Part 10:
File Organization and and Indexing

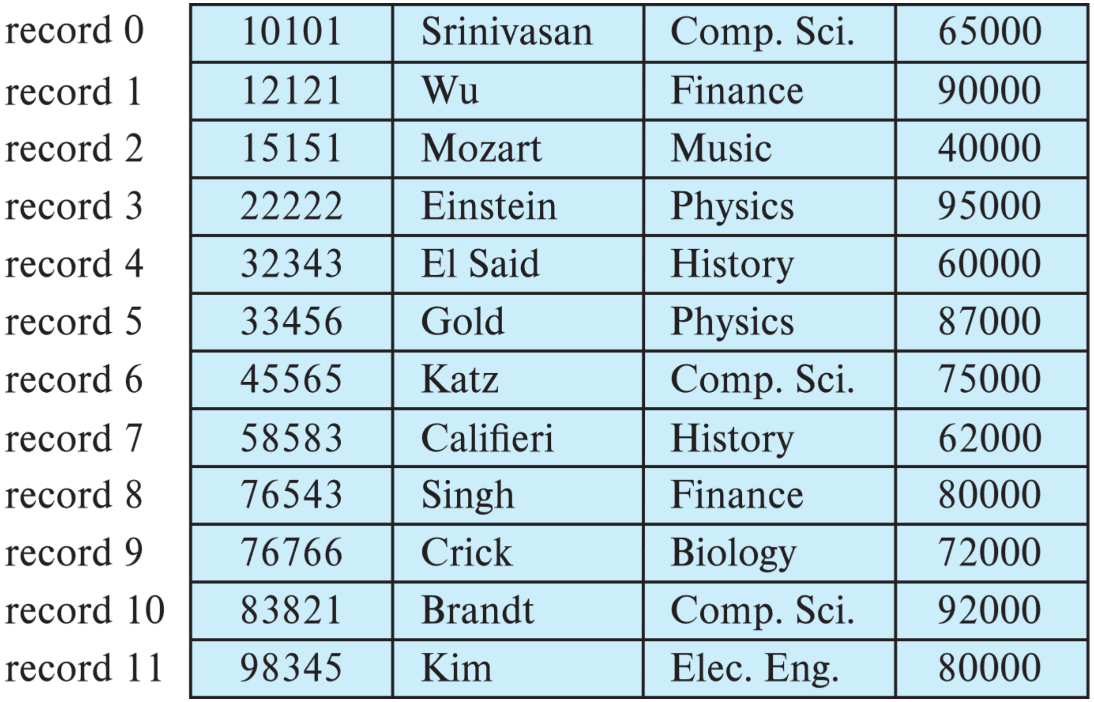
Database System Concepts, 7th Ed.

File Organization

- The database is stored as a collection of operating systems **files**
  - Each file is a sequence of **records**
  - A record is a sequence of **fields**
- From the O/S point of view, each file consists of fixed-length **blocks** (also called **physical records**)
  - Blocks are the units of both storage allocation and data transfer
- One approach
  - Assume record size is fixed
  - Each file has records of one particular type only
  - Different files are used for different relations

Fixed-Length Records

- Simple approach:
  - Store record *i* starting from byte *n* ∗ (*i* – 1), where *n* is the size of each record.
  - Record access is simple but records may cross blocks
    - Modification: do not allow records to cross block boundaries

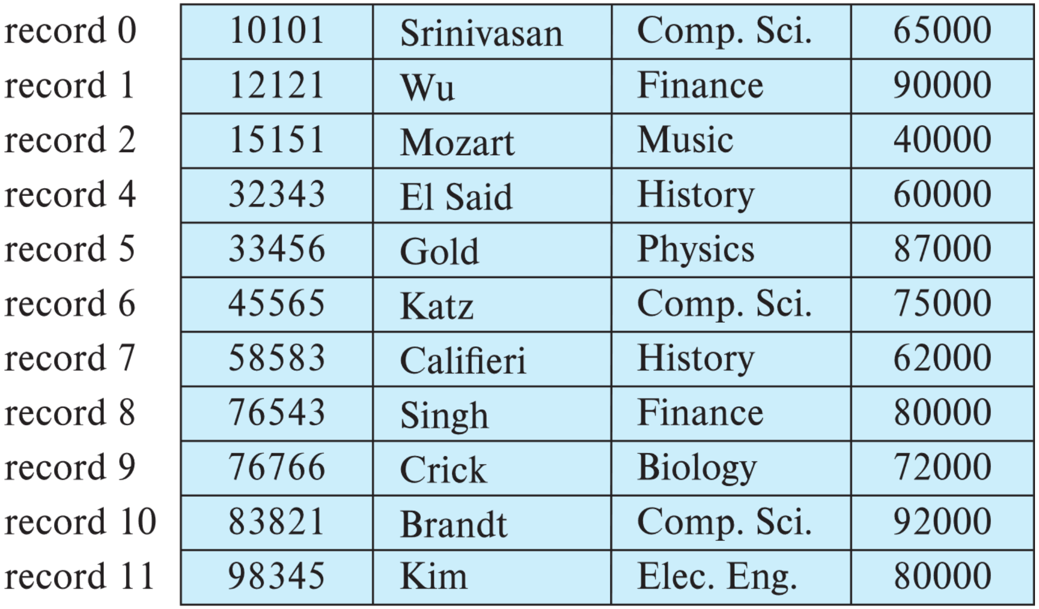


record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

Fixed-Length Records

- Deletion of record *i:* different alternatives*:*
  - ***move records i + 1, . . ., n to i, . . . , n – 1***
  - move record *n* to *i*
  - do not move records, but link all free records on a *free list*

