Chapter 16

Disk Storage, File Structure, and Hashing

7 March 2017

Chris Sexton

Disk Storage

Primary and Secondary Storage

* Primary storage: media directly accessible by the CPU

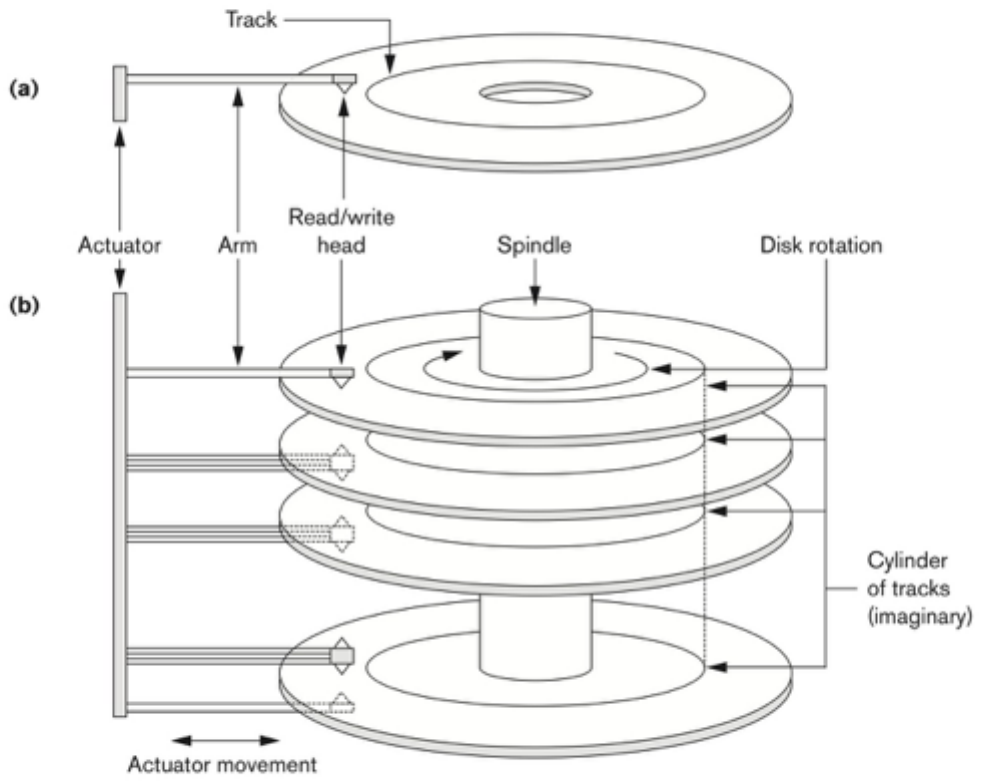
- Limited size
- Fast access

* Secondary/Tertiary Storage: Spinning disks, SSDs, Tapes, other media

- Much slower
- can be viewed as "infinite" storage

Spinning Disks

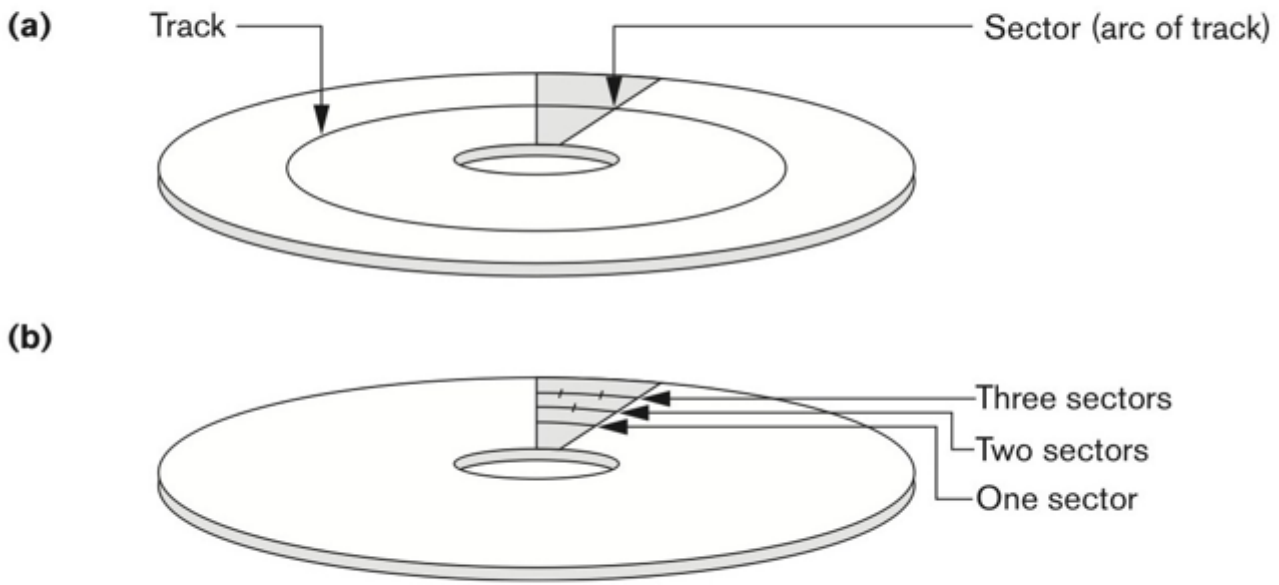
Figure 16.1 (a) A single-sided disk with read/write hardware. (b) A disk pack with read/write hardware.



