Indexing

What is an Index?

- A simple index is a table containing an ordered list of keys and reference fields.
  - e.g. the index of a book
- In general, indexing is another way to handle the searching problem.

Uses of an index

1. An index lets us **impose order on a file** without rearranging the file.

2. Indexes provide **multiple access paths** to a file.

   – e.g. library catalog providing search for author,book and title

3. An index can provide **keyed access to variable-length record** files.

A simple index for a pile file

	Label ID Title Composer Artist
17	LON|2312|Symphony No.9| Beethoven|Giulini
62	RCA|2626|Romeo and Juliet|Prokofiev|Maazel
117	WAR|23699|Nebraska| …
152	ANG|3795|Violin Concerto| …

Address of record
(i.e. Byte offset)

Primary key = (Label, ID)

- Index is sorted (in main memory).
- Records appear in file in the order they are entered.

Index array:

Key	Reference
ANG3795	152
LON2312	17
RCA2626	62
WAR23699	117

- How to search for a recording with given LABEL ID?
  - Binary search in the index and then seek for the record in position given by the reference field.

Operations to maintain an indexed file

- Create the original empty index and data files.
- Load the index file into memory before using it.
- Rewrite the index file from memory after using it.
- Add data records to the data file.
- Delete records from the data file
- Update records in the data file.
- Update the index to reflect changes in the data file

Rewrite the index file from memory

- When the data file is closed, the index in memory needs to be written to the index file.
- An important issue to consider is what happens if the rewriting does not take place (e.g. power failures, turning machine off, etc.)
- Two important safeguards:
  - Keep a status flag in the header of the index file.
  - If the program detects the index is out of date it calls a procedure that reconstructs the index from the data file.

Record Addition

1. Append the new record to the end of the data file.
2. Insert a new entry to the index in the right position.
   - needs rearrangement of the index

   Note: this rearrangement is done in the main memory.

Record Deletion

- Use the techniques for reclaiming space in files when deleting records from the data file.
- We must also delete the corresponding entry from the index in memory.

Record Updating

There are two cases to consider:

1. The update changes the value of the key field:
   – Treat this as a deletion followed by an insertion
2. The update does not affect the key field:
   – If record size is unchanged, just modify the data record. If record size is changed treat this as a delete/insert sequence.

Indexing by Multiple Keys

- We could build additional indexes for a file to provide multiple views of a data file.
  - e.g. Find all recordings of Beethoven’s work.
- LABEL ID is a **primary key.**
- There may be **other search keys**: title, composer, artist.
- We can build **secondary indexes**.

Composer index:

Composer	Primary key
Beethoven	ANG3795
Beethoven	DG139201
Beethoven	DG18807
Beethoven	RCA2626
Corea	WAR23699
Dvorak	COL31809
Prokofiev	LON2312

- Note that reference is to the primary key rather than to the byte offset.

Retrieval using combinations of secondary keys

- Secondary indexes are useful in allowing the following kinds of queries:
  - Find all recordings of Beethoven's work.
  - Find all recordings titled "Violin concerto"
  - Find all recordings with composer Beethoven and title Symphony No.9.
- Boolean operators "and", "or" can be used to combine secondary search keys to qualify a request.

Example

- The last query is executed as follows:

Matches from composer index	Matches from title index	Matched list (logical "and")
ANG3795	ANG3795	ANG3795
DG139201	COL31809	DG18807
DG18807	DG18807	
RCA2626		

Problems with simple indexes

If index does not fit in memory:

1. Seeking the index is slow (binary search):
   - We don't want more than 3 or 4 seeks for a search.

N	**Log(N+1)**
15 keys	4
1000	~10
100,000	~17
1,000,000	~20

2. Insertions and deletions take O(N) disk accesses.

Indexes too large to fit into Memory

- Two main alternatives:
  1. Tree-structured (multi-level) index such as B+trees.
  2. Hashed organization (when access speed is a top priority)

Multilevel Indexing and B+ Trees

Outline

- Single-level index
- Multi-level index
- B+tree index

All can be classified as:

- Dense vs. sparse index
- Primary vs. secondary index
- Clustered vs. unclustered index

Indexed Sequential Access

Provide a choice between two alternative views of a file:

1. **Indexed**: the file can be seen as a set of records that is indexed by key; or
2. **Sequential:** the file can be accessed sequentially (physically contiguous records), returning records in order by key.