Record 3 deleted

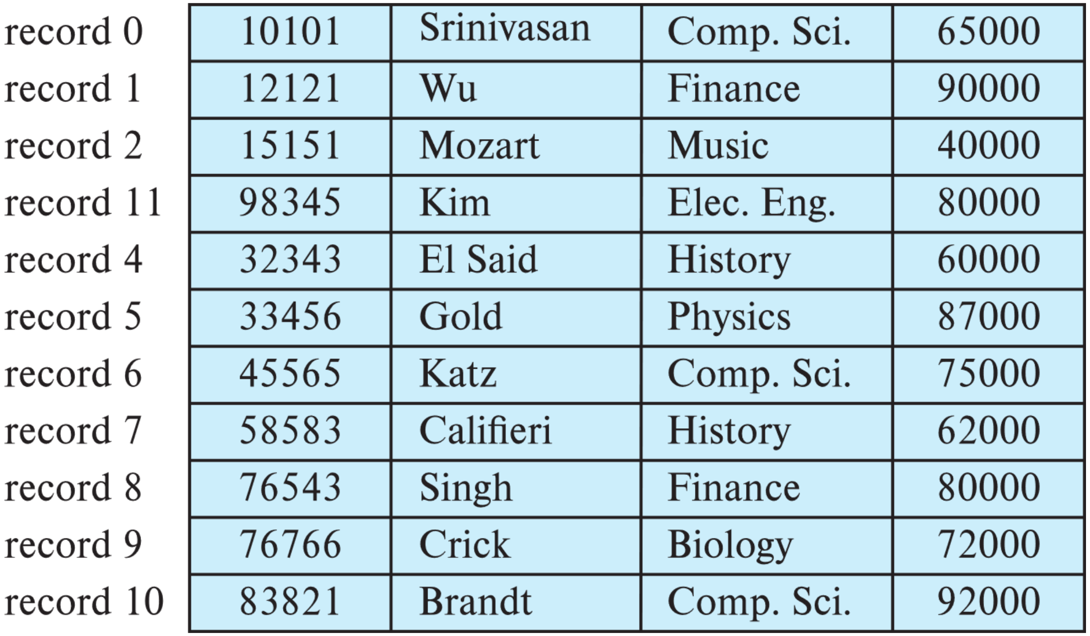


record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

Fixed-Length Records

- Deletion of record *i:* different alternatives*:*
  - move records *i* + 1, . . ., *n* to *i, . . . , n* – 1
  - **move record *n* to *i***
  - do not move records, but link all free records on a *free list*

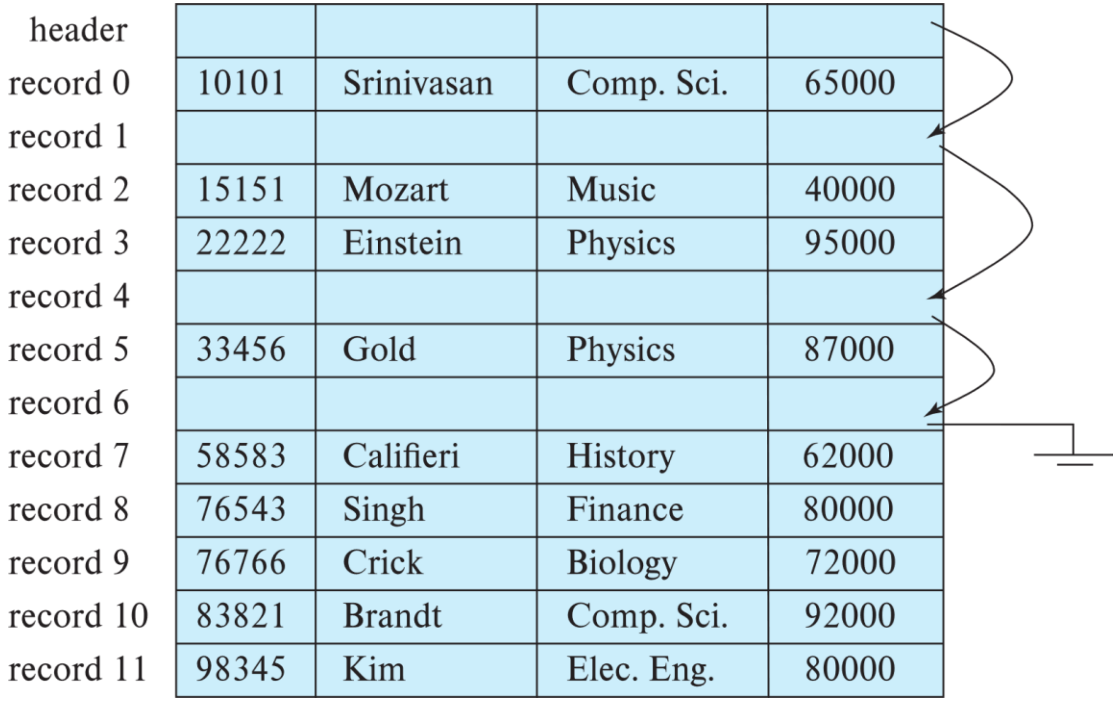
Record 3 deleted and replaced by record 11



record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 11	98345	Kim	Elec. Eng.	80000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000

Fixed-Length Records

- Deletion of record *i:* different alternatives*:*
  - move records *i* + 1, . . ., *n* to *i, . . . , n* – 1
  - move record *n* to *i*
  - **do not move records, but link all free records on a *free list***



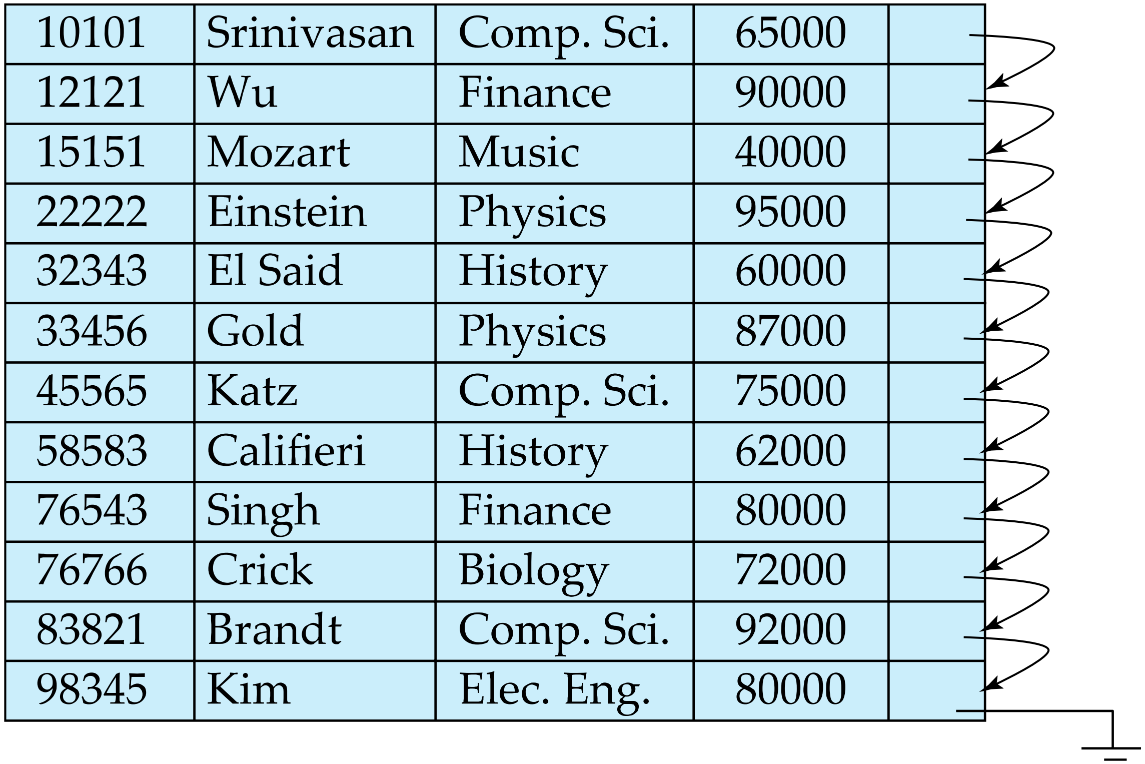
header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1				
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4				
record 5	33456	Gold	Physics	87000
record 6				
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

