File Organization

by

Dr. Pratik Roy

Introduction

- Databases are stored physically as files which are stored on magnetic disks.
  - Accessed using physical database file structures
- Typical database applications need only a small portion of the database at a time for processing.
- Whenever a certain portion of the data is needed, it must be located on disk, copied to main memory for processing, and then rewritten to the disk if the data is changed.
- Data stored on disk is organized as **files** of **records**.
- Each record is a collection of data values that can be interpreted as facts about entities, their attributes, and their relationships.
- Records should be stored on disk in a manner that makes it possible to locate them efficiently when they are needed.

File organization

- Determines how records are physically placed on the disk
- Determines how records are accessed

Placing File Records on Disk

- **Record:** Collection of related data values or items
  - Each value is formed of one or more bytes and corresponds to a particular field of the record.
- Records usually describe entities and their attributes.
- For example, an EMPLOYEE record represents an employee entity, and each field value in the record specifies some attribute of that employee, such as Name, Birth_date, Salary, or Supervisor.
- A collection of field names and their corresponding data types constitutes a record type or record format definition.
- A **data type**, associated with each field, specifies the types of values a field can take.

Placing File Records on Disk

- Data types
  - Numeric
  - String
  - Boolean
  - Date/time
- A **file** is a sequence of records.
- In many cases, all records in a file are of the same record type.
- If every record in the file has exactly the same size (in bytes), the file is said to be made up of **fixed-length records**.
- If different records in the file have different sizes, the file is said to be made up of **variable-length records**.

Placing File Records on Disk

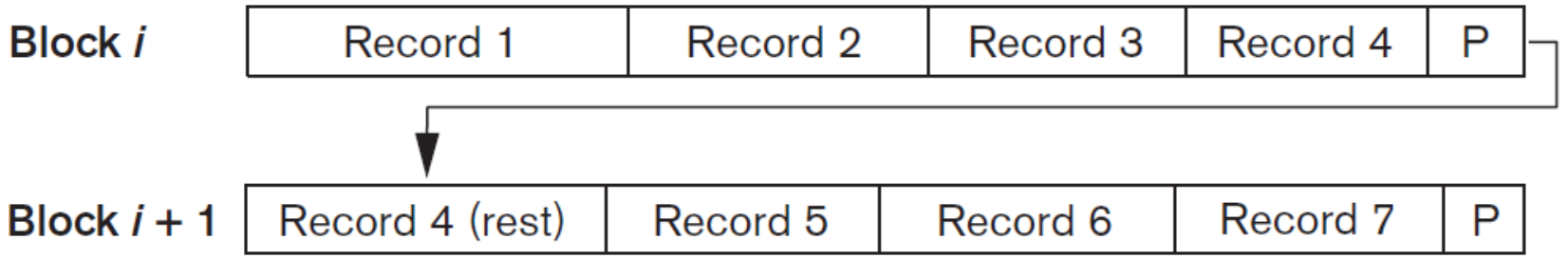
- Reasons for variable-length records
  - One or more fields have variable length
  - One or more fields are repeating
  - One or more fields are optional
  - File contains records of different types

Record Blocking

- Records of a file must be allocated to disk blocks because a block is the unit of data transfer between disk and memory.
- When the block size is larger than the record size, each block will contain numerous records.
- Although some files may have unusually large records that cannot fit in one block.

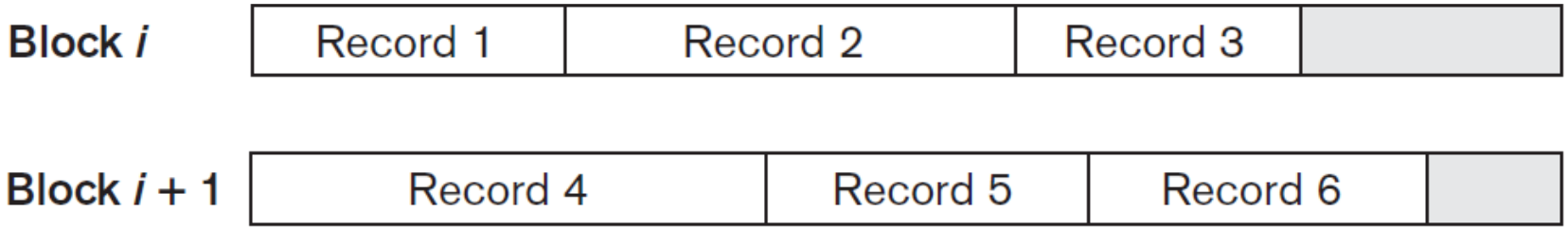
Spanned records

- Larger than a single block
- Pointer at end of first block points to block containing remainder of record.
- Records can span more than one block.



