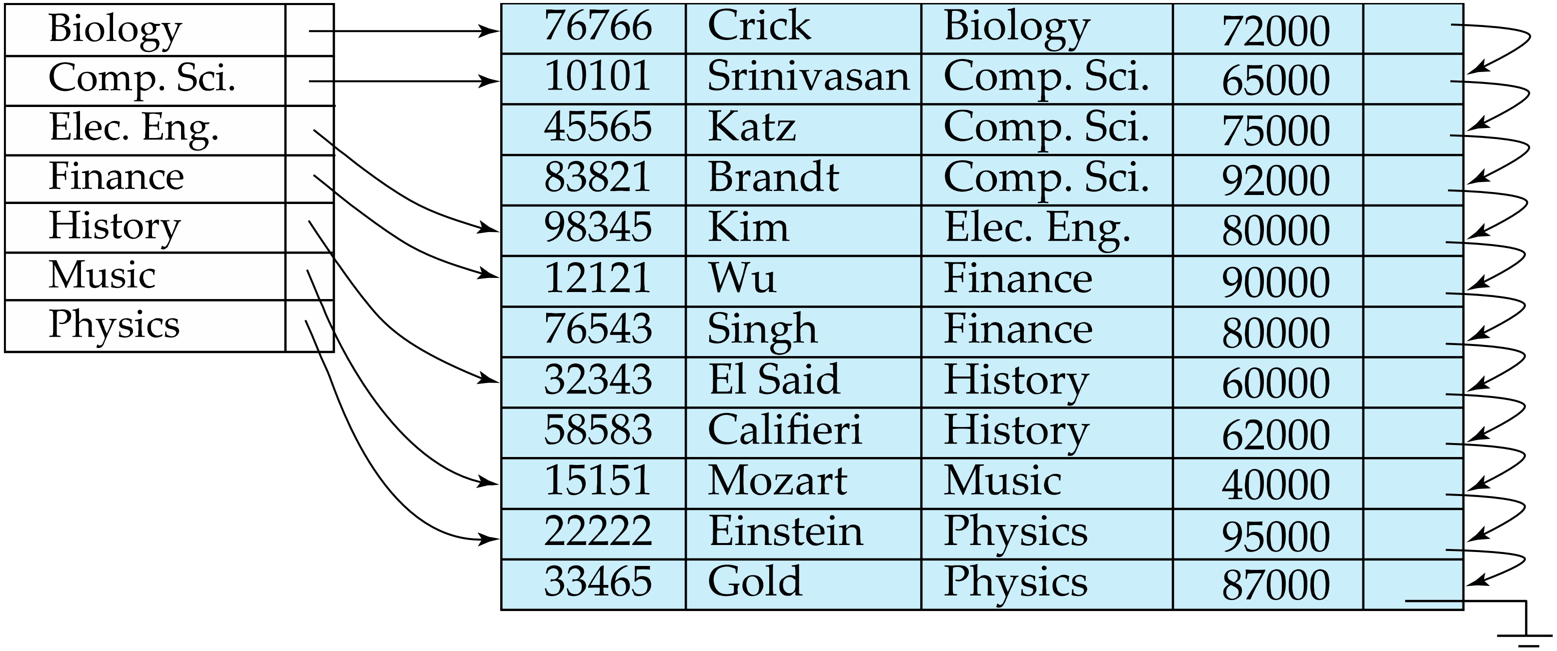


10101	
12121	
15151	
22222	
32343	
33456	
45565	
58583	
76543	
76766	
83821	
98345	

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

Dense Index Files (Cont.)