Spinning Disks

man fdisk (https://linux.die.net/man/8/fdisk)

Figure 16.2 Different sector organizations on disk. (a) Sectors subtending a fixed angle. (b) Sectors maintaining a uniform recording density.



Why does my disk matter?

- Seek time: Time required to move the disk arm/head into position
- Rotational delay (latency): Time required for disk to spin the requested block under the head
- Block transfer time: Time needed to transfer data from the head to the disk controller

Why does my disk matter?

Table 16.2 (b) Internal Drive Characteristics of 300 GB–900 GB Seagate Drives

	ST900MM0006 ST900MM0026 ST900MM0046 ST900MM0036	ST600MM0006 ST600MM0026 ST600MM0046	ST450MM0006 ST450MM0026 ST450MM0046	ST300MM0006 ST300MM0026 ST300MM0046	
Drive capacity	900	600	450	300	GB (formatted, rounded off value)
Read/write data heads	6	4	3	2	
Bytes per track	997.9	997.9	997.9	997.9	KBytes (avg, rounded off values)
Bytes per surface	151,674	151,674	151,674	151,674	MB (unformatted, rounded off value)
Tracks per surface (total)	152	152	152	152	KTracks (user accessible)
Tracks per inch	279	279	279	279	KTPI (average)
Peak bits per inch	1925	1925	1925	1925	KBPI
Areal density	538	538	538	538	Gb/in2
Disk rotation speed	10K	10K	10K	10K	rpm
Avg rotational latency	2.9	2.9	2.9	2.9	ms

Disk shopping (Magnetic spinning disks)

- Low price/GB
- High failure rate spread over many disks
- Very slow compared to SSD

Disk shopping (SSD)

- Becoming more reasonably priced
- Possibly lower failure rates

Debunking SSD Myths (http://www.networkworld.com/article/2873551/data-center/debunking-ssd-myths.html)

Disk Benchmarks

Table 16.1 Types of Storage with Capacity, Access Time, Max Bandwidth (Transfer Speed), and Commodity Cost

Type	Capacity*	Access Time	Max Bandwidth	Commodity Prices (2014)**
Main Memory- RAM	4GB–1TB	30ns	35GB/sec	$100–$20K
Flash Memory- SSD	64 GB–1TB	50µs	750MB/sec	$50–$600
Flash Memory- USB stick	4GB–512GB	100µs	50MB/sec	$2–$200
Magnetic Disk	400 GB–8TB	10ms	200MB/sec	$70–$500
Optical Storage	50GB–100GB	180ms	72MB/sec	$100
Magnetic Tape	2.5TB–8.5TB	10s–80s	40–250MB/sec	$2.5K–$30K
Tape jukebox	25TB–2,100,000TB	10s–80s	250MB/sec–1.2PB/sec	$3K–$1M+

*Capacities are based on commercially available popular units in 2014.

**Costs are based on commodity online marketplaces.

Disk shopping

Let's go shopping and find current prices and capabilities.