Unspanned records

- Records not allowed to cross block boundaries.



- **Blocking factor**
  - Average number of records per block for the file

Allocating File Blocks on Disk

- **Contiguous allocation**
  - file blocks are allocated to consecutive disk blocks.
- **Linked allocation**
  - each file block contains a pointer to the next file block.
- **Indexed allocation**
  - one or more index blocks contain pointers to the actual file blocks.

Files of Unordered Records (Heap Files)

- Records placed in file in order of insertion
- New records are inserted at the end of the file
- Such an organization is called a **heap** or **sequential file**.

Files of Ordered Records (Sorted Files)

- Records sorted by **ordering field**
  - Called **ordering key** if ordering field is a key field
  - Ordered (sequential) file

Indexing Structures for Files

- **Index - Auxiliary access structure**
- Indexes used to speed up record retrieval in response to certain search conditions.
- Index structures are additional files on disk that provide **secondary access paths**, which provide alternative ways to access the records without affecting the physical placement of records in the primary data file on disk.
- **Indexing field** is used to construct the index.
- Any field of the file can be used to create an index.
  - Multiple indexes can be constructed.
- Most prevalent types of indexes are based on ordered files (single-level indexes) and tree data structures (multilevel indexes, B^+-trees)
- Indexes can also be constructed based on hashing or other search data structures.

