- File Structures
- Heap files
  - Overview
  - Performance
    - Equality Searching
    - Deleting
    - Range Search
    - Inserting
- Sorted files
  - Advantages
  - Performance
    - Range / Equality Searching
    - Inserting
  - Maintaining Sorted Order Solution
- Index Files
  - Properties
  - Integrated vs Separate
    - Integrated Storage Structure
    - Separate Storage Structure
  - Clustered Integrated Index
  - Clustered Separate vs Unclustered Separate Index
    - Clustered Separate Index
    - Unclustered Separate Index
  - Sparse vs Dense Index
    - Dense Index
    - Sparse Index

A few notes

1. Magnetic disks and SSDs are block structured
2. Databases deal with records (usually smaller than blocks)
   Assumptions
3. No record is larger than a block
4. Each record is entirely contained in a single block
5. No record is party contained in one block and partly in another

File Structures

1. Heap files
2. Sorted files
3. Index files

Heap files

Overview

- Records (rows) are appended to the end of the file or in the next available space, therefore they are **unsorted** and have no particular order
- Deleted rows create gaps in file
  - file must be periodically compacted to recover space (defragmentation)

- A heap file is divided into pages (blocks of fixed size), and each page can hold multiple records

Performance

Assuming a file contains F pages

Equality Searching

- Access path is a scan
- Average: **F/2** transfers if the row exists
- **F** page transfers if row does not exist

Deleting

- Access path is scan
- Average: **F/2 + 1** of the row exists
  - The +1 is for deletion
- **F** page transfers of row does not exist

Range Search

Organization inefficient when a subset of rows is request

- **F** pages must be read

Inserting

Instantaneous

Sorted files

Rows are sorted based on some attribute

Advantages

1. Access path is **binary search**
2. Successive rows are in same (or successive) page(s) and **cache hits are likely**
3. By storing all pages on the same track, **seek time can be minimized**

Performance

Range / Equality Searching

- Costs log_2F to retrieve page containing first row

Inserting

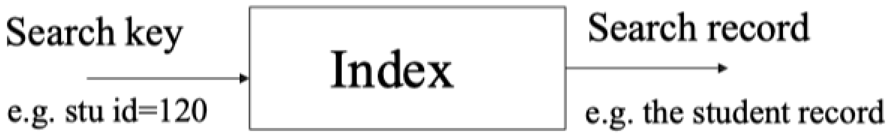
After the correct position for an insert has been determined, inserting rows require

- average **F/2 reads** and **F/2 writes** to account for shifting

Maintaining Sorted Order Solution

1. **fillfactor**: Leave empty space in each page
2. **overflow pages** (chains):
   - successive pages no longer stored contiguously
   - overflow chain not sorted, hence cost no lo longer **log_2F
   -

Index Files



Mechanism for efficiently locating row(s) without having to scan entire table. Do not confuse with candidate key

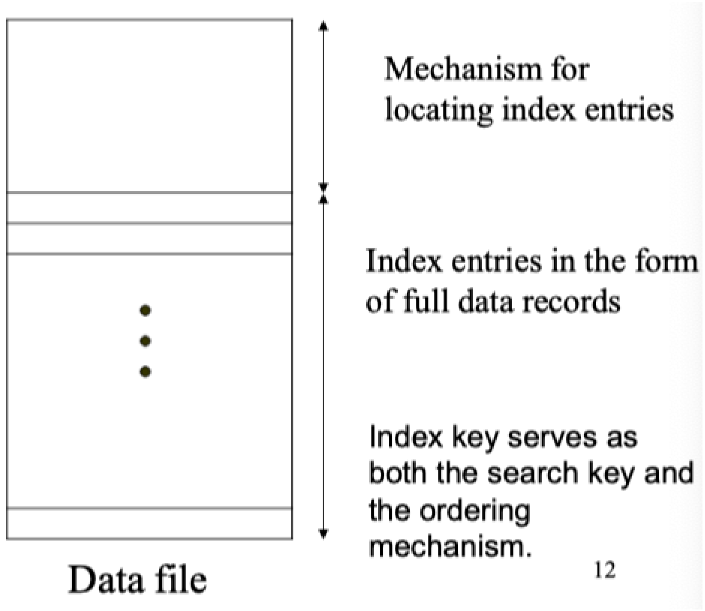
- Candidate key: set of attributes; guarantees uniqueness
- Search key: *sequence* of attributes; *does not guarantee* uniqueness

Properties

1. Full record vs key and a pointer
2. Integrated vs separate
3. Clustered vs unclustered
4. Dense vs sparse

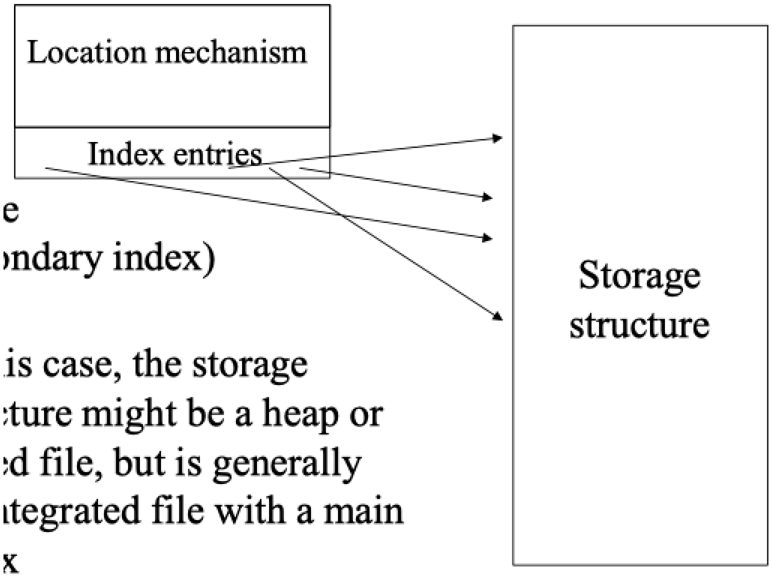
Integrated vs Separate

Integrated Storage Structure


Mechanism for locating index entries
Index entries in the form of full data records
Index key serves as both the search key and the ordering mechanism.
12
Data file