Example of applications

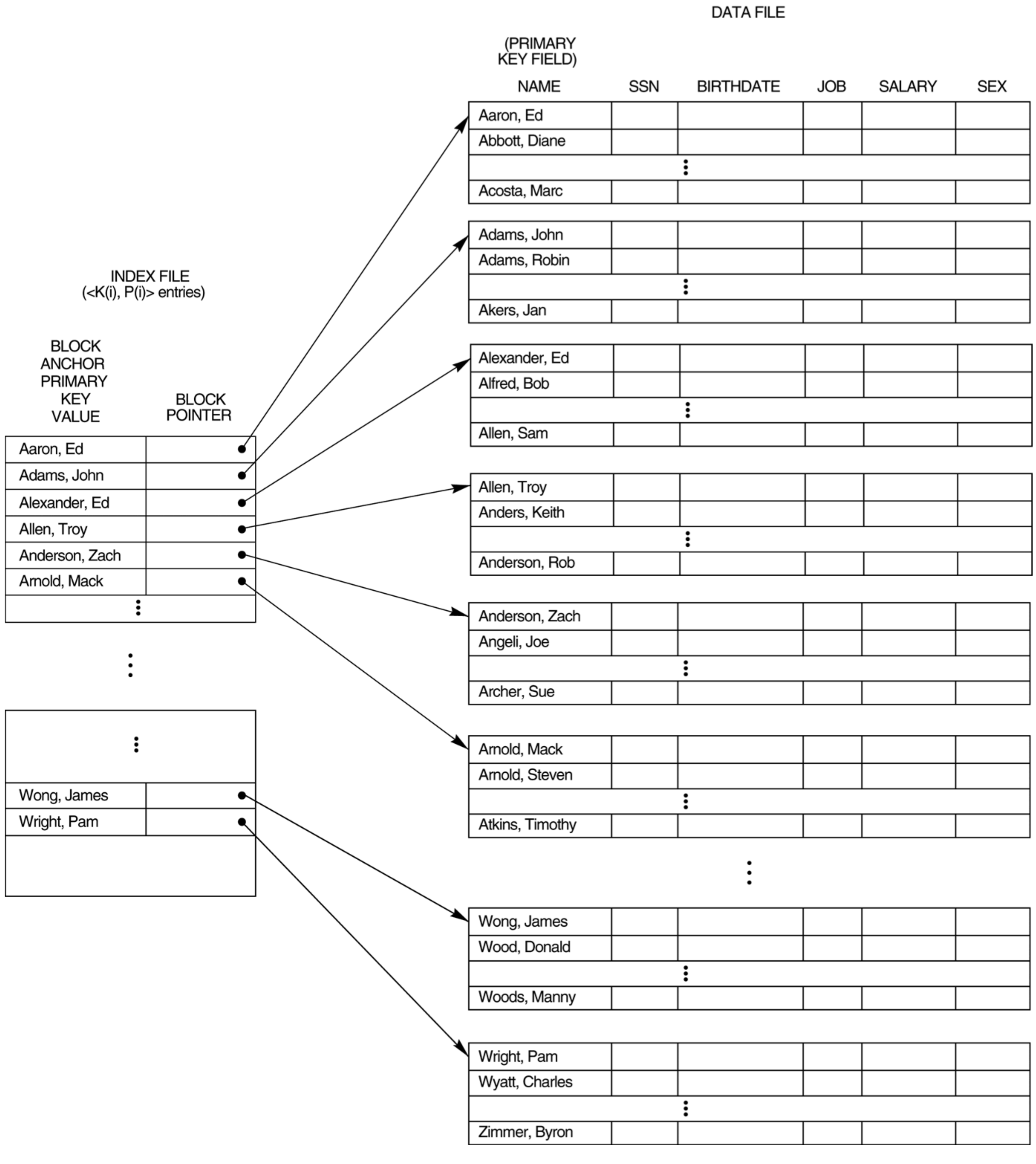
- Student record system in a university:
  - Indexed view: access to individual records
  - Sequential view: batch processing when posting grades
- Credit card system:
  - Indexed view: interactive check of accounts
  - Sequential view: batch processing of payments