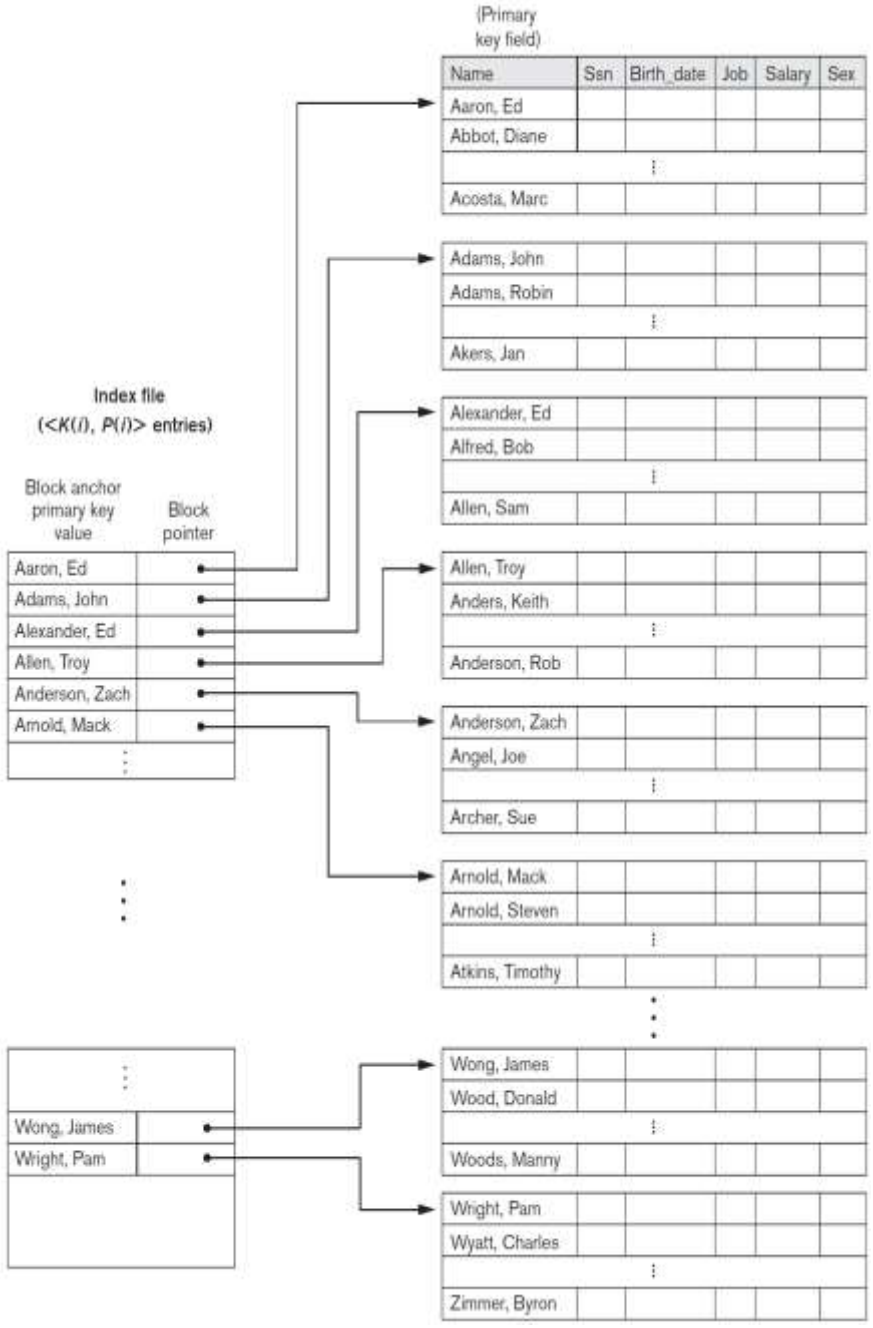
Types of Single-Level Ordered Indexes

- Ordered index similar to index in a textbook.
- **Indexing field (attribute)**
  - Index stores each value of the index field with list of pointers to all disk blocks that contain records with that field value.
- Values in index are ordered.
- **Primary index**
  - Specified on the ordering key field of ordered file of records.
- **Clustering index**
  - Used If the ordering field is not a key field i.e. if numerous records can have the same value for the ordering field.
- **Secondary index**
  - Can be specified on any nonordering field
  - Data file can have several secondary indexes

Primary Indexes

- Ordered file with two fields
  - Primary key, *K(i)*
  - Pointer to a disk block, *P(i)*
- One index entry in the index file for each block in the data file.
- Total number of entries in the index is same as the number of disk blocks in the ordered data file.
- First record in each block of the data file is called **anchor record** of the block.

Primary Indexes



Dense and Sparse Index

- Indexes may be dense or sparse.
- **Dense index** has an index entry for every search key value (hence every record) in the data file.
- **Sparse index** has index entries for only some of the search values.
- Sparse index has fewer entries than the number of records in the file.
- Primary index is a sparse index, since it includes an entry for each disk block of the data file and the keys of its anchor record rather than for every search value (or every record).