- Dense (clustering) index on *dept_name*, with *instructor* file sorted on *dept_name*



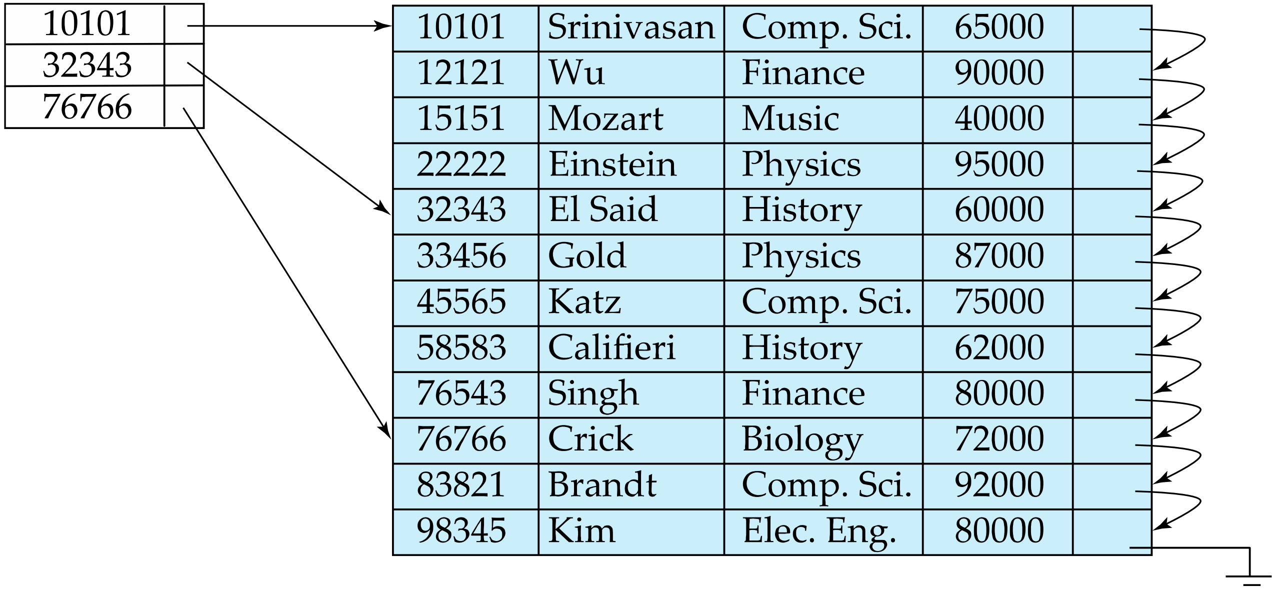
Biology
Comp. Sci.
Elec. Eng.
Finance
History
Music
Physics

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

Most databases create an index on the primary key. Why ? What needs to be checked when a tuple is inserted ?

Sparse Index Files

- **Sparse Index**: contains index records for only some search-key values.
  - Applicable only 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

