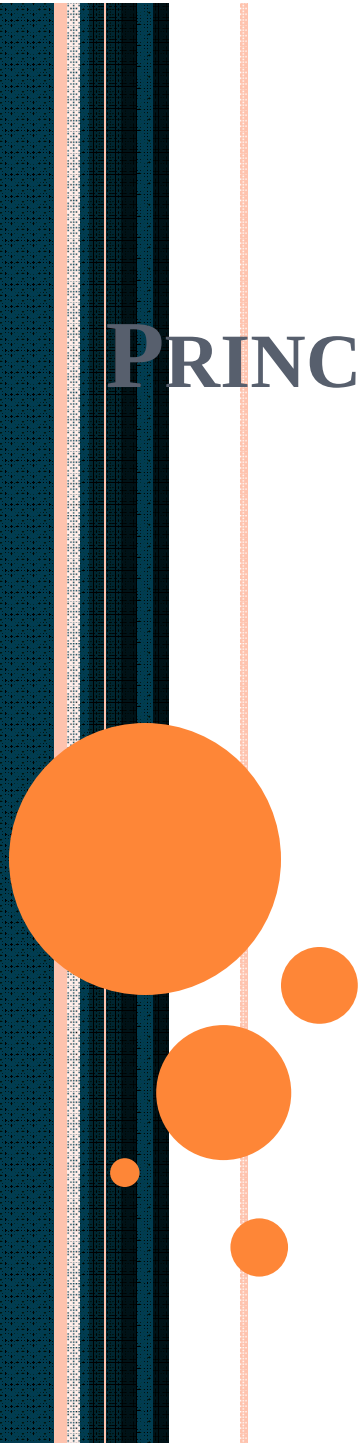
PRINCIPLES OF OPERATING SYSTEMS



Lecture- 19
File
Access methods

Attributes of a File

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring

- Information about files is kept in the directory structure, which is maintained on the disk.

File operations

A file is an abstract data type. It can be defined by operations:

- Create a file
- Write a file
- Read a file
- Reposition within file -file seek
- Delete a file
- Truncate a file
- Open(Fi): search the directory structure on disk for entry Fi, and move the content of entry to memory.
- Close(Fi):move the content of entry Fi in memory to directory structure on disk.

Types of File

File Type	Possible extension	Function
Executable	Exe,com,bin	Machine language program
Object	Obj, o	Compiled machine lang.,not linked
Source codec,	CC, p, java,	Source code in variouslanguages
Batch	Bat, sh	Commands to commandinterpreter
text	Txt, doc	Textual data, documents
Print, view	ps, dvi, gif	ASCII or binary file
archive	Arc, zip, tar	Group of files, sometimes compressed
Library	Lib, a	Libraries of routines

13-1 ACCESS METHODS

When we design a file, the important issue is how we will retrieve information (a specific record) from the file. Sometimes we need to process records one after another, whereas sometimes we need to access a specific record quickly without retrieving the preceding records. The access method determines how records can be retrieved: **sequentially** or **randomly**.

Sequential access

If we need to access a file sequentially—that is, one record after another, from beginning to end—we use a **sequential file structure**.

Random/Direct/Relative access

If we need to access a specific record without having to retrieve all records before it, we use a file structure that allows random access. Two file structures allow this: indexed files and hashed files. This taxonomy of file structures is shown in Figure 13.1.

