# File Organization

Dr. R. Nageshwar Rao

Department of Mathematics
School of Advanced Sciences
Vellore Institute of Technology, Vellore.

October 1, 2024

## File Organization

File organization refers to the arrangement of data on storage devices. The method chosen can have a profound effect on the efficiency of various database operations.

**Common methods of file organization include:**

- Sequential (or Serial) File Organization
- Direct (or Hashed) File Organization
- Indexed File Organization

## Sequential (or Serial) File Organization

### Features of Sequential File Organization

**Ordered Records:** Records in a sequential file are stored in sequence, one after the other, based on a key field.

**Continuous Memory Allocation:** The records are stored in contiguous memory locations. The records are inserted at the end of the file, ensuring the sequence is maintained.

**No Direct Access:** To access a record, you have to traverse from the first record until you find the desired one.

## Sequential (or Serial) File Organization

### Advantages Sequential File Organization

**Simplicity:** The design and logic behind sequential file organization are straightforward.

**Efficient for Batch Processing:** Since records are stored in sequence, sequential processing (like batch updates) can be done efficiently.

**Less Overhead:** There's no need for complex algorithms or mechanisms to store records.

## Sequential (or Serial) File Organization

### Disadvantages Sequential File Organization

**Inefficient for Random Access:** If you need a specific record, you may have to go through many records before finding the desired one. This makes random access slow.

**Insertion and Deletion:** Inserting or deleting a record (other than at the end) can be time-consuming since you may need to shift records to maintain the order.

**Redundancy Issues:** There's a risk of redundancy if checks are not made before inserting records. For example, a record with the same key might get added twice if not checked.

## Sequential (or Serial) File Organization

### Practical Application:

Suppose you have a file of students ordered by their roll number:

| Roll No. | Name |
|---|---|
| 1 | Madhu |
| 2 | Naveen |
| 4 | Shivaji |
| 5 | Durga |

- In a sequential file, if you wanted to add a student with roll number 6, you would append them at the end.
- However, if you wanted to insert a student with a roll number 3 which is between 2 and 4, you would need to shift all subsequent records to maintain the sequence, which can be time-consuming.

## Direct (or Hashed) File Organization

- In hash file organization, a hash function is used to compute the address of a block (or bucket) where the record is stored.
- The value returned by the hash function using a record's key value is its address in the database.

**Features of Hash File Organization**

**Hash Function:** A hash function converts a record's key value into an address.

**Buckets:** A bucket typically stores one or more records. A hash function might map multiple keys to the same bucket.

**No Ordering of Records:** Records are not stored in any specific logical order.

## Direct (or Hashed) File Organization

### Advantages Hash File Organization

**Rapid Access:** If the hash function is efficient and there's minimal collision, the retrieval of a record is very quick.

**Uniform Distribution:** A good hash function will distribute records uniformly across all buckets.

**Efficient Search:** Searching becomes efficient as only a specific bucket needs to be searched rather than the entire file.

# Direct (or Hashed) File Organization

## Disadvantages Hash File Organization

**Collisions:** A collision occurs when two different keys hash to the same bucket. Handling collisions can be tricky and might affect access time.

**Dependency on Hash Function:** The efficiency depends on the hash function used. A bad hash function can lead to clustering and inefficient utilization of space.

**Dynamic Growth and Shrinking:** If the number of records grows or shrinks significantly, rehashing might be needed which is an expensive operation.

## Direct (or Hashed) File Organization

### Practical Application:

- Imagine a database that holds information about books.
- Each book has a unique ISBN number.
- A hash function takes an ISBN and returns an address.
- When you want to find a particular book's details, you hash the ISBN, which directs you to a particular bucket.
- If two books' ISBNs hash to the same value, you handle that collision, maybe by placing the new record in a linked list associated with that bucket.

## Indexed File Organization

- Indexed file organization is a method used to store and retrieve data in databases.
- It is designed to provide quick random access to records based on key values.
- In this organization, an index is created which helps in achieving faster search and access times.

**Features Indexed File Organization:**

**Primary Data File:** The actual database file where records are stored.

**Index:** An auxiliary file that contains key values and pointers to the corresponding records in the data file.

**Multi-level Index:** Sometimes, if the index becomes large, a secondary (or even tertiary) index can be created on the primary index to expedite searching further.

## Indexed File Organization

### Advantages Indexed File Organization:

**Quick Random Access:** Direct access to records is possible using the index.

**Flexible Searches:** ince an index provides a mechanism to jump directly to records, different types of search operations (like range queries) can be efficiently supported.

**Ordered Access:** If the primary file is ordered, then indexed file organization can support efficient sequential access too.

## Indexed File Organization

### Disadvantages Indexed File Organization:

**Overhead of Maintaining Index:** Every time a record is added, deleted, or updated, the index also needs to be updated. This can introduce overhead.

**Space Overhead:** Indexes consume additional storage space.

**Complexity:** Maintaining multiple levels of indexes can introduce complexity in terms of design and implementation.

## Indexed File Organization

### Practical Application:

Consider a database that holds information about students,

- where each student has a unique student ID.
- The main file would contain detailed records for each student.
- A separate index file would contain student IDs and pointers to the location of the detailed records in the main file.
- To fetch a specific student's details, first search the index, find the pointer and then use that pointer to fetch the record from the main file.