Organization of Records in Files

- **Heap** – record can be placed anywhere in the file where there is space
- **Sequential** – store records in sequential order, based on the value of the search key of each record
- In a **multitable clustering file organization** records of several different relations can be stored in the same file
  - Motivation: store related records on the same block to minimize I/O
- **B^+-tree file organization**
  - Ordered storage even with inserts/deletes
- **Hashing** – a hash function computed on search key; the result specifies in which block of the file the record should be placed

Sequential File Organization

- Suitable for applications that require sequential processing of the entire file
- The records in the file are ordered by a search-key



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	

Sequential and Binary Search

- For a sequential search for *N* items, the mean number of comparisons E(*X*) is ≈ *N*/2
  - Assuming each item has the same probability of being the required one (i.e., 1/*N*), then the first item has probability 1/*N* of being the required item, the second item has probability 1/*N* of being the required item, and so on, and the search will terminate upon locating the required item:

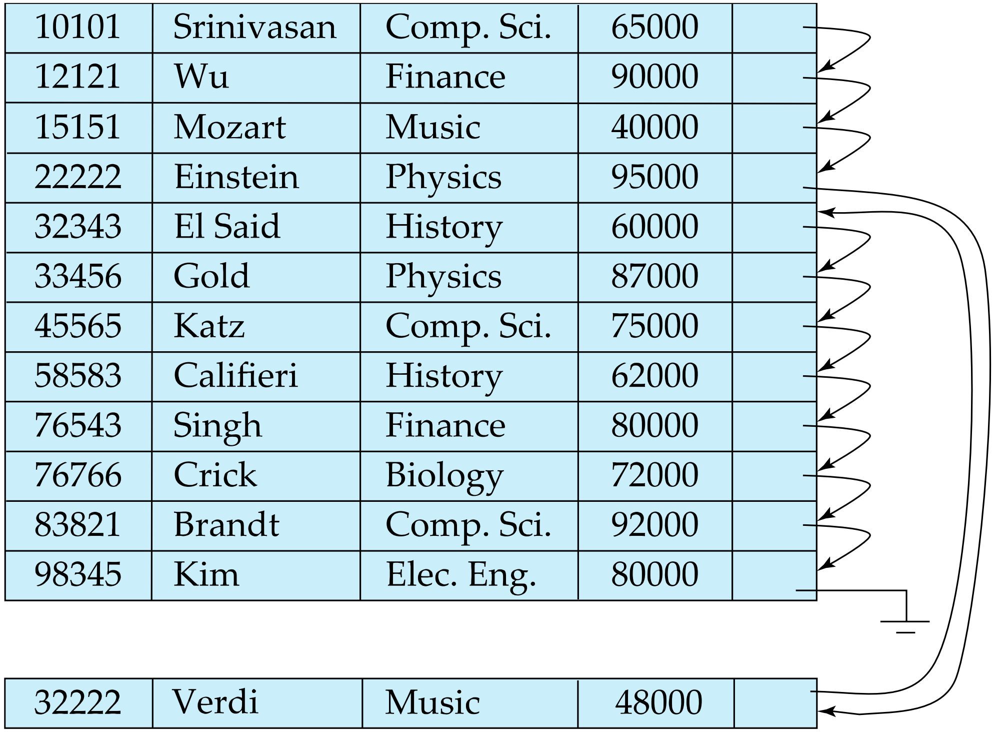
$$E(X) = \frac{1}{N} + \frac{2}{N} + \frac{3}{N} + \cdots + \frac{N}{N}$$

$$= \frac{1}{N}\sum_{k=1}^{N} k = \frac{1}{N} \times \frac{N(N+1)}{2} = \frac{(N+1)}{2} \approx \frac{N}{2}$$

- Binary Search for *N* ordered items, the maximum number of comparisons *k* is ≈ $\log_2 N$
  - Consider a line of length *N,* and keep dividing the length into two until only one item remains
  - $\frac{N}{2^k} = 1 \Rightarrow N = 2^k \Rightarrow k = \log_2 N$

Sequential File Organization

- Deletion – use pointer chains
- Insertion – locate the position where the record is to be inserted
  - if there is free space insert there
  - if no free space, insert the record in an overflow block
  - In either case, pointer chain must be updated
- Need to reorganize the file from time to time to restore sequential order



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

32222	Verdi	Music	48000

Multitable Clustering File Organization

Store several relations in one file using a **multitable clustering** file organization

department

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Physics	Watson	70000

instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

multitable clustering of *department* and *instructor*

Comp. Sci.	Taylor	100000	
10101	Srinivasan	Comp. Sci.	65000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
Physics	Watson	70000	
33456	Gold	Physics	87000