The initial idea: Single level index

- The data file is ordered on a ***key field***.
  - The records are grouped into blocks in a sorted way.
- Construct a simple, single level index for these blocks.
  - Includes one index entry *for each block* in the data file; the index entry has the key field value for the *first record* in the block, which is called the *block anchor*.
  - A similar scheme can use the *last record* in a block.

Single-level index on the ordering key field of the file.

- This is a **sparse (nondense)** index, since it includes an entry for each disk block of the data file (not for each record of the data file).
- This is also **primary index,** because data is ordered on the primary key field (no duplicates).
- This is also a **clustered index** because the data is in the same order as the search key.

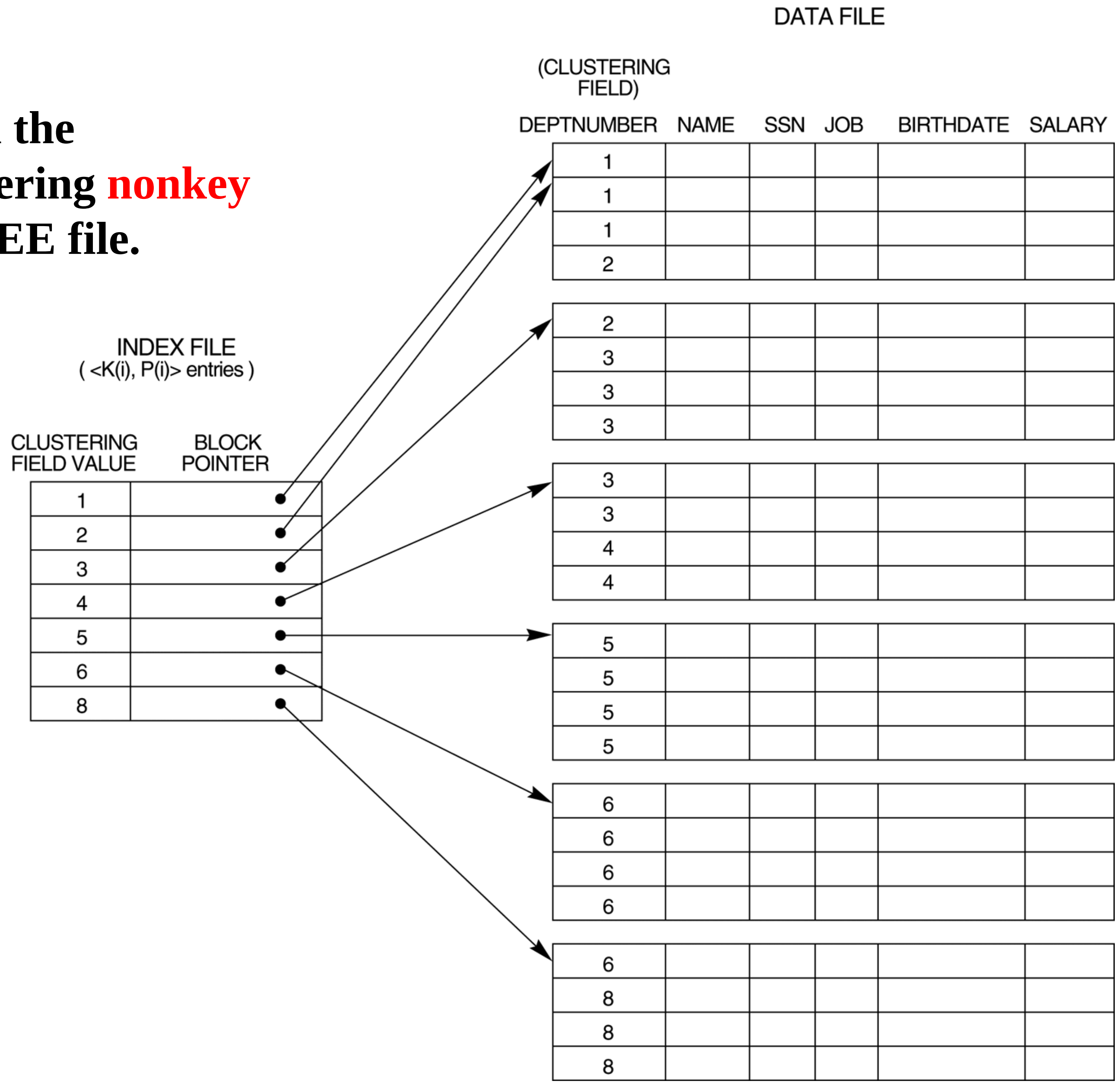


Another Example of a Clustered Single-Level Index

- The data file is ordered on a ***non-key field*** unlike primary index which requires that the ordering field of the data file have a distinct value for each record.
- Includes one index entry *for each distinct value* of the field; the index entry points to the first data block that contains records with that field value.
- It is another example of *sparse (nondense)* index.

Single-level index

A clustering index on the DEPTNUMBER ordering nonkey field of an EMPLOYEE file.



Single-Level Secondary Index

- A secondary index provides a secondary means of accessing a file for which some primary access already exists.
- The secondary index may be on a field which is a candidate key and has a unique value in every record, or a nonkey with duplicate values.
- The index is an ordered file with two fields.
  - The first field is the *indexing field*.
  - The second field is either a *block* pointer or a *record* pointer.