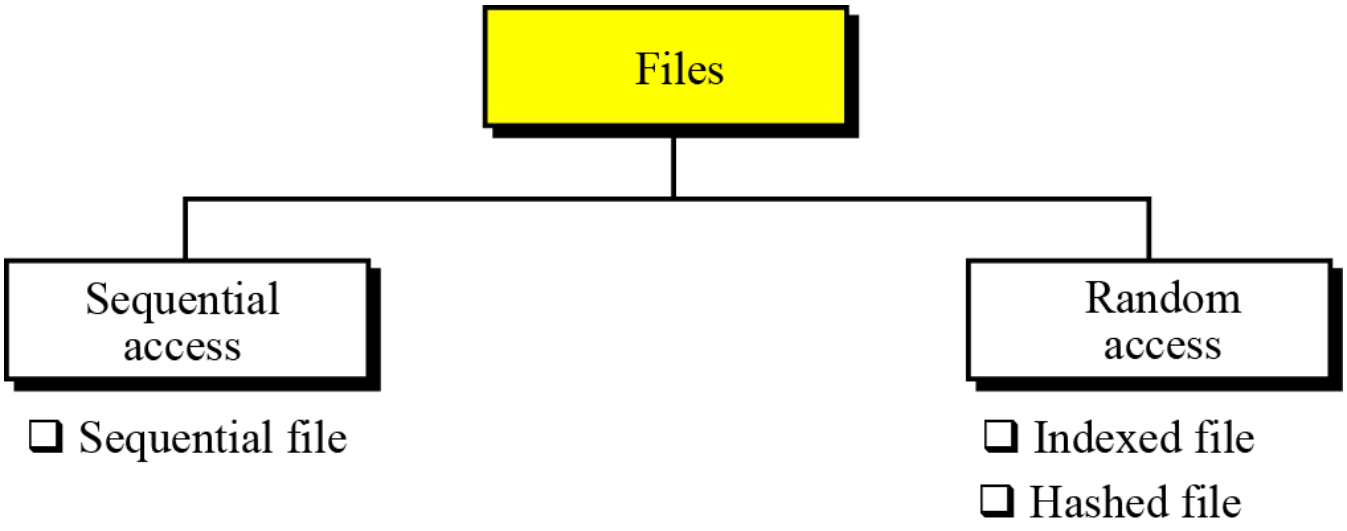


Figure 13.1 **A taxonomy of file structures**

13-2 SEQUENTIAL FILES

A sequential file is one in which records can only be accessed one after another from beginning to end. Figure 13.2 shows the layout of a sequential file. Records are stored one after another in auxiliary storage, such as tape or disk, and there is an EOF (end-of-file) marker after the last record. The operating system has no information about the record addresses, it only knows where the whole file is stored. The only thing known to the operating system is that the records are sequential.

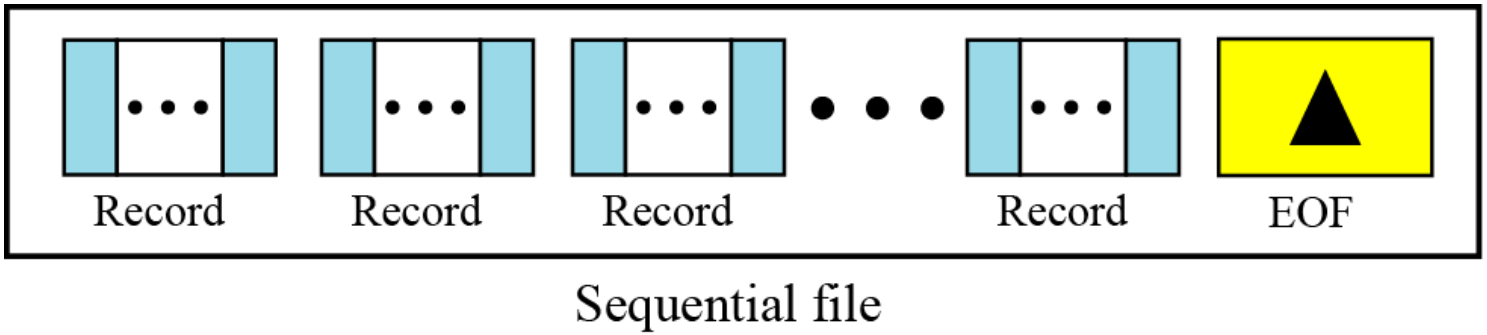


Figure 13.2 **A sequential file**

Algorithm 13.1 shows how records in a sequential file are processed.

Algorithm 13.1 Pseudocode for processing records in a sequential file

```
Algorithm: SequentialFileProcessing (file)
Purpose: Process all records in a sequential file
Pre: Given the beginning address of the file on the auxiliary storage
Post: None
Return: None
{
        while (Not EOF)
        {
                Read the next record from the auxiliary storage into memory
                Process the record
        }
}
```

Updating sequential files

Sequential files must be updated periodically to reflect changes in information. The updating process is very involved because all the records need to be checked and updated (if necessary) sequentially.

Files involved in updating

There are four files associated with an update program: the **new master file**, the **old master file**, the **transaction file** and the **error report file**. All these files are sorted based on key values. Figure 13.3 is a pictorial representation of a sequential file update.