Multitable Clustering File Organization

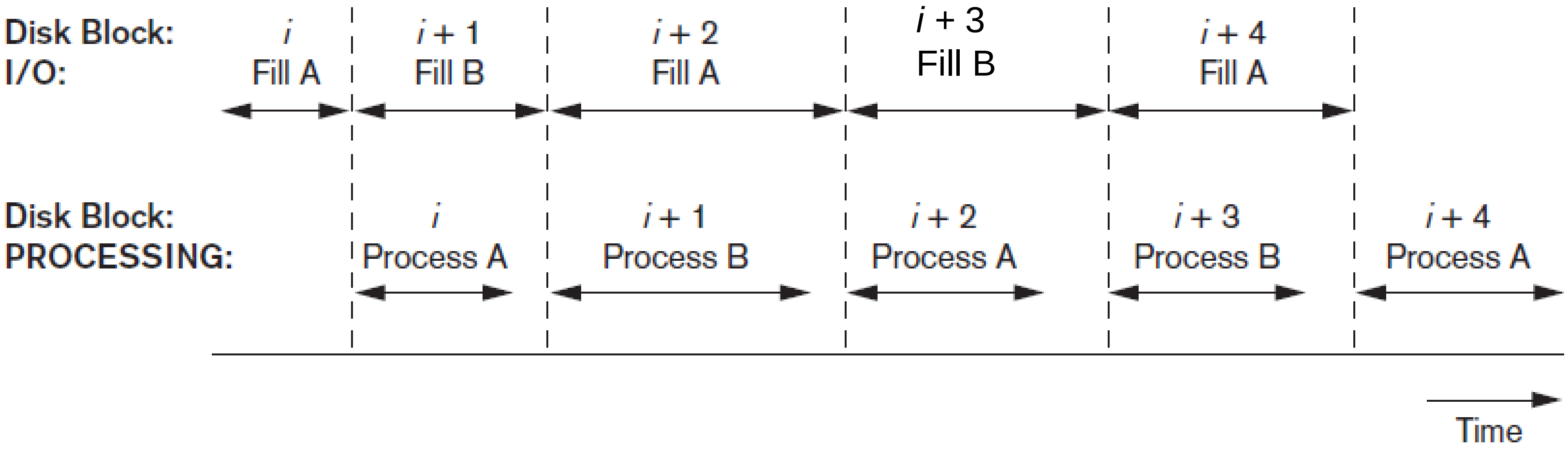
- good for queries involving *department* $\bowtie$ *instructor*, and for queries involving one single department and its instructors
- bad for queries involving only *department*
- results in variable size records
- Can add pointer chains to link records of a particular relation

Storage Access

- Blocks are units of both storage allocation and data transfer
- Database system seeks to minimize the number of block transfers between the disk and memory. We can reduce the number of disk accesses by keeping as many blocks as possible in main memory
- **Buffer** – portion of main memory available to store copies of disk blocks
- **Buffer manager** – subsystem responsible for allocating buffer space in main memory

Buffering of Blocks

- Double buffering can be used to read continuous stream of blocks
- Use of two buffers, A and B, for reading from disk
- More than 2 buffers can be used



Index Entries

- Indexing mechanisms is used to speed up access to desired data.
  - E.g., author catalog in library
- **Search Key** - attribute to a set of attributes used to look up records in a file.
- An **index is a file** consists of records (called **index entries**) of the form

search-key	pointer

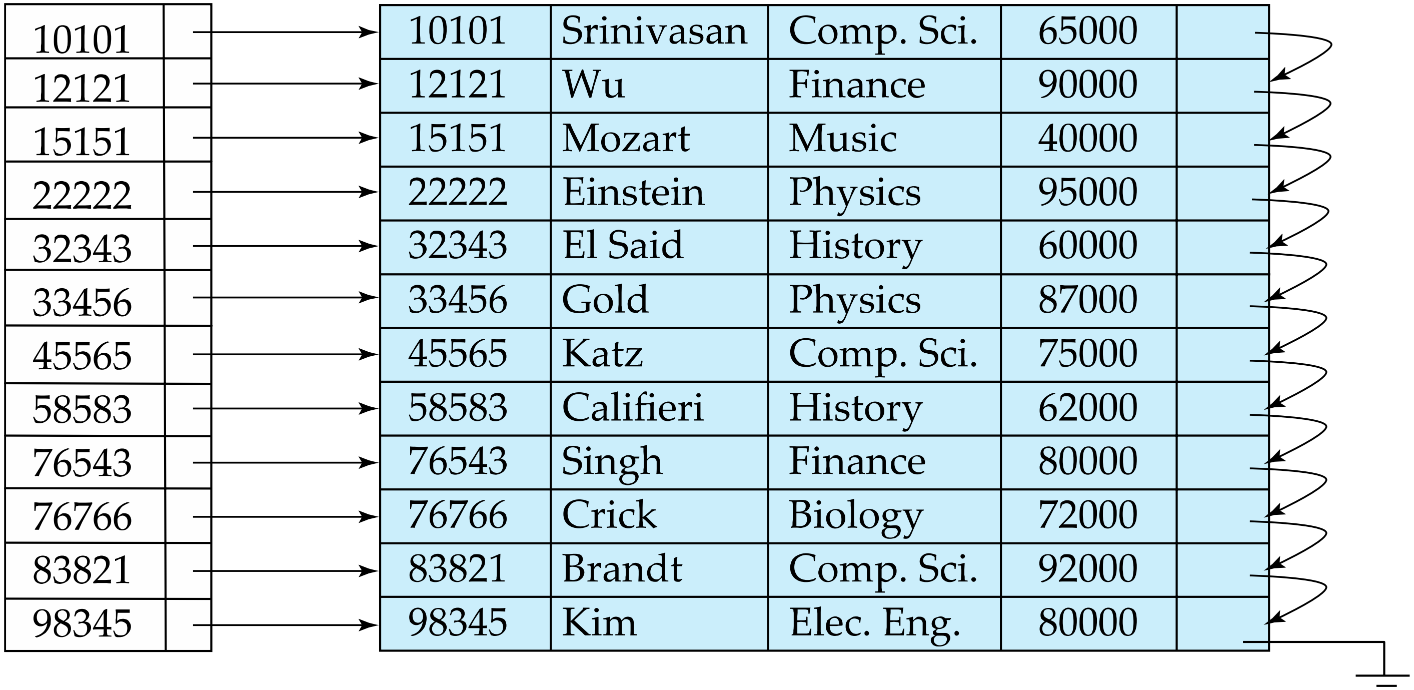
- Index files are typically much smaller than the original file