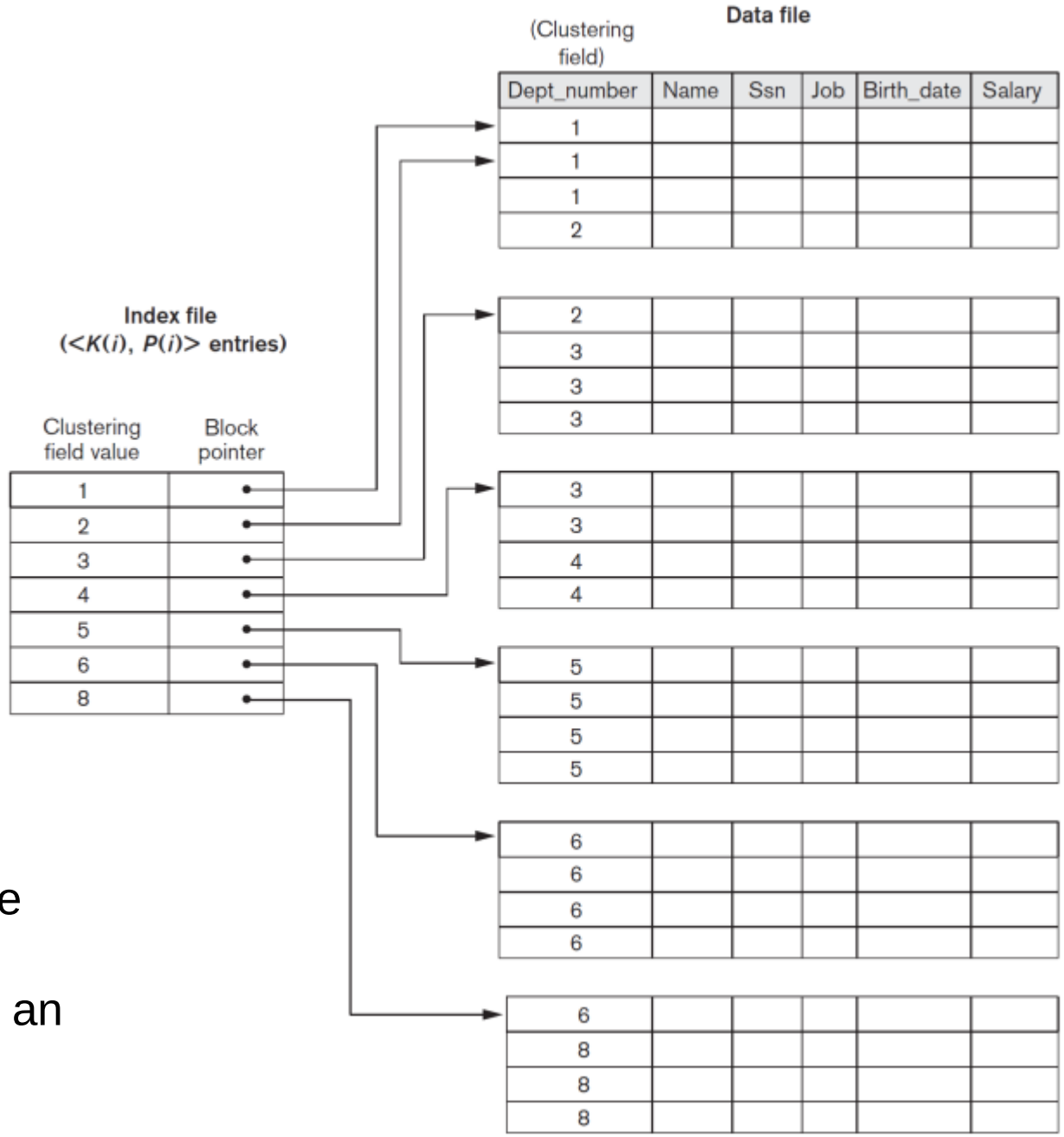
Primary Indexes

- **Major problem:** Insertion and deletion of records
- If we attempt to insert a record in its correct position in the data file, we must not only move records to make space for the new record but also change some index entries, since moving records will change the anchor records of some blocks.
- Move records around and change index values.
- **Solutions**
  - Use unordered overflow file
  - Use linked list of overflow records for each block in the data file.

Clustering Indexes

- If file records are physically ordered on a nonkey field without a distinct value for each record, that field is called **clustering field**.
- Data file is called **clustered file**.
- We can create clustering index to speed up retrieval of all the records that have the same value for the clustering field.
- This differs from primary index, which requires that the ordering field of the data file have a distinct value for each record.
- A clustering index is an ordered file with two fields:
  - First field is of the same type as the clustering field of the data file
  - Second field is a disk block pointer
- There is one entry in the clustering index for each distinct value of the clustering field, and it contains the value and a pointer to the first block in the data file that has a record with that value for its clustering field.