Amazon internal spinning disks (https://www.amazon.com/b?ie=UTF8&node=1254762011)

Amazon solid state drives (https://www.amazon.com/b?ie=UTF8&node=1292116011)

File structure

Records

- Datatypes or record types

```
struct employee {
    char name[30];
    char ssn[9];
    double salary;
    int job_code;
    char department[20];
}
```

- char: 1 byte
- int: 2-4 bytes
- double: 8 bytes

Files of Fixed-length Records

- Every record in a file is exactly the same size
- All records are of the same type
- no repeating fields
- no optional fields

Files of Variable-length Records

```
struct employee2 {
    char name[30];
    char ssn[9];
    double salary;
    int job_code;
    dept *department;
}
struct list_t {
    void *value;
    struct list_t *next;
}
struct department {
    char name[30];
    char location[30];
    employee2 manager;
    list_t *employees;
}
```

- Allow optional fields
- Repetition (lists) may be used in a record

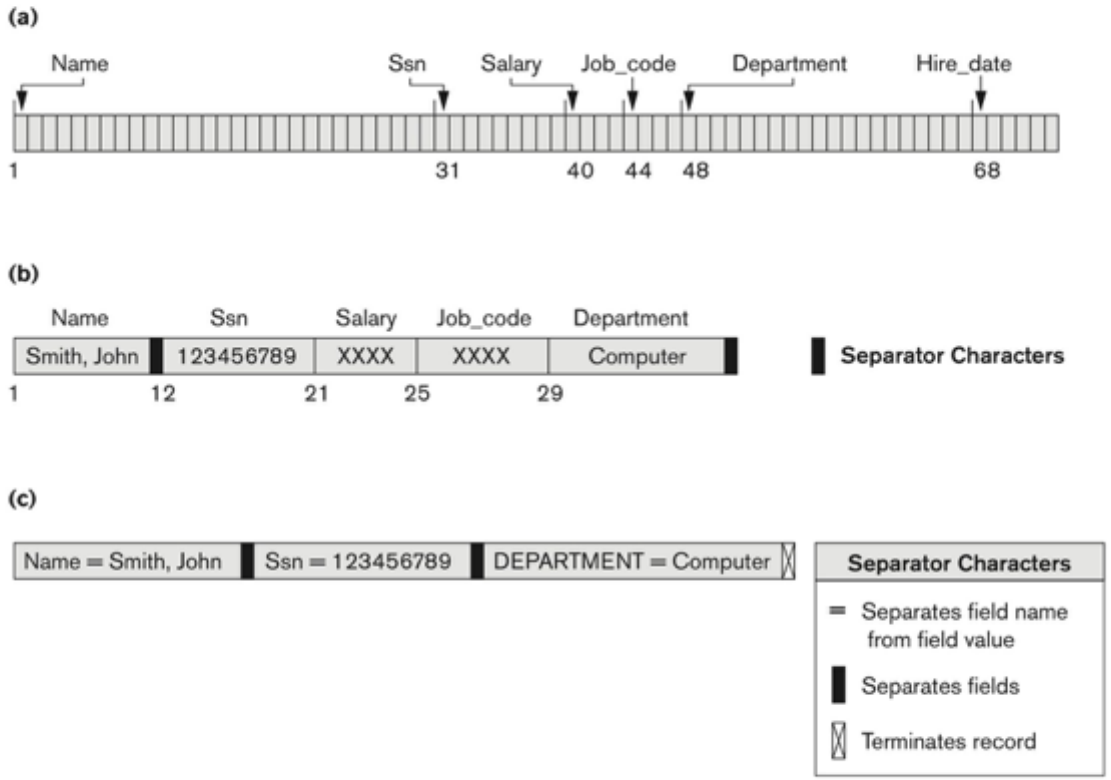
Files of Variable-length Records

```
struct student{
    char name[30];
    char id[9];
    list_t *classes;
}
struct instructor{
    char name[30];
    char id[9];
    double salary;
}
union studentOrInstructor {
    student s;
    instructor i;
}
studentOrInstructor si;
si.s.name;
si.i.name;
```

- Records may be of different types (Type UNION)

Fixed vs Variable-length

Figure 16.5 Three record storage formats. (a) A fixed-length record with six fields and size of 71 bytes. (b) A record with two variable-length fields and three fixed-length fields. (c) A variable-field record with three types of separator characters.



File Operations

- open
- reset
- find
- read
- findNext
- delete
- modify
- insert
- close

File Operations (open)

- Prepares the file for reading or writing.
- Allocates appropriate buffers (typically at least two) to hold file blocks from disk, and retrieves the file header.
- Sets the file pointer to the beginning of the file.