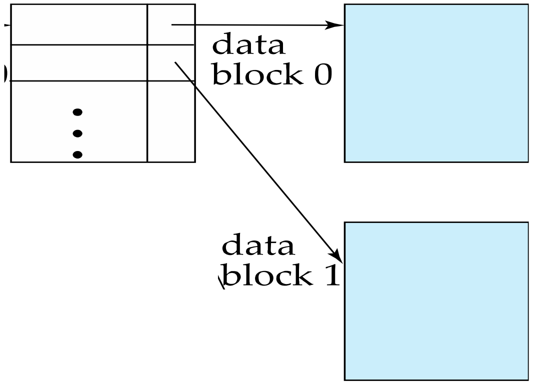


10101	
32343	
76766	

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

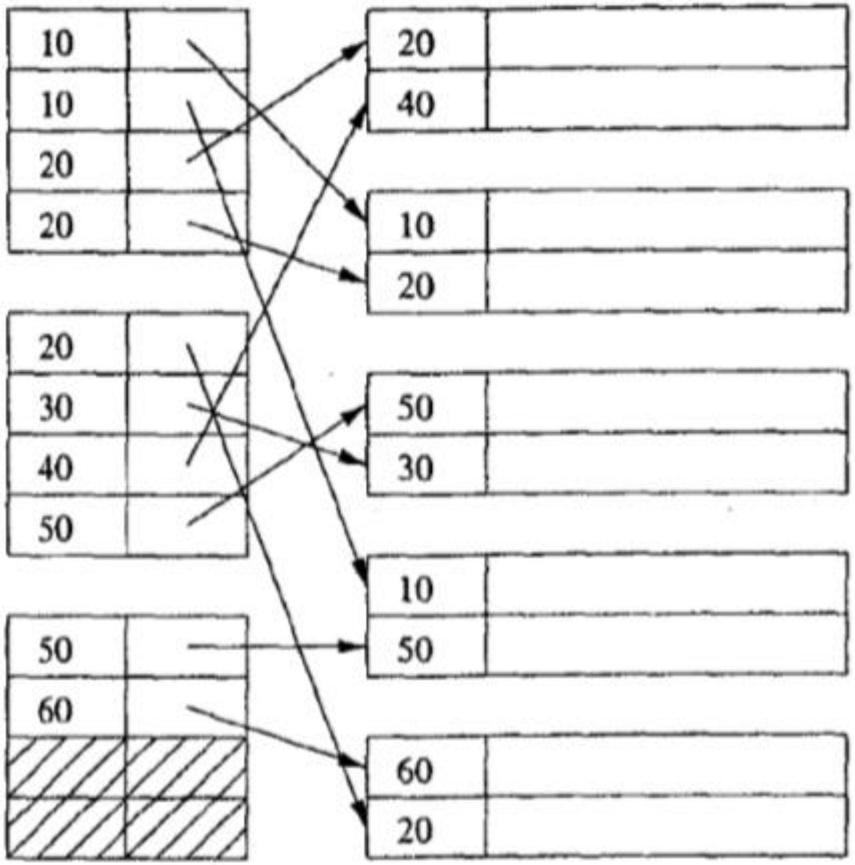
Sparse Index Files (Cont.)

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



  - For un-clustered index: sparse index on top of dense index (multilevel index)