Ordered Indices

- In an **ordered index,** index entries are stored sorted on the search key value
- **Primary index:** in a sequentially ordered file, the index whose search key specifies the sequential order of the file
  - 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
  - Key values may or may not be unique
- **Indexed-sequential file (ISAM – Indexed Sequential Access Method):** sequential file ordered on a search key, with an index on the search key

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

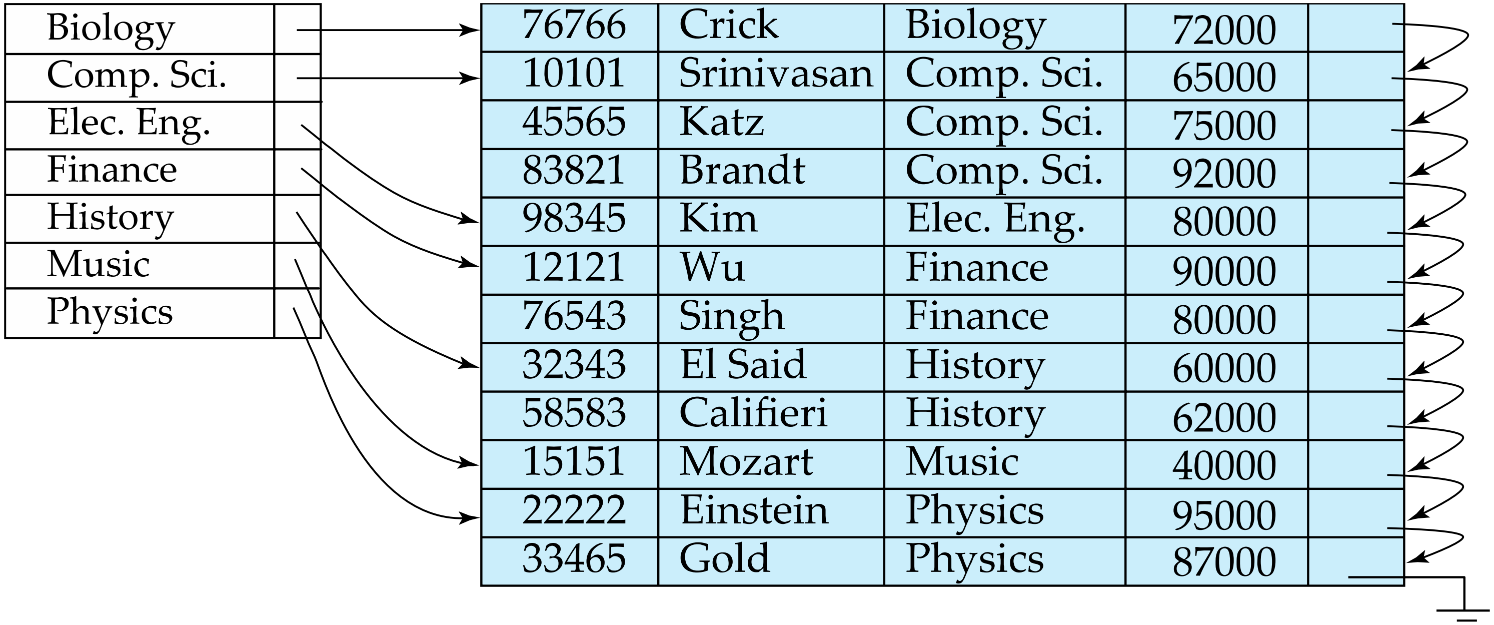


10101	
12121	
15151	
22222	
32343	
33456	
45565	
58583	
76543	
76766	
83821	
98345	