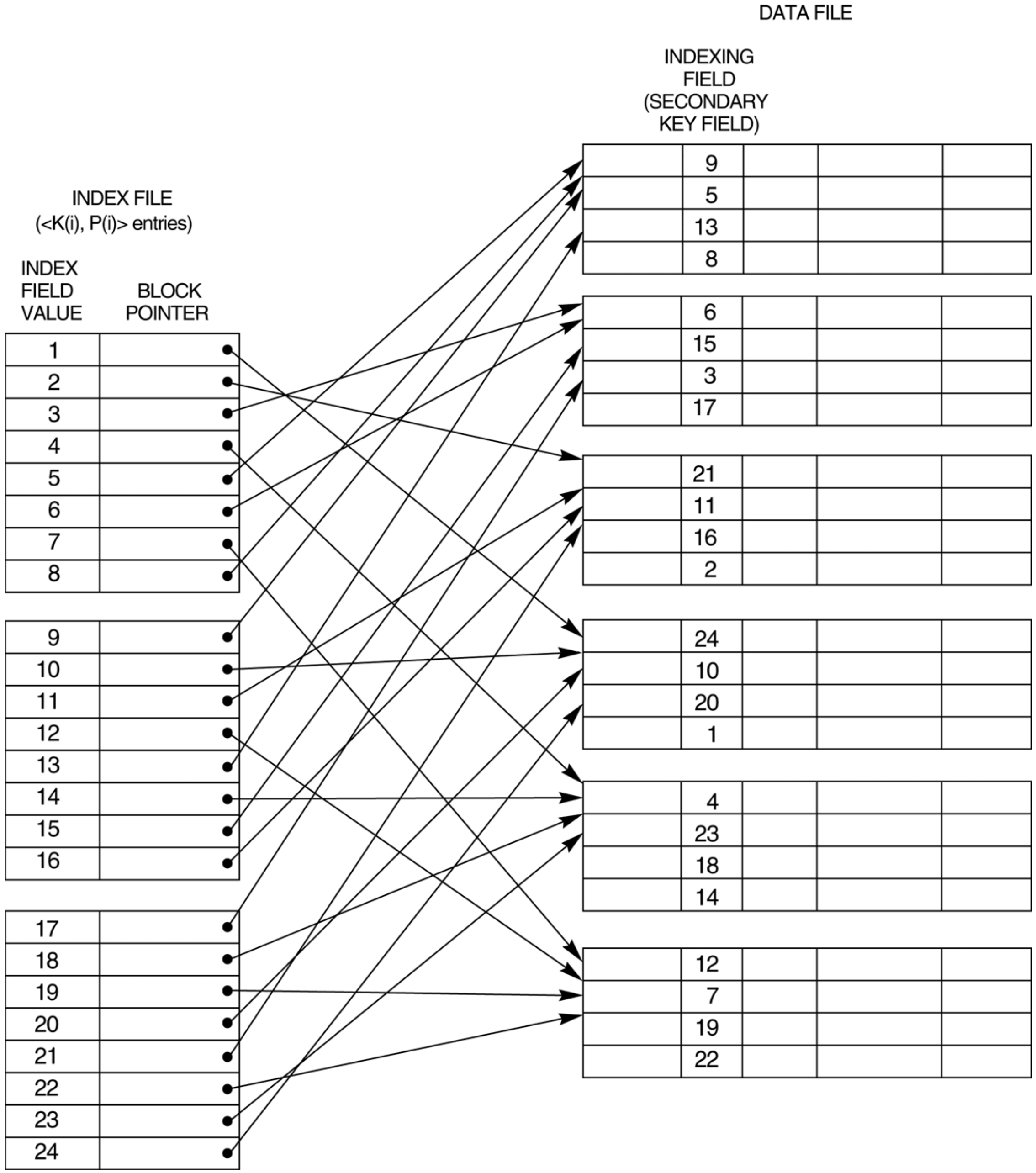
  There can be *many* secondary indexes for the same file.
- Includes one entry *for each record* in the data file; hence, it is a ***dense index.***

A **secondary index** on a candidate key (with block pointers)

