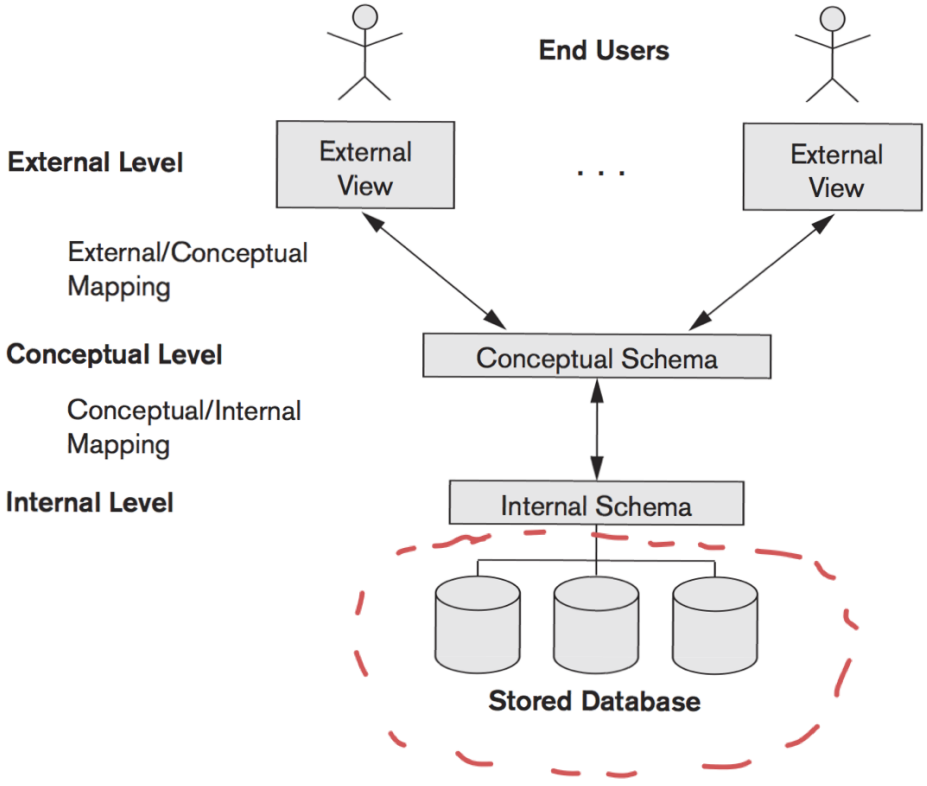
COS221 - L24 - File Organisation (Part 1)

Linda Marshall

3 May 2023

Figure 2.2
The three-schema architecture.



Terms you should be familiar with:

- primary storage: cache memory, main memory
  flash memory!
- secondary storage: mass storage, storage capacity
- tertiary storage: backup storage

Physical Level

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.

Physical Level

- Programs are typically in main memory and execute there.
- The database resides on secondary storage.
- Large parts of a database can be loaded into main memory (referred to as *main memory databases*), these are useful for real-time applications were response time is critical.

Secondary Storage Devices

- Database data is vast and must persist over long periods of time. This is referred to as **persistent data** in contrast to **transient data** which persists for a limited time, typically during program execution.
- Most data is stored permanently on secondary storage, for the following reasons:
  - data is too large to store in main memory;
  - storage must be nonvolatile; and
  - cost of secondary storage is way less than primary storage.
- DBMS's usually present the DBA with alternative physical database storage options during DBMS installation. The DBA must choose the best options for the application requirements.

Secondary Storage Devices

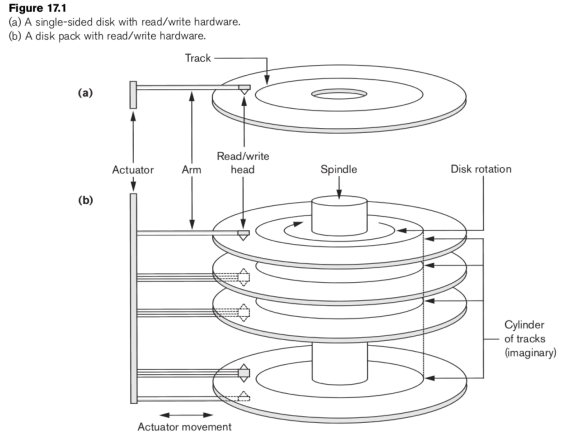
- The data stored on secondary storage needs to be organised to enable access to it.
- The data is organised as records (relating to the tuples).
- The records are stored in files on the disk.
- This is referred to **primary file organisation**.

Secondary Storage Devices

- There are several **primary file organisations** which determine how files are physically stored on the disk and therefore how the records are accessed.
  - **Heap** (unordered) file - records are appended at the end of the file
  - **Sorted** (sequential) file - keeps records sorted by a particular field (the sort key)
  - **Hashed** file - uses a hash function on a particular field (the hash key) to place records
  - **B-trees** - used to place records in a tree structure
- A **secondary organisation** (or auxiliary access structure) allows access to files based on alternate fields, not used by those by the primary file organisation. These are mostly indexes.