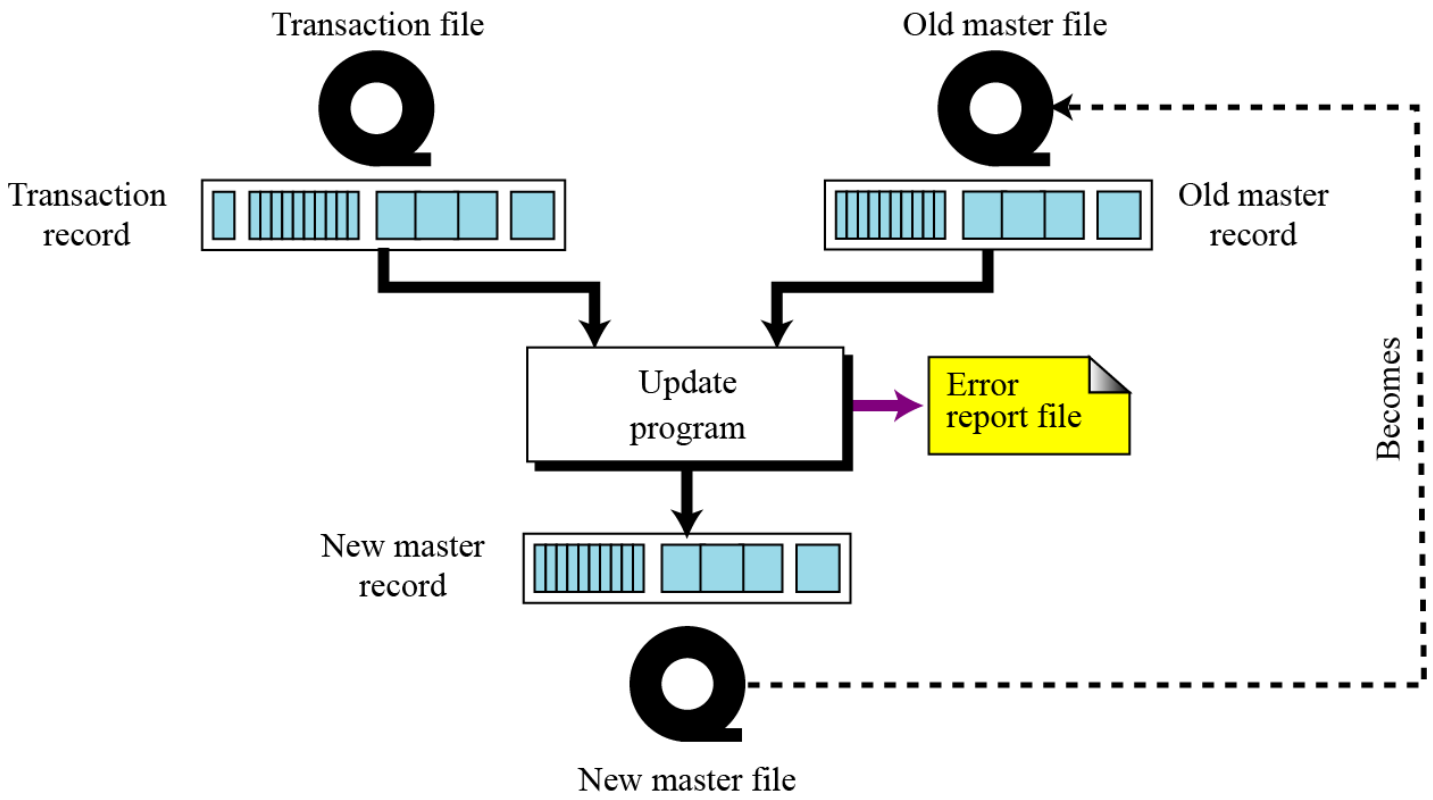


Figure 13.3 **Updating a sequential file**

Processing file updates

To make the updating process efficient, all files are sorted on the same key. This updating process is shown in Figure 13.4.