File Operations (reset)

- Sets the file pointer of an open file to the beginning of the file

File Operations (find)

- Searches for the first record that matches a condition
- Transfers that block into main memory
- Sets the file pointer to the record's buffer

File Operations (read)

- Copies the current record from the buffer to the program
- Advances the record pointer to the next record

File Operations (findNext)

- Searches for the next record that matches a condition
- Transfers that block into main memory
- Sets the file pointer to the record's buffer
- VS find, this does not have to be the first record matching

File Operations (delete)

- Removes the current record and updates the disk to match

File Operations (modify)

- Change some fields of a record
- Write the changes to disk

File Operations (insert)

- Insert a new record in a file
- Transfers block on disk to main memory
- Writes the changes to the record
- Writes the buffer to disk

File Operations (close)

- Complete file access
- Release all buffers
- Perform cleanup actions

Hashing

Files of Ordered Records

- physically ordered on disk by an **ordering field**
- no sorting required for retrieval
- no sorting required for find operations
- find operation may utilize binary search for log2 access

Files of Ordered Records

Figure 16.7 Some blocks of an ordered (sequential) file of EMPLOYEE records with Name as the ordering key field.

Hash Tables

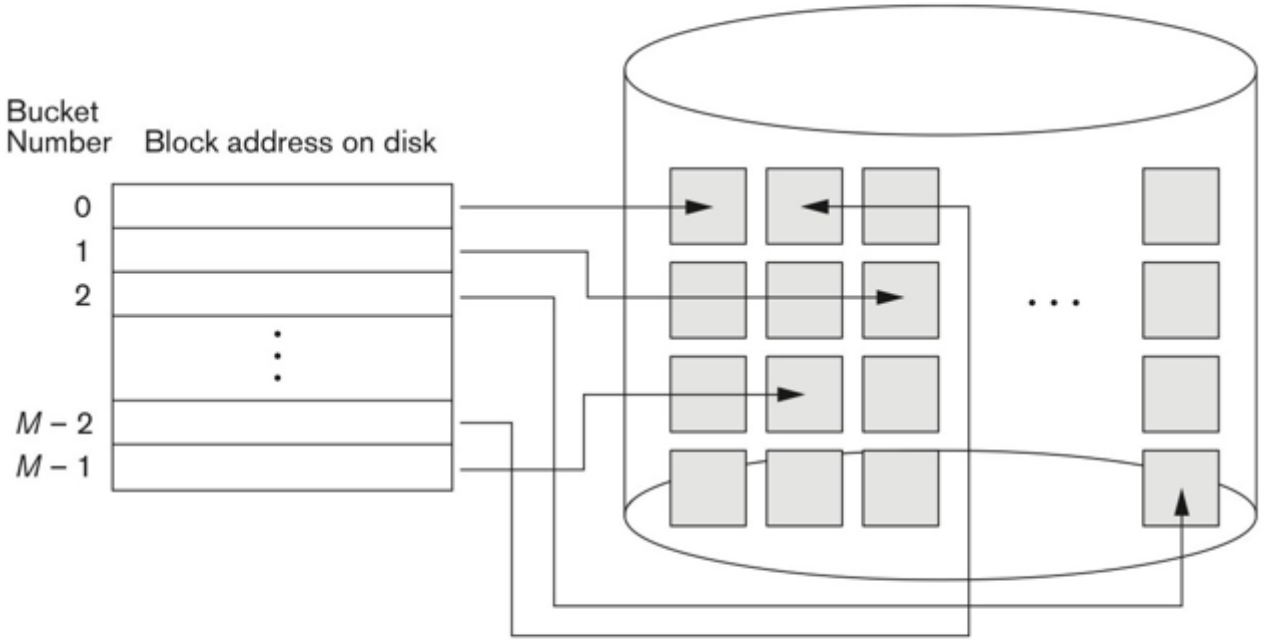
- **hash file**: A file organized to provide fast, unordered access to particular fields
- **hash field/key**: The unique key field of the data used as identification
- **hash function**: Any function *f(x)->y* that randomizes a piece of data, *x* to produce an address, *y*
- **hash table**: An array of records with *M* slots and a hash function that transforms a key field into an integer between 0 and *M-1*

Amortized Constant Time!!! (http://www.cs.cornell.edu/courses/cs312/2008sp/lectures/lec20.html)

