File Support for Database Systems

- Modern DBMS have their own in-built File Manager [File System, Storage Engine] – as opposed to using the underlying OS-provided file system
- Sole interaction with the native file system is to request allocation [or deallocation] of large quotas of disk storage
- DBMS Storage Engine is tuned to cater specifically for data file processing [e.g. SELECT/UPDATE … of table data, concurrent access to file/table data, data conflict resolution, indexing, etc] – whereas OS-FS more general-purpose in nature

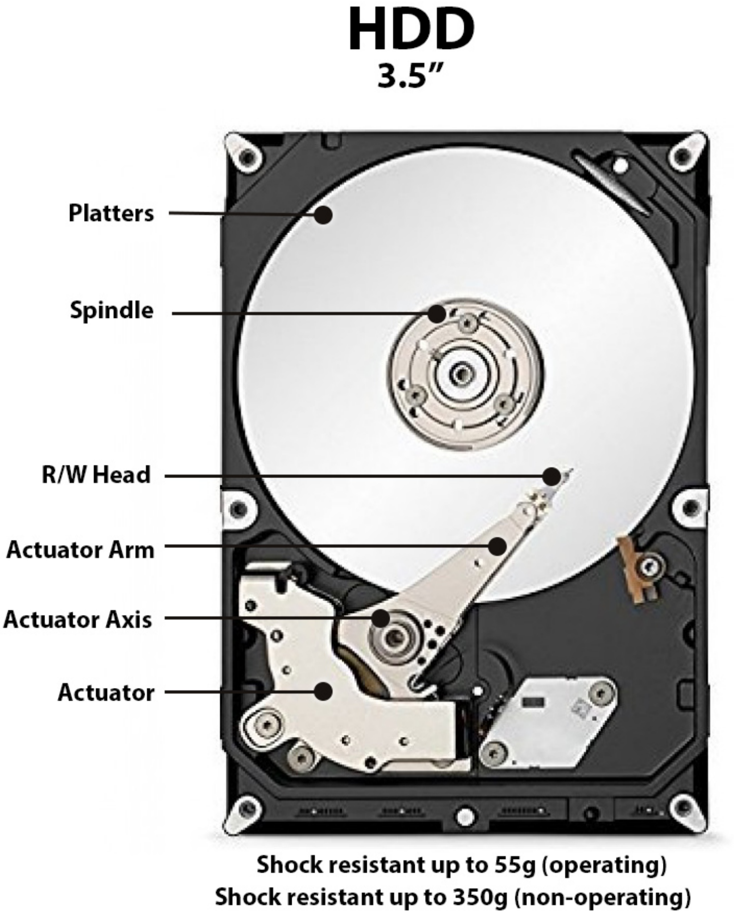
Example [of File Processing Program]

```
program staff_lookup
declare
     EmpRec struct
       {  lname:   char (12);
          minit:    char;
          fname:  char (12);
          .
          .
          dno:  char (5)   }

     employee.db:  file of EmpRec;

     Search_lname:  char (12);

begin
     display (“Enter Search Value (Last Name): ”);
     accept (Search_lname);
     open (input, employee.db);
     read (employee.db, EmpRec);
     while not eof (employee.db) do
          {
          if EmpRec.lname = Search_lname) then
               display (EmpRec.fname, Emp.Rec.lname,
                                              EmpRec.dno);
           read (employee.db, EmpRec)
          }
     close employee.db;
     end.
```

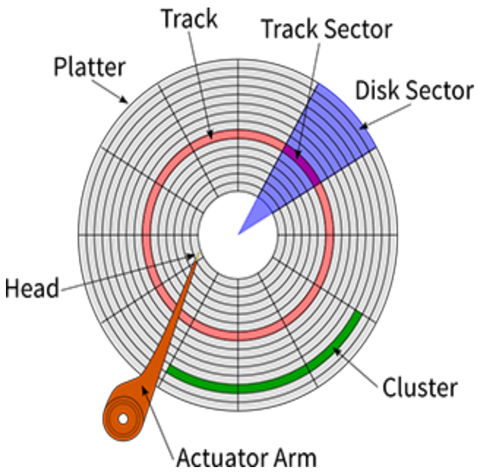
```
SELECT fname, lname, dno

FROM employee

WHERE Lname = ‘Murphy’
```

Hard Disk Characteristics

- A HD consists of:
  - multiple platters [usually 2 disks, 4 surfaces] that rotate at constant speed
  - read/write heads – laser reader/writer that can read data from, or write data to, disk surface; RW heads move in unison


HDD
3.5"
Platters
Spindle
R/W Head
Actuator Arm
Actuator Axis
Actuator
Shock resistant up to 55g (operating)
Shock resistant up to 350g (non-operating)

- through formatting, the HD is divided into tracks & sectors [and cylinders (concentric tracks), clusters], which are used as storage bins for records of a data file
- each sector has a fixed storage capacity [e.g. 512 bytes, or 0.5K] and is addressable [Surface No, Track No, Sector No]
- To read data from a specific sector, the RW head must be positioned over it: there is a seek delay (~ 4-9ms) and a rotational/latency delay (~ 2ms)
- For efficiency of processing, it is best if s file is stored on a single, or adjacent, cylinders of disk – so as to reduce delays during access; alternative is to risk disk thrashing during access



File System / File Manager / Storage Manager

- this is software that manages hard disk, allowing users/programs to store and retrieve files on disk
- it maintains a *file directory* data structure [partly visible to a user, partly hidden] for file/disk administration
- it supports different *file organisations:* serial, sequential and indexed sequential

File Allocation [Record-Sector Mapping]

- records could be packed into sectors [e.g. with 110-byte records, could fit 4 records per sector, with 80 bytes unused at end of sector]
- could daisy-chain sectors pertaining to file by placing "next sector address" [or "end-of-file"] at end of each sector; then place address of first file sector in file directory

insert diagram here

- this approach would be limited: only serial files possible; also, reorganizing of file on disk very difficult

- to solve reorganization issues, most file systems use a *File Master Sector* pointed to from directory, and which then points to each data sector of file; for large files, one such sector may be insufficient: we could extend it to a tree-like structure

insert diagram here

- to support sequential files, a file system might keep the data sectors only partially filled, and split a sector when an insertion causes it to overflow – on average the data sectors will be 66% full

insert diagram here

- to support indexed sequential files, a file system also keep the lowest key value of any data sector in the master sector(s) along with the data sector pointer; the master sector(s) are now called *index sector(s)* – and they form a data structure known as a B+-tree; note that this approach supports a *sparse index* (because only one index entry per data sector is recorded, and not one per record)

insert diagram here