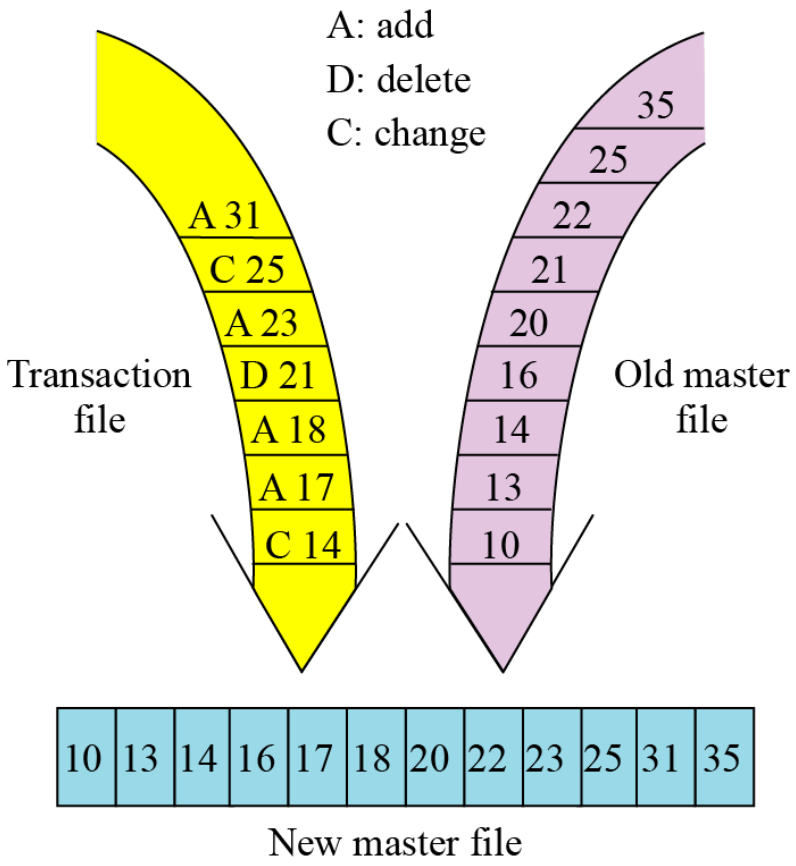


Figure 13.4 **Updating process**

13-2 INDEXED FILES

To access a record in a file randomly, we need to know the address of the record.