Clustering Indexes

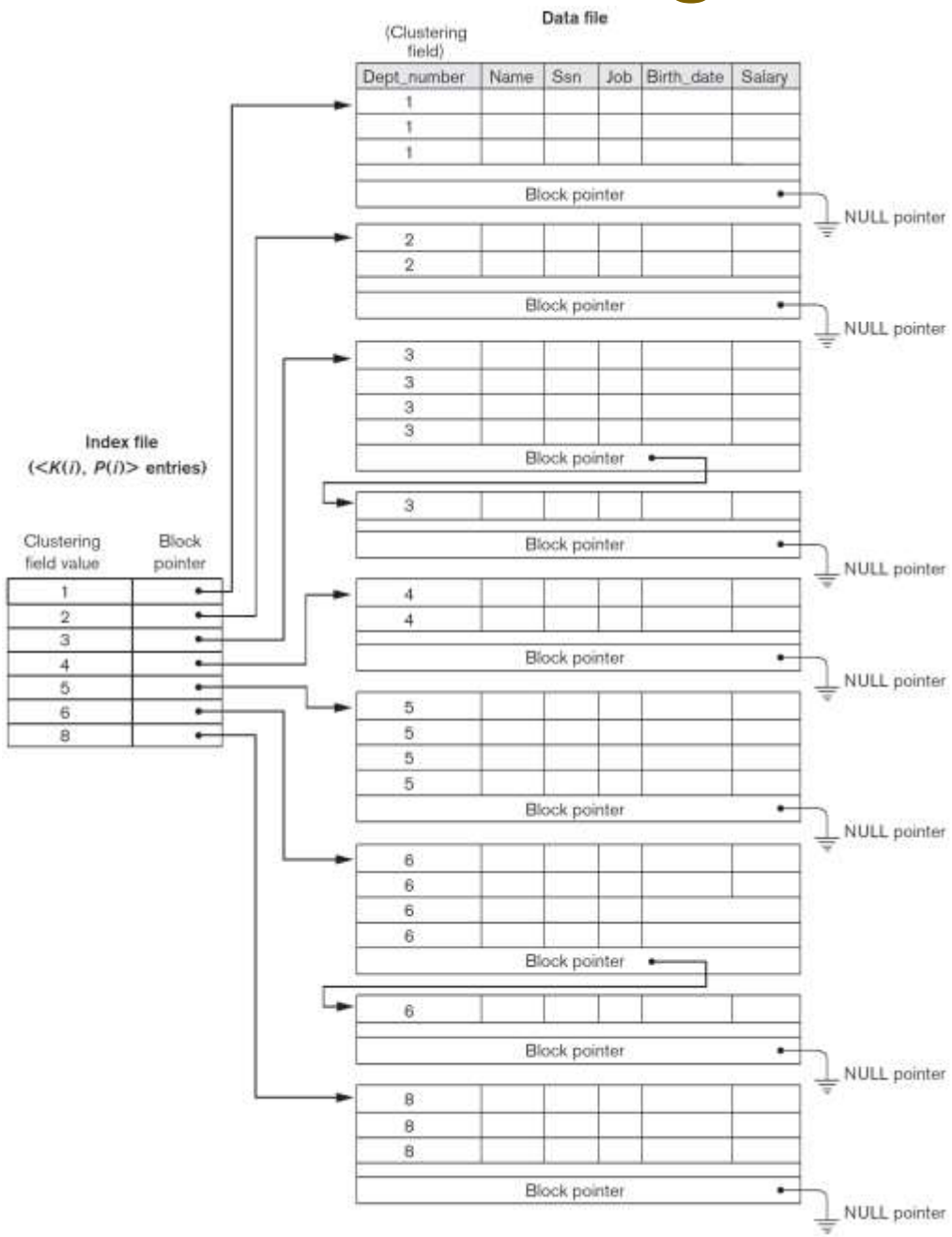


A clustering index on the Dept_number ordering nonkey field of an EMPLOYEE file

Clustering Indexes

- Record insertion and deletion cause problems because data records are physically ordered.
- To reduce the problem of insertion, it is common to reserve a whole block (or a cluster of contiguous blocks) for each value of the clustering field.
  - All records with that value are placed in the block (or block cluster).
- Clustering index is nondense index because it has an entry for every distinct value of the indexing field, which is a nonkey by definition and has duplicate values rather than a unique value for every record in the file.