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

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

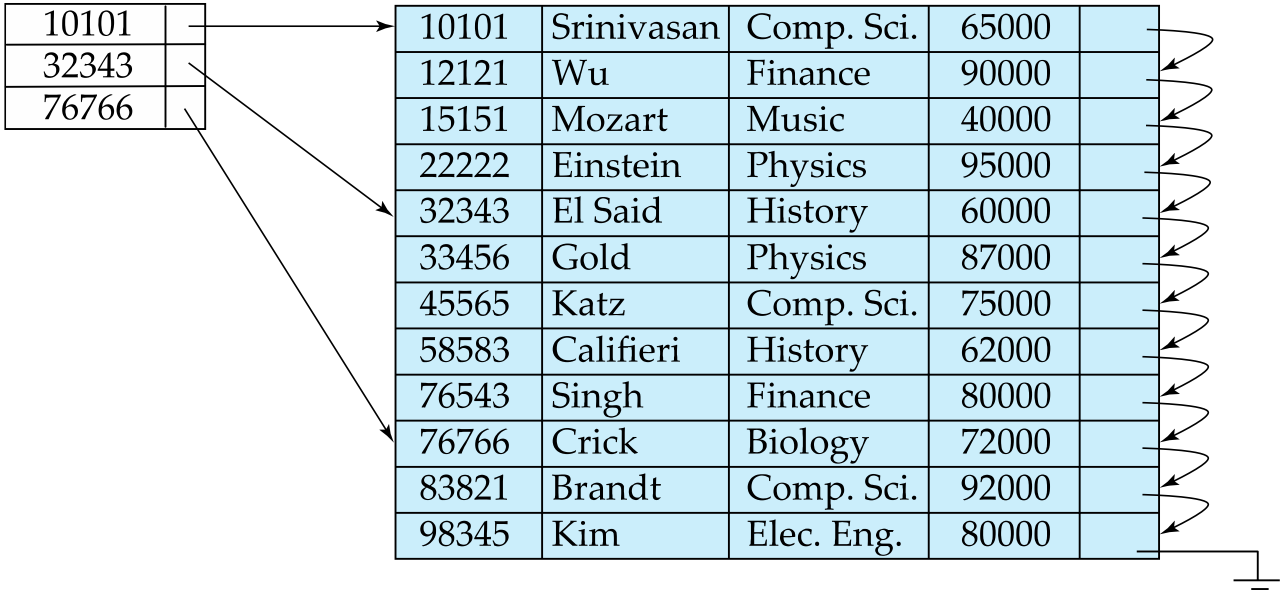


Biology	
Comp. Sci.	
Elec. Eng.	
Finance	
History	
Music	
Physics	

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	

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

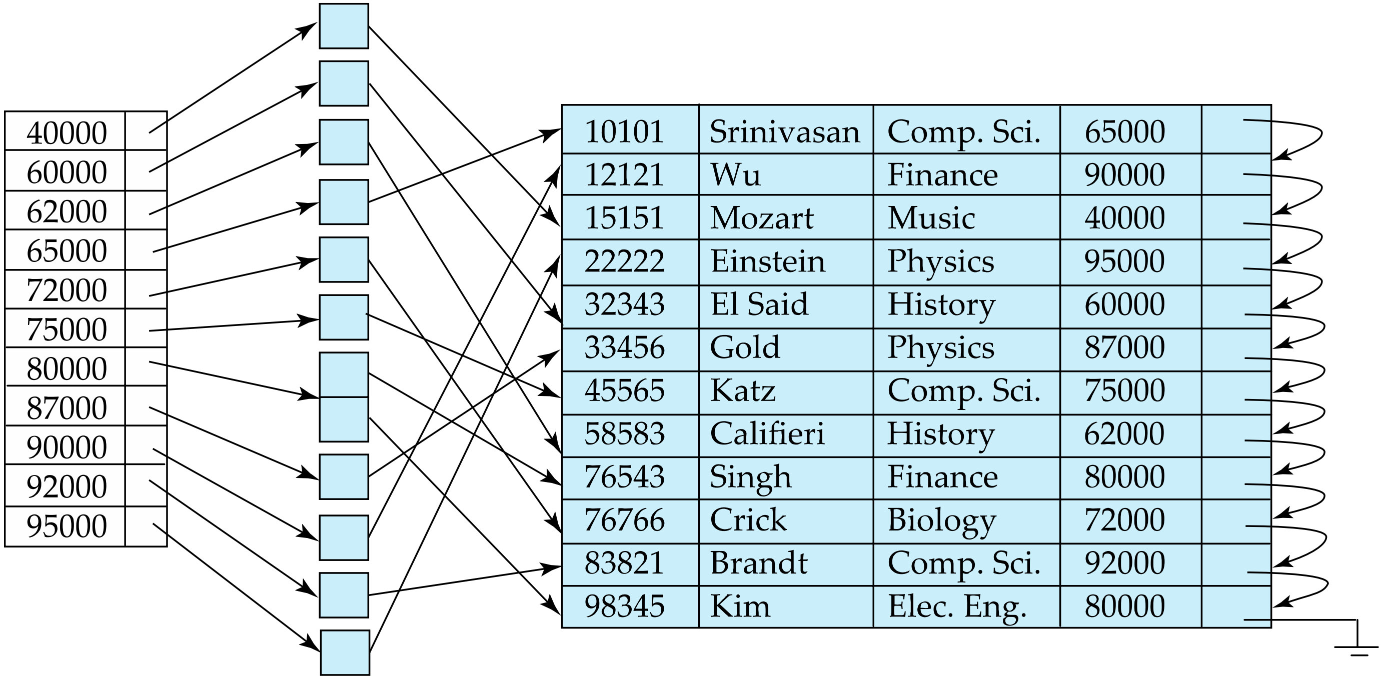


10101	
32343	
76766	

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	

Secondary Index

- Secondary index on salary field of instructor

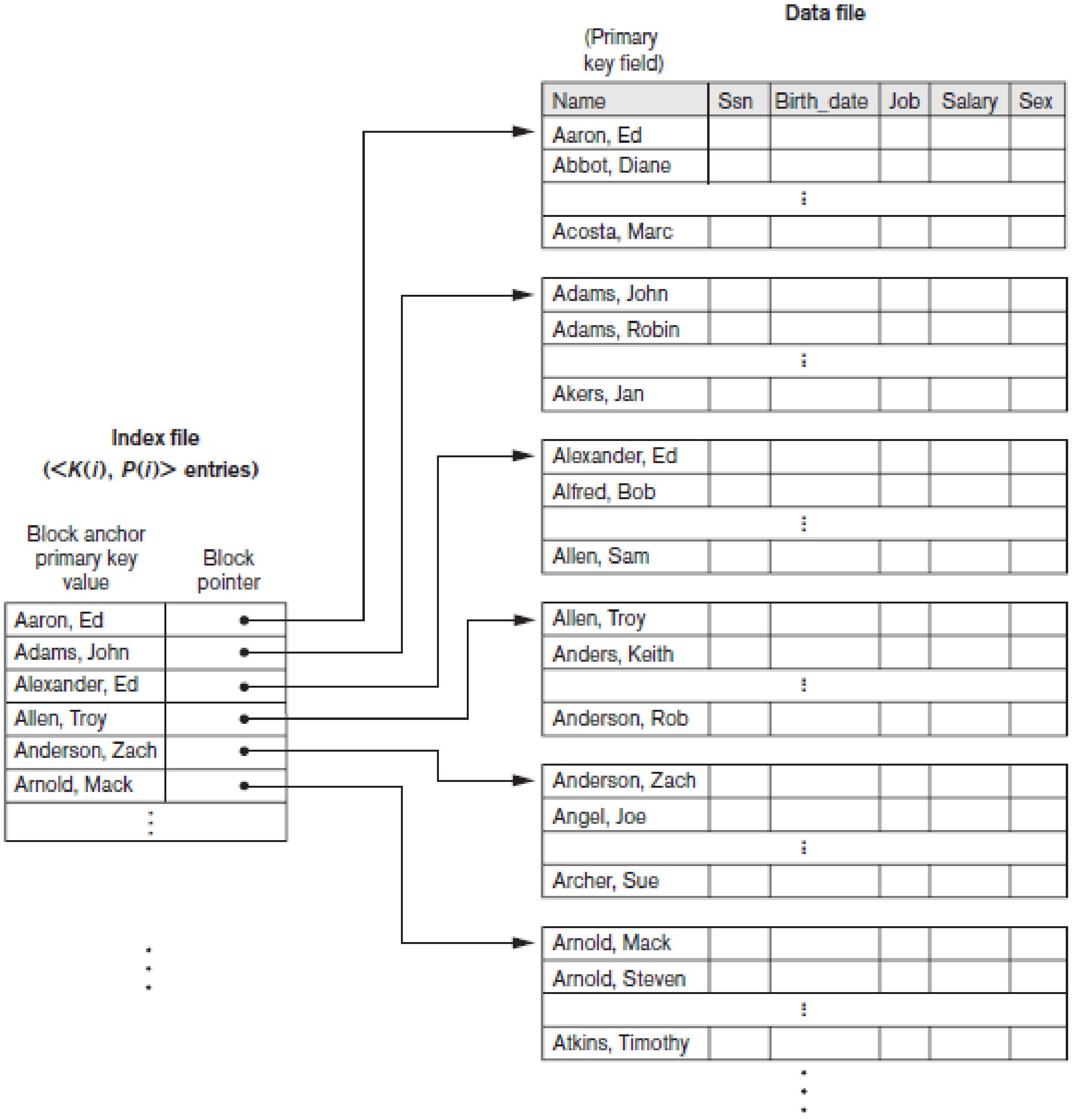


40000	
60000	
62000	
65000	
72000	
75000	
80000	
87000	
90000	
92000	
95000	

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	

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

Primary Index



Index file
(<*K*(*i*), *P*(*i*)> entries)