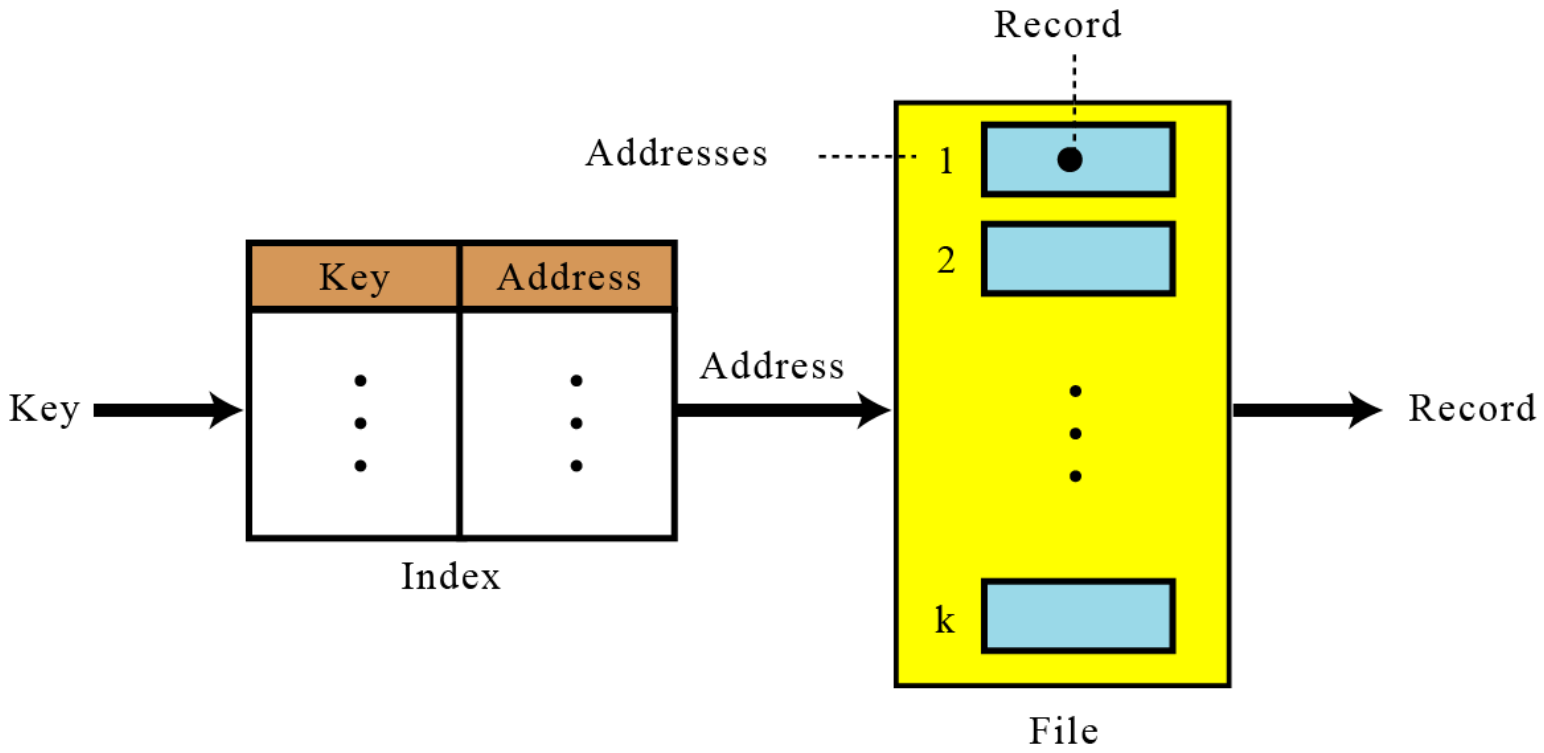


Figure 13.5 **Mapping in an indexed file**

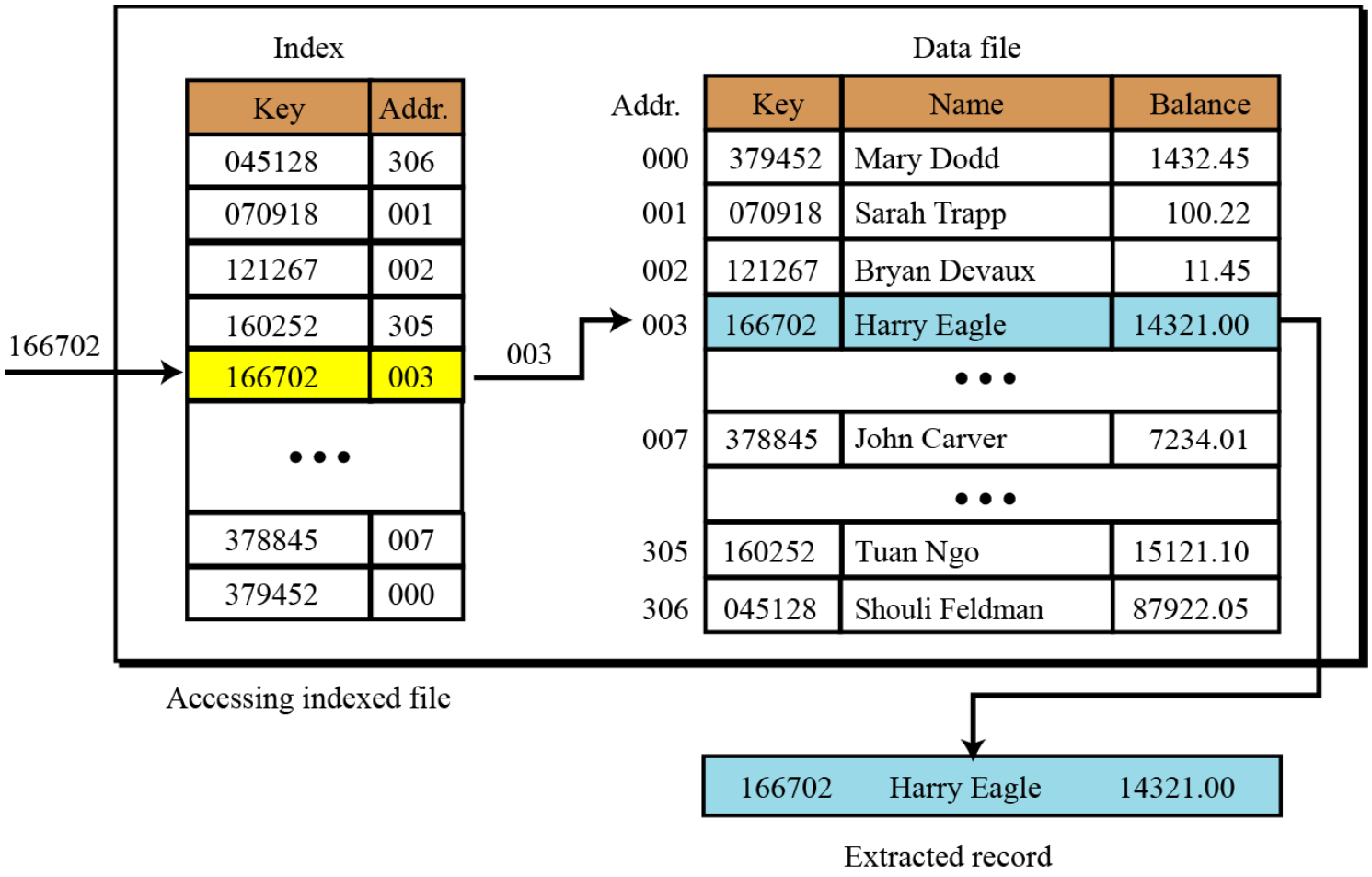


Figure 13.6 Logical view of an indexed file

12-2 HASHED FILES

A **hashed file** uses a mathematical function to accomplish this mapping. The user gives the key, the function maps