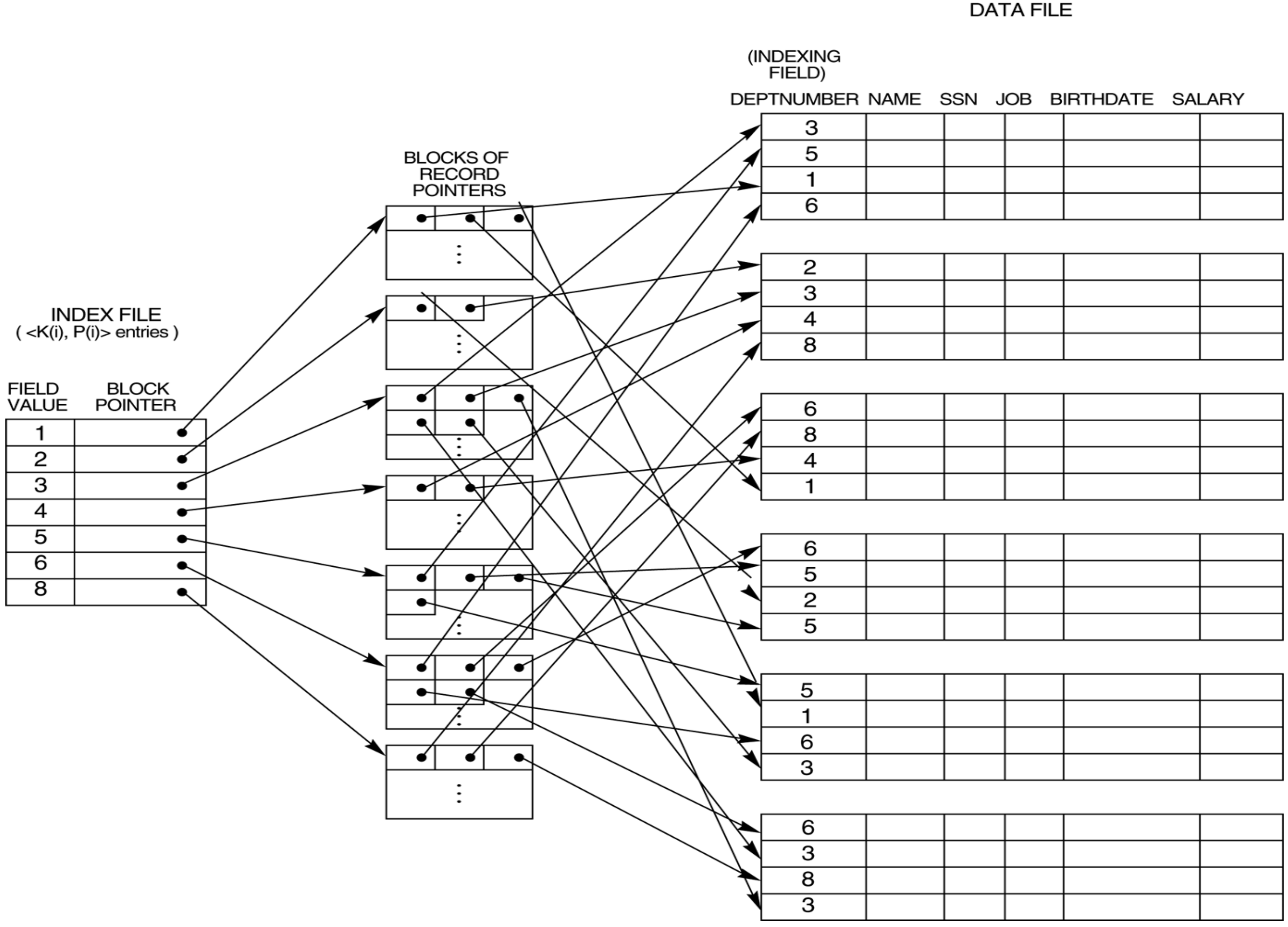
This is a **dense** index.

No duplicates.

Note that the data file is *not* ordered according to the index field. Therefore it is an **unclustered** index