Block anchor primary key value	Block pointer
Aaron, Ed	●
Adams, John	●
Alexander, Ed	●
Allen, Troy	●
Anderson, Zach	●
Arnold, Mack	●
⋮	

⋮

Data file

(Primary key field)

Name	Ssn	Birth_date	Job	Salary	Sex
Aaron, Ed					
Abbot, Diane					
⋮					
Acosta, Marc					

Adams, John					
Adams, Robin					
⋮					
Akers, Jan					

Alexander, Ed					
Alfred, Bob					
⋮					
Allen, Sam					

Allen, Troy					
Anders, Keith					
⋮					
Anderson, Rob					

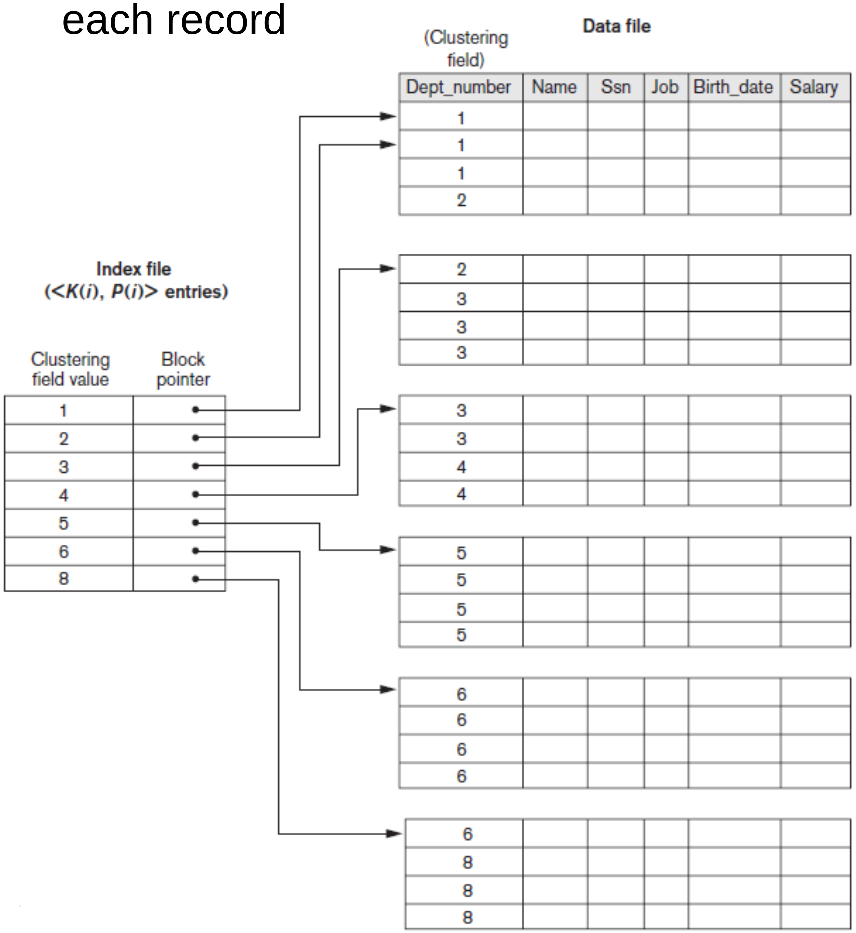
Anderson, Zach					
Angel, Joe					
⋮					
Archer, Sue					

Arnold, Mack					
Arnold, Steven					
⋮					
Atkins, Timothy					

⋮

Clustering Index

- Clustering field
  - records are physically ordered on a nonkey field without a distinct value for each record



Index file (<*K*(*i*), *P*(*i*)> entries)