the key to the address and passes it to the operating system, and the record is retrieved (Figure 13.7).

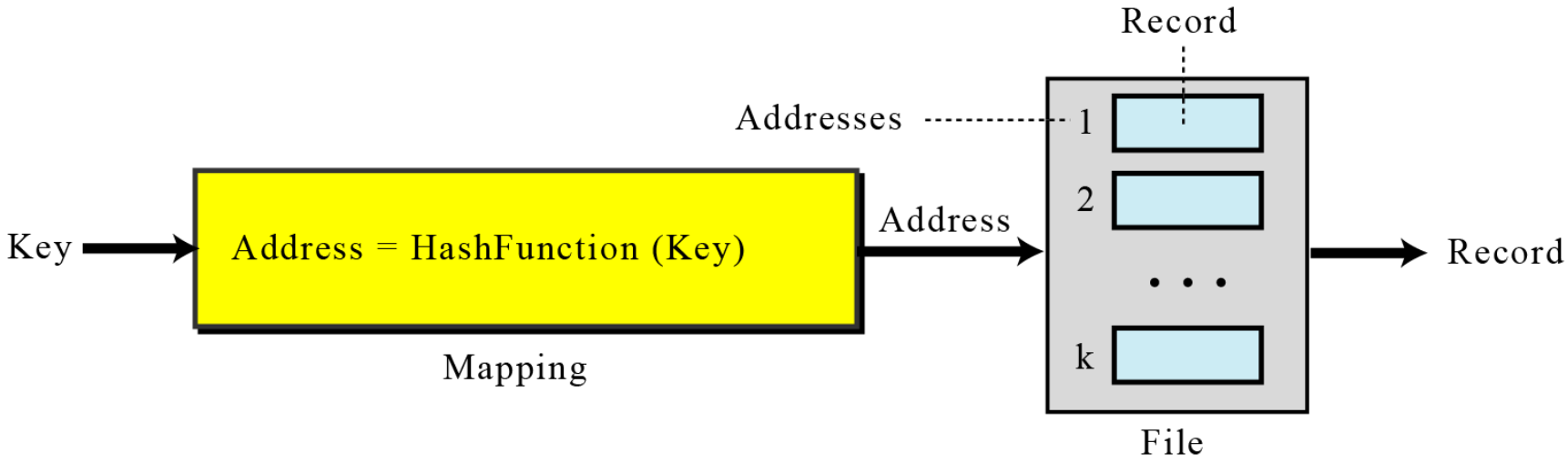


Figure 13.7 **Mapping in a hashed file**

Hashing methods

For key-address mapping, we can select one of several hashing methods. We discuss a few of them here.