- O(1) insert
- O(1) remove
- O(1) contains

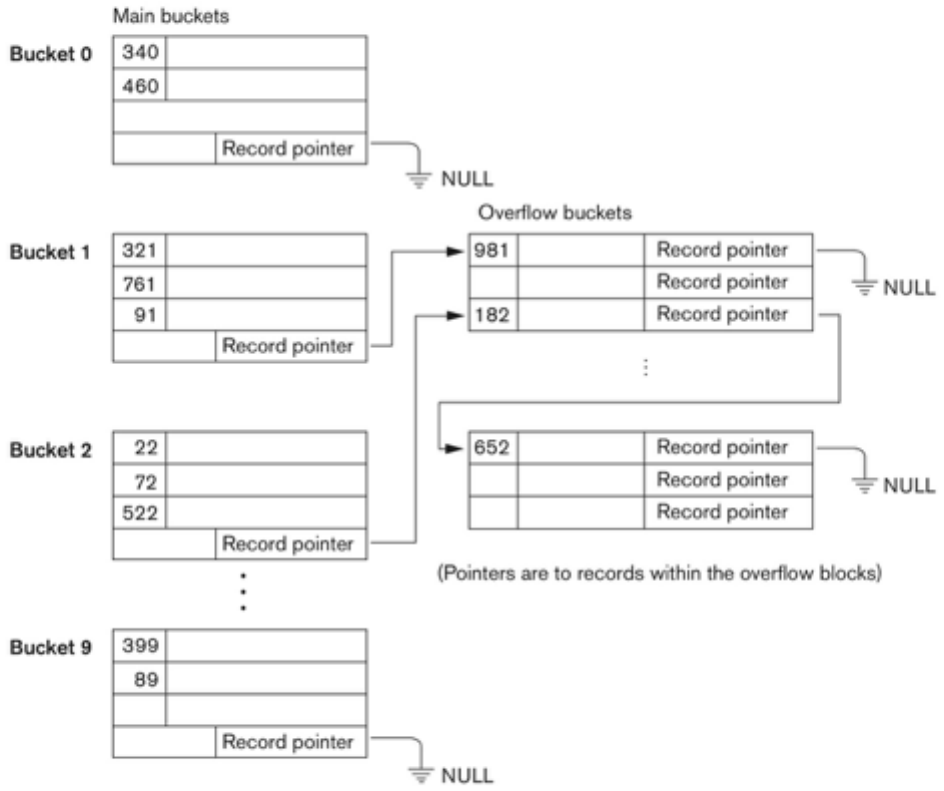
External Hashing

Figure 16.9 Matching bucket numbers to disk block addresses.



Extendable Hashing

Figure 16.10 Handling overflow for buckets by chaining.



Example (17.31)

- A PARTS file with Part# as the hash key includes records with the following Part# values: 2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, 6975, 4981, and 9208.
- The file uses eight buckets, numbered 0 to 7. Each bucket is one disk block and holds two records.
- Load these records into the file in the given order, using the hash function h(K) = K mod 8.

Example (17.31)

- Part# values: 2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, 6975, 4981, and 9208; buckets 0-7 with 2 records each.
- h(K) = K mod 8

* Table

- 0: ???, ???
- 1: ???, ???
- 2: ???, ???
- 3: ???, ???
- 4: ???, ???
- 5: ???, ???
- 6: ???, ???
- 7: ???, ???

Example (17.31)

h(2369) = 2369 % 8 = 1

* Table

- 0: ???, ???
- 1: 2369, ???
- 2: ???, ???
- 3: ???, ???
- 4: ???, ???
- 5: ???, ???
- 6: ???, ???
- 7: ???, ???

Example (17.31)

h(3760) = 3760 % 8 = 0

* Table

- 0: 3760, ???
- 1: 2369, ???
- 2: ???, ???
- 3: ???, ???
- 4: ???, ???
- 5: ???, ???
- 6: ???, ???
- 7: ???, ???

Example (17.31)

- Part# values: 2369, 3760, 4692, 4871, 5659, 1821, 1074, 7115, 1620, 2428, 3943, 4750, 6975, 4981, and 9208; buckets 0-7 with 2 records each.
- h(K) = K mod 8

* Table

- 0: 3760, 9208
- 1: 2369, ???
- 2: 1074, ???
- 3: 5659, 7115
- 4: 4692, 1620 -> (overflow) 2428, ???
- 5: 1821, 4981
- 6: 4750, ???
- 7: 4871, 3943 -> (overflow) 6975, ???

Thank you

Chris Sexton