- Contains table and (main) index
- An integrated storage structure is when both the **index entries** and **data records** are stored **together in the same file**
- Index key serves as **both the search key** and **ordering mechanism**

Separate Storage Structure



e
ndary index)

is case, the storage ture might be a heap or d file, but is generally tegrated file with a main x

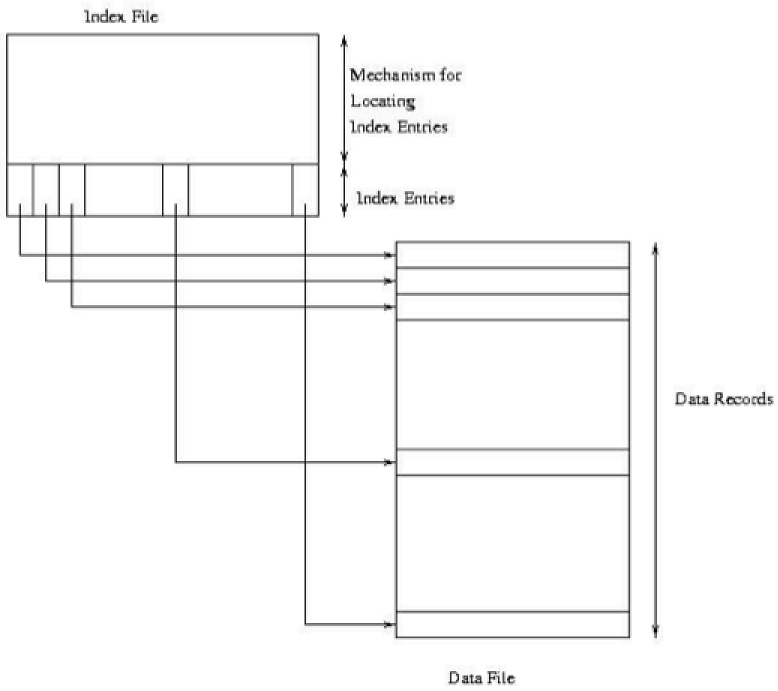
In this case, the storage might be a heap, or a sorted file. But is generally integrated with the main index.

Clustered Integrated Index

1. An integrated storage is **clustered by default**
2. Data is organized within the same structure as the index
3. Data records are sorted according to the values of the index key
4. Data is physically arranged in the storage based on the index key
5. Index entries in the form of full data records

Clustered Separate vs Unclustered Separate Index

Clustered Separate Index

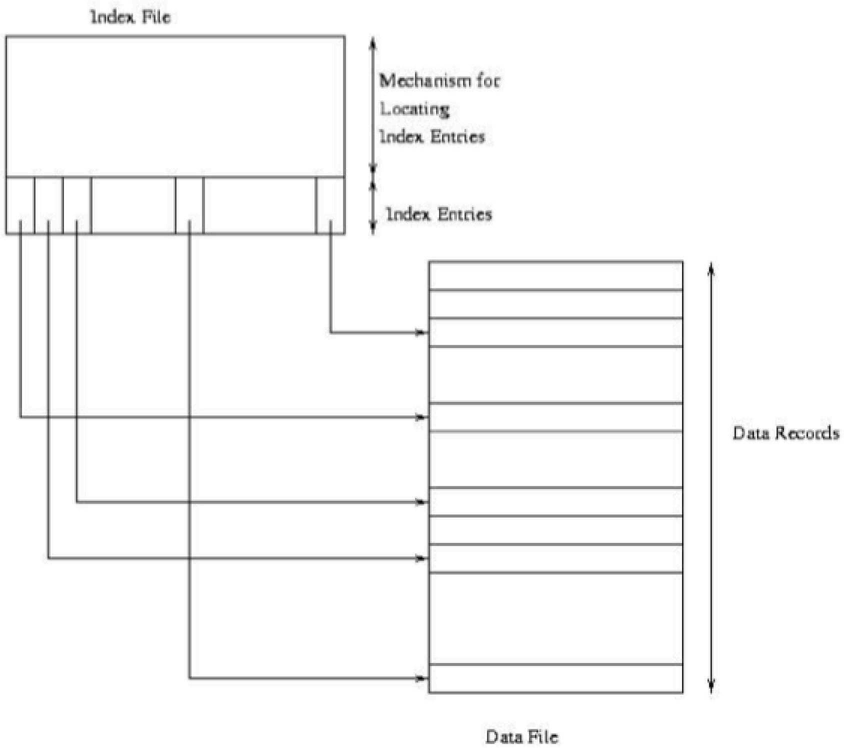


Data Storage Order: Data stored in the same order as the index key
Data Location: Data records are physically contiguous for the index key
Range Query Performance: High Efficiency (fewer page transfers)

Index Usage: Only clustered index per table

Best For: Range queries

Unclustered Separate Index



Data Storage Order: Data stored independently of index key order

Data Location: Data records are scattered across different pages

Range Query Performance: Low Efficiency (more page transfers)

Index Usage: Multiple unclustered indexes allowed

Best For: Individual record lookups

Sparse vs Dense Index

Dense Index

Has index for each data record. This means that

- **Unclustered index** must be dense, however, clustered index need not be dense

Sparse Index

Has index entry for each page of data file