Direct hashing

In direct hashing, the key is the data file address without any algorithmic manipulation. The file must therefore contain a record for every possible key. Although situations suitable for direct hashing are limited, it can be very powerful, because it guarantees that there are no synonyms or collisions (discussed later in this chapter), as with other methods.

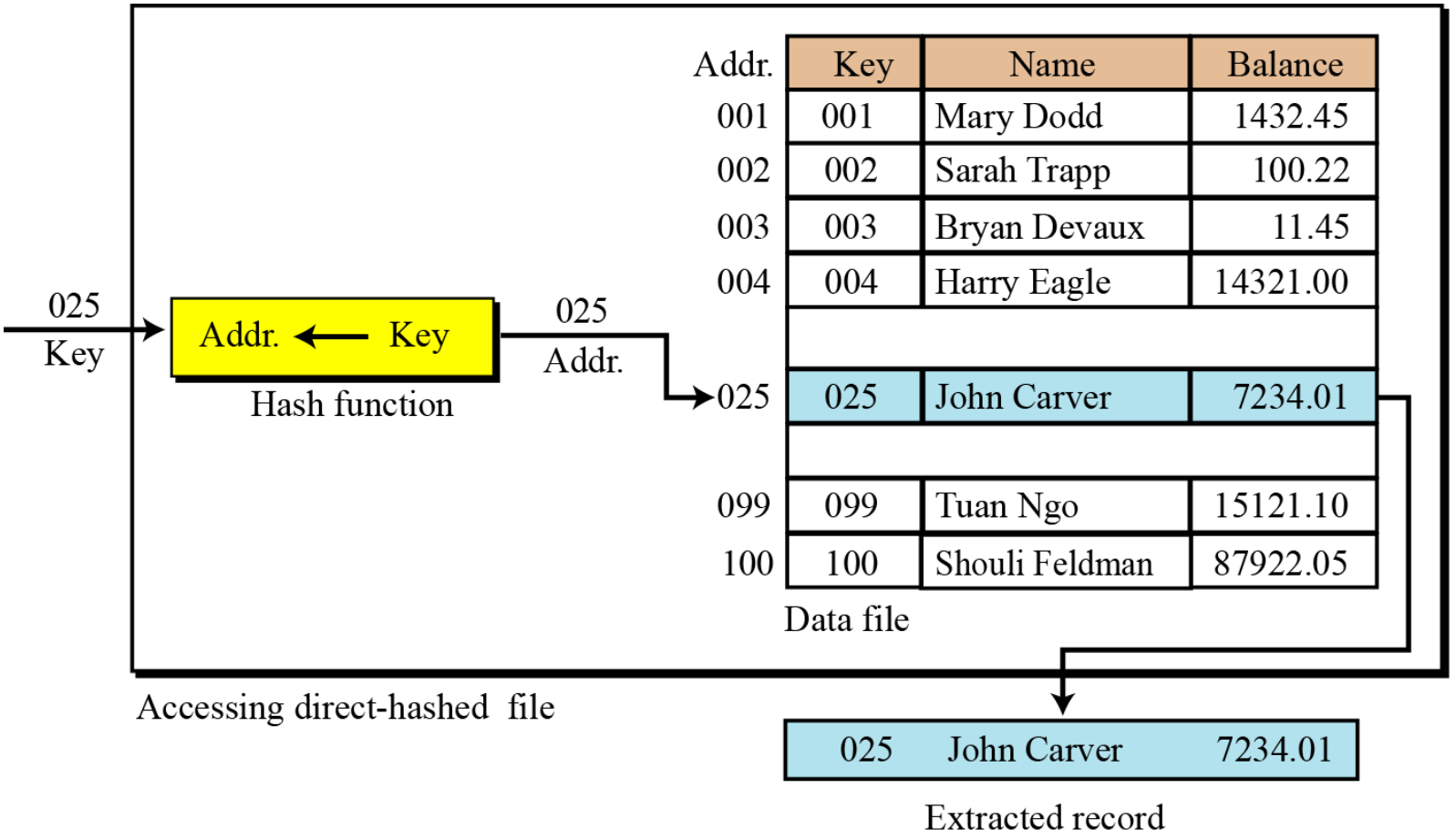


Figure 13.8 Direct hashing

Modulo division hashing

Also known as division remainder hashing, the modulo division method divides the key by the file size and uses the remainder plus 1 for the address. This gives the simple hashing algorithm that follows, where list_size is the number of elements in the file. The reason for adding a 1 to the mod operation result is that our list starts with 1 instead of 0.