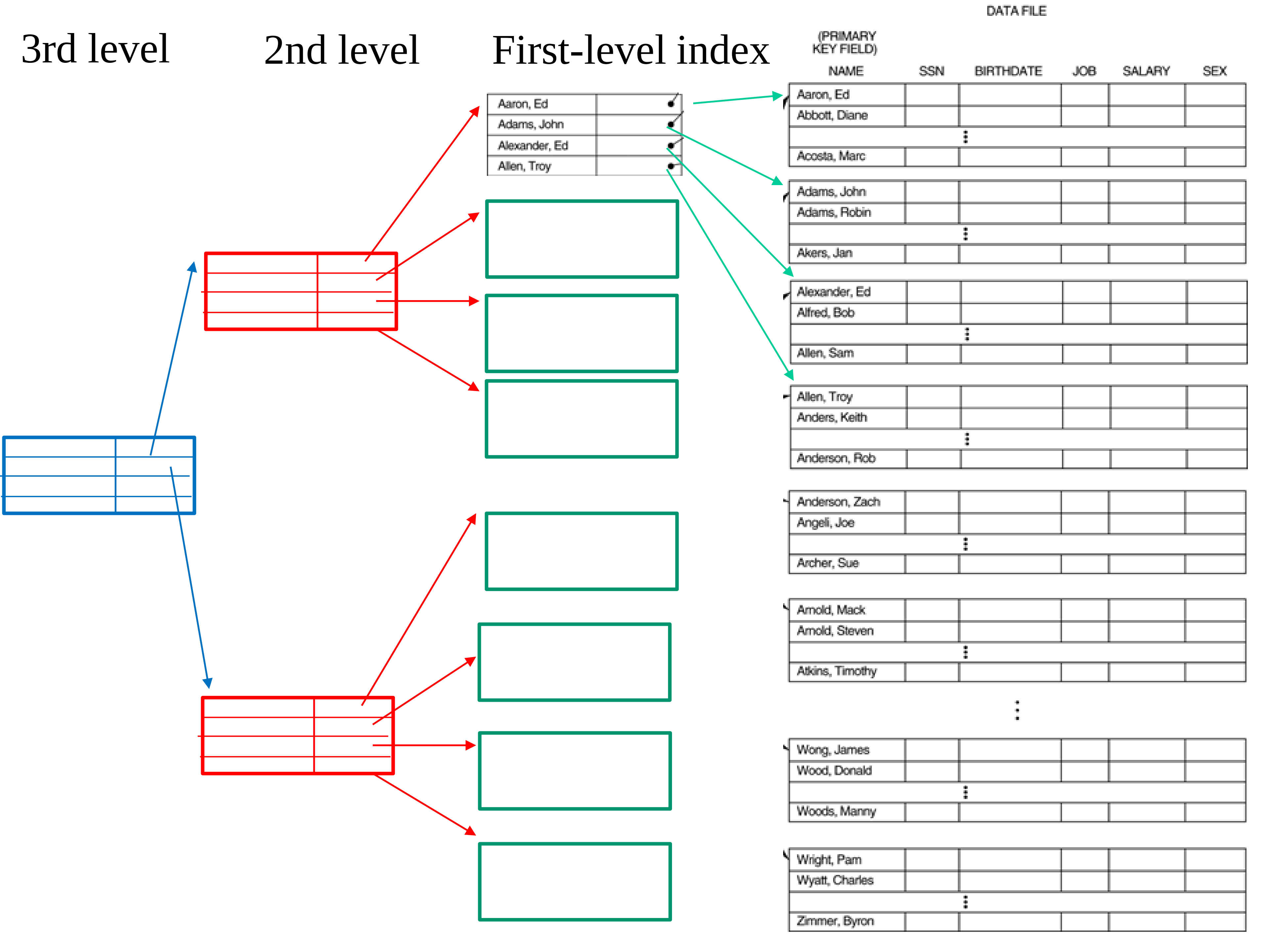


A secondary index (with record pointers) on a nonkey field implemented using one level of indirection so that index entries are of fixed length and have unique field values. This is an unclustered index.