Clustering Indexes

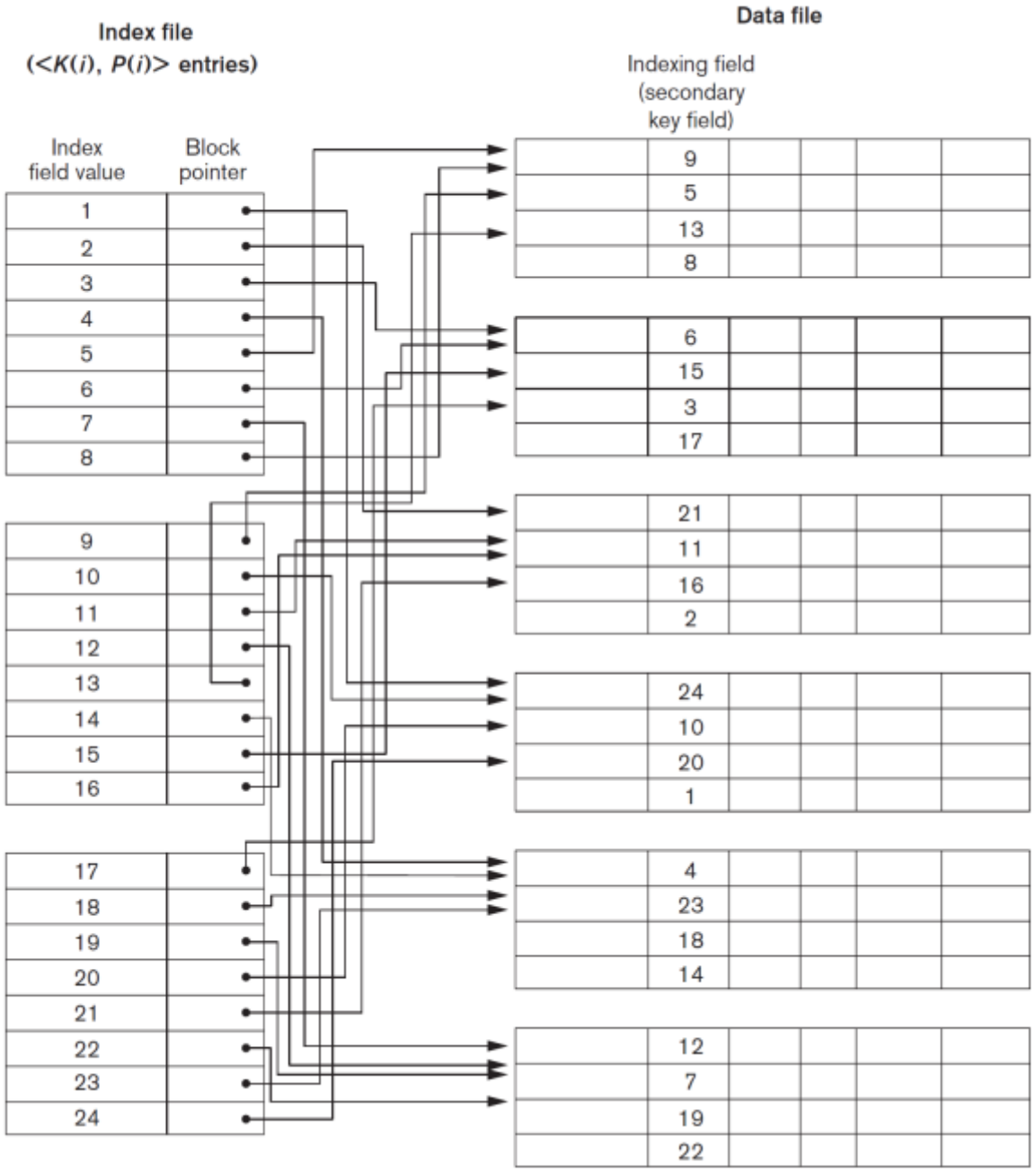


Clustering index with a separate block cluster for
each group of records that share the same value
for the clustering field