address = key **mod** *list_size* + 1

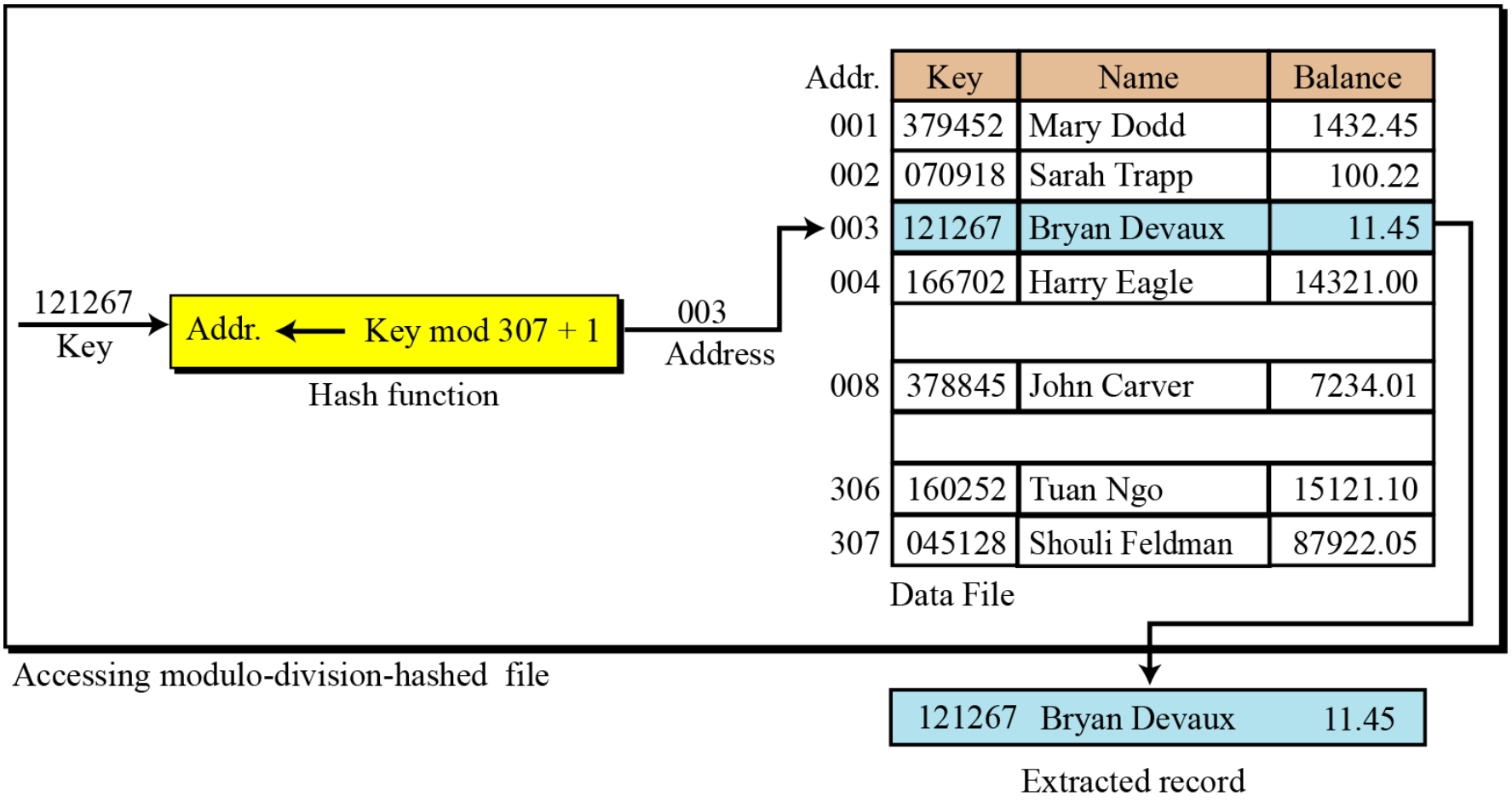


Figure 13.9 **Modulo division**

Assignment

- Q. What are the attributes of a file and explain file access methods.