Non-clustering/secondary index

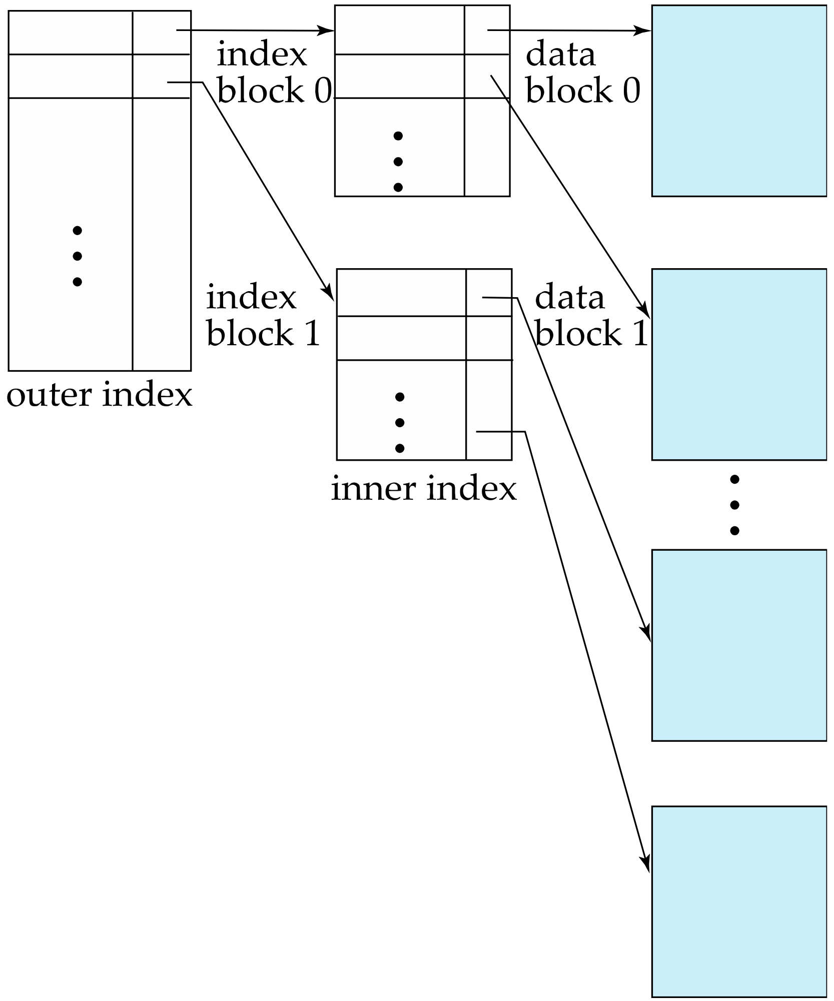


- A dense index, usually with duplicates
- Improves performance of queries that use keys other than the key of the clustering index
- A sparse secondary index is not possible. Why ?

Multilevel Index

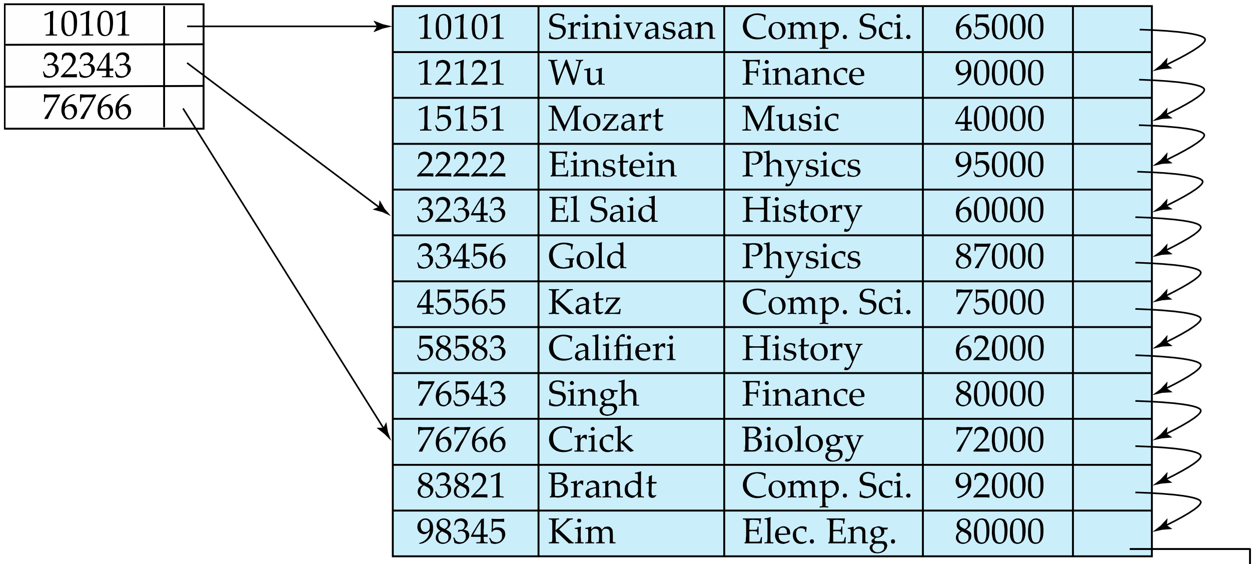
- If a primary index does not fit in memory, access becomes expensive.
- Solution: treat primary index kept on disk as a sequential file and construct a sparse index on it.
  - Outer index – a sparse index of primary index
  - Inner index – the primary index file
- If even outer index is too large to fit in main memory, yet another level of index can be created, and so on.
- Indices at all levels must be updated on insertion or deletion from the file.

Multilevel Index Example



Deletion of Records

- **Dense indices** – If deleted record was the only record in the file with its particular search-key value,
  - the search-key is deleted from the index also
- **Sparse indices** –
  - If an entry for the search key exists in the index,
    - it is deleted by replacing the entry in the index with the next search-key value in the file (in search-key order).
    - If the next search-key value already has an index entry, the entry is deleted instead of being replaced.



10101	
32343	
76766	

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	