Clustering field value	Block pointer
1	•
2	•
3	•
4	•
5	•
6	•
8	•

Data file

Dept_number	Name	Ssn	Job	Birth_date	Salary
1					
1					
1					
2					

2					
3					
3					
3					

3					
3					
4					
4					

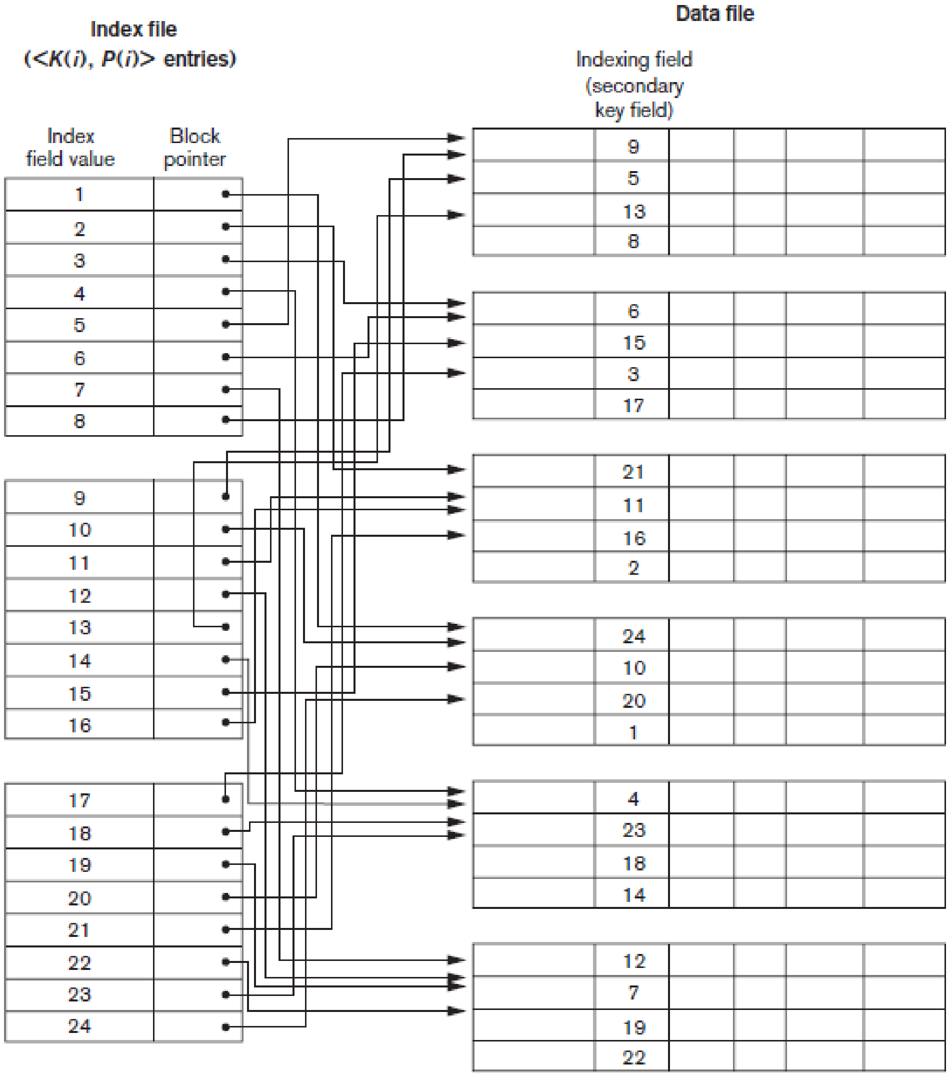
5					
5					
5					
5					

6					
6					
6					
6					

6					
8					
8					
8					

A clustering index on the Dept_number which is an ordering nonkey field of an Employee file

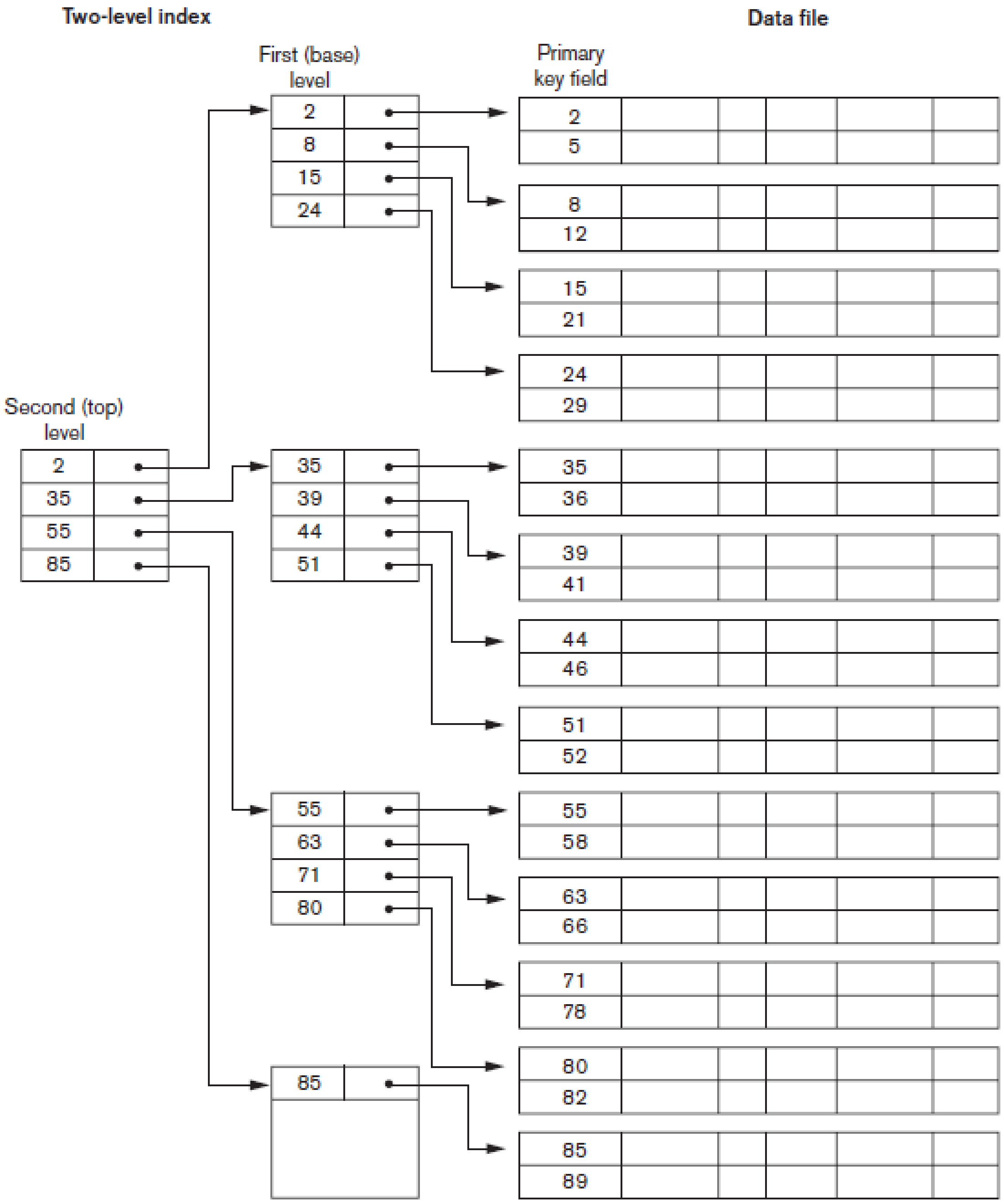
Secondary Index



Dense secondary index (with block pointers) on a nonordering key field of a file.

Multilevel Index

- If index does not fit in memory, access becomes expensive
- Solution: treat index kept on disk as a sequential file and construct a sparse index on it
  - outer index – a sparse index of the basic index
  - inner index – the basic 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



A two-level ISAM (indexed sequential access method) organization