Secondary Indexes

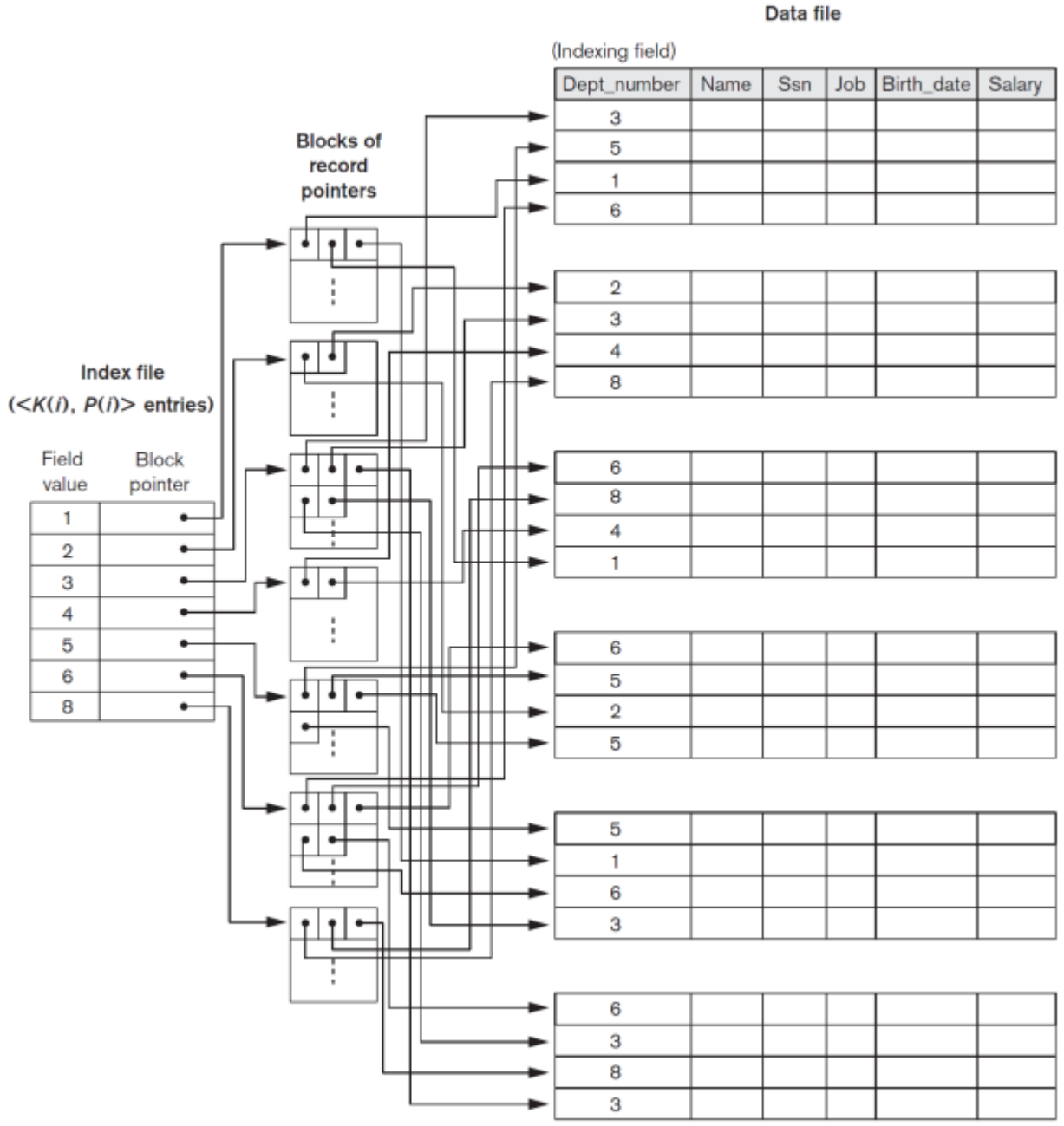
- Provide secondary means of accessing a data file
  - Some primary access exists
- Data file records could be ordered or unordered
- Secondary index may be created on a field that is a candidate key and has a unique value in every record, or on a nonkey field with duplicate values.
- Index is ordered file with two fields:
  - First field is of the same data type as some nonordering field of the data file that is an indexing field.
  - Second field is either a block pointer or a record pointer.

Secondary Indexes (on nonordering key field)



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

Secondary Indexes (on nonkey, nonordering field)



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.

Types of Single-Level Ordered Indexes

	Index Field Used for Physical Ordering of the File	Index Field Not Used for Physical Ordering of the File
Indexing field is key	Primary index	Secondary index (Key)
Indexing field is nonkey	Clustering index	Secondary index (NonKey)

Type of Index	Number of (First-level) Index Entries	Dense or Nondense (Sparse)	Block Anchoring on the Data File
Primary	Number of blocks in data file	Nondense	Yes
Clustering	Number of distinct index field values	Nondense	Yes/no[a]
Secondary (key)	Number of records in data file	Dense	No
Secondary (nonkey)	Number of records[b] or number of distinct index field values[c]	Dense or Nondense	No