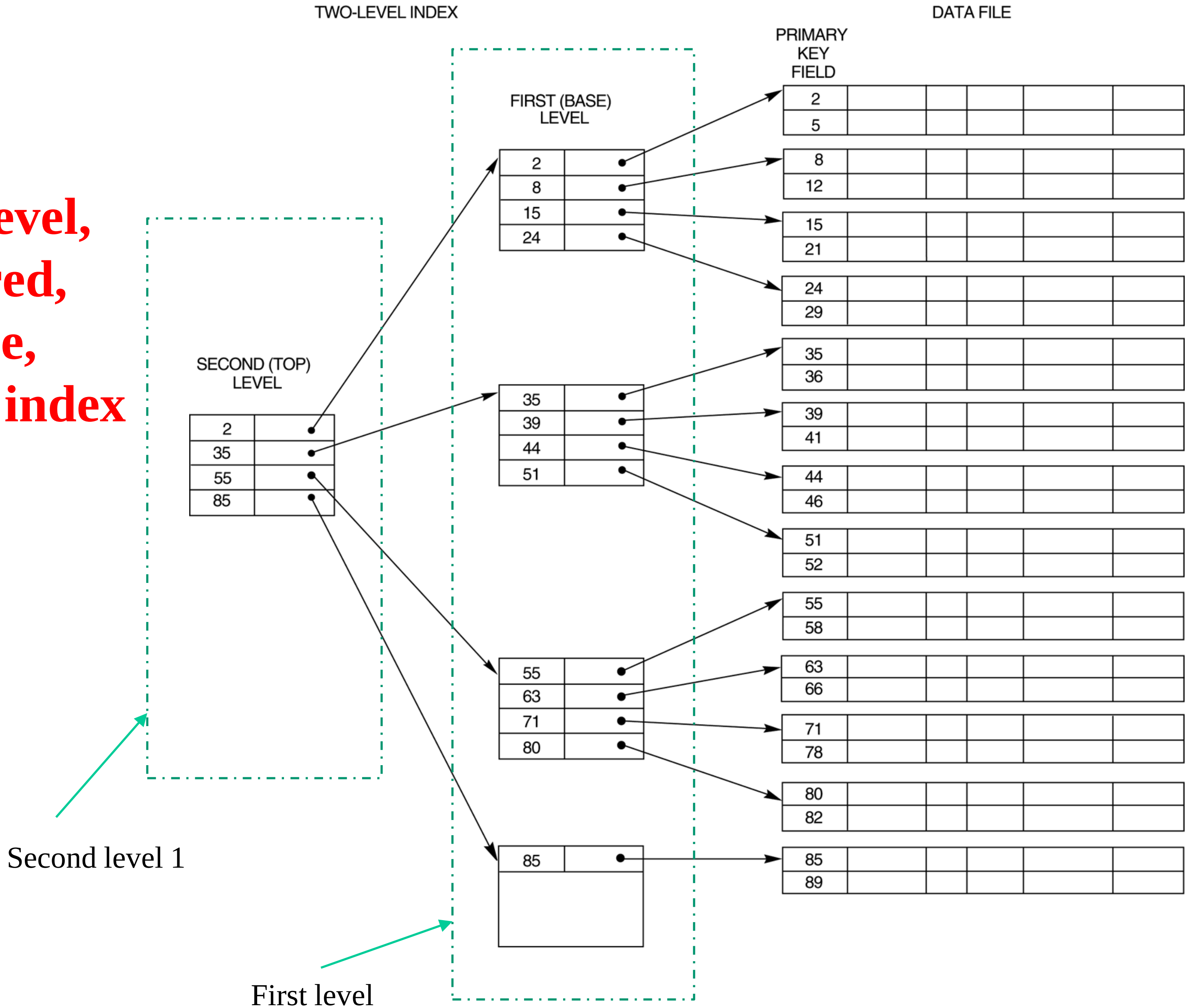
Multi-Level Indexes

- Because a single-level index is an ordered file, we can create an index *to the index itself* ; in this case, the original index file is called the *first-level index* and the index to the index is called the *second-level index*.
- We can repeat the process, creating a third, fourth, ..., top level until all entries of the *top level* fit in one disk block
- A multi-level index can be created for any type of first-level index (primary, secondary, clustering).



3rd level
2nd level
First-level index
Aaron, Ed
Adams, John
Alexander, Ed
Allen, Troy
DATA FILE
(PRIMARY KEY FIELD)
NAME
SSN
BIRTHDATE
JOB
SALARY
SEX
Aaron, Ed
Abbott, Diane
Acosta, Marc
Adams, John
Adams, Robin
Akers, Jan
Alexander, Ed
Alfred, Bob
Allen, Sam
Allen, Troy
Anders, Keith
Anderson, Rob
Anderson, Zach
Angeli, Joe
Archer, Sue
Arnold, Mack
Arnold, Steven
Atkins, Timothy
Wong, James
Wood, Donald
Woods, Manny
Wright, Pam
Wyatt, Charles
Zimmer, Byron

A two-level, clustered, sparse, primary index



Multi-Level Indexes

- Such a multi-level index is a form of *search tree*; however, insertion and deletion of new index entries is a severe problem because every level of the index is an *ordered file*.
- So this brings us to B+tree index structure.