Secondary Storage Devices

- Magnetic disks



- Disk controller controls the disk and communicates with the computer system
- What can cause delay? Rotation of disk, block transfer time (consecutive vs butterfly)
- Random access

Secondary Storage Devices

- Magnetic tapes
  - Sequential device
  - To write the n^{th} block, $n-1$ blocks need to be scanned
  - Mostly used for backups

Secondary Storage Devices - Buffering of blocks

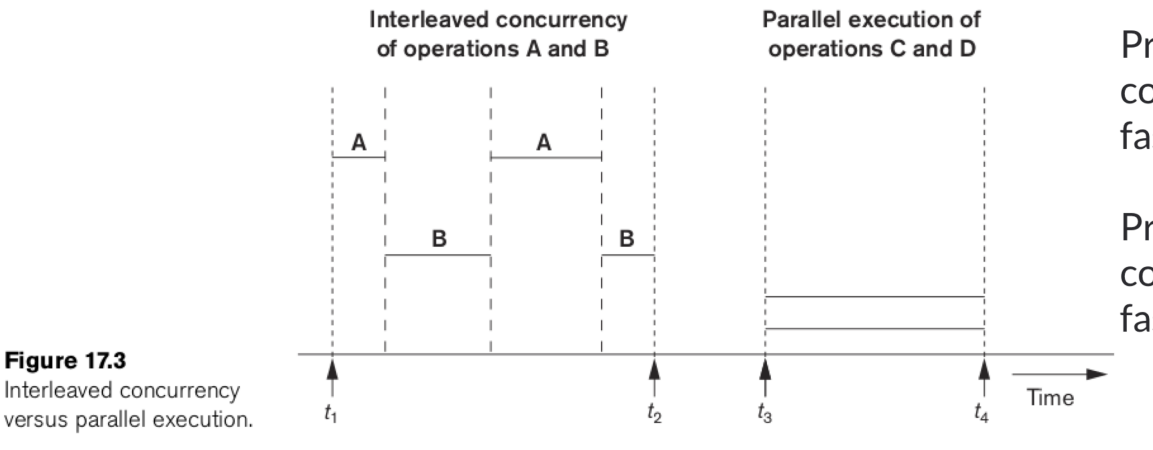


Figure 17.3
Interleaved concurrency versus parallel execution.

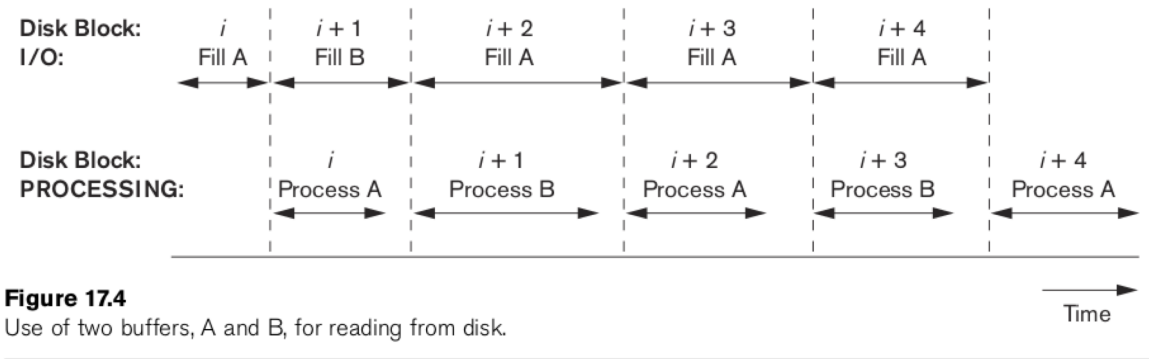


Figure 17.4
Use of two buffers, A and B, for reading from disk.

Process A and B are running concurrently in an interleaved fashion. Single CPU.

Process C and D are running concurrently in a parallel fashion. More than 1 CPU.

Double buffering used to read (or write) a continuous stream of blocks into memory (or from memory onto disk) - while one block is being processed, the next block is read in.

Limits seek time and rotational delay.

Placing files on disk

Consider both the entities (records and files) to be placed on the disk as well as how they are placed.

- A collection of field names and their corresponding data types is referred to as a record type (or record format). An example of a record type for EMPLOYEE, given in C, is:

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

- A large unstructured object (BLOB) is usually stored separately from its record in a pool of disk blocks. A pointer to the BLOB is stored in the record.

Placing files on disk

File and record entities

- A file is a sequence of records. Files in which the records are all the same size are said to be made up of *fixed-length* records. Where the record sizes vary, the file is made up of *variable-length* records.
- Variable-length records exist, because:
  - some fields in the record may be of varying length
  - fields in the record may have multiple values, referred to as repeating field. A group of values belonging to a repeating field are referred to as a repeating group.
  - one or more fields of the record are optional.
  - the record types in the file are different - referred to as a mixed file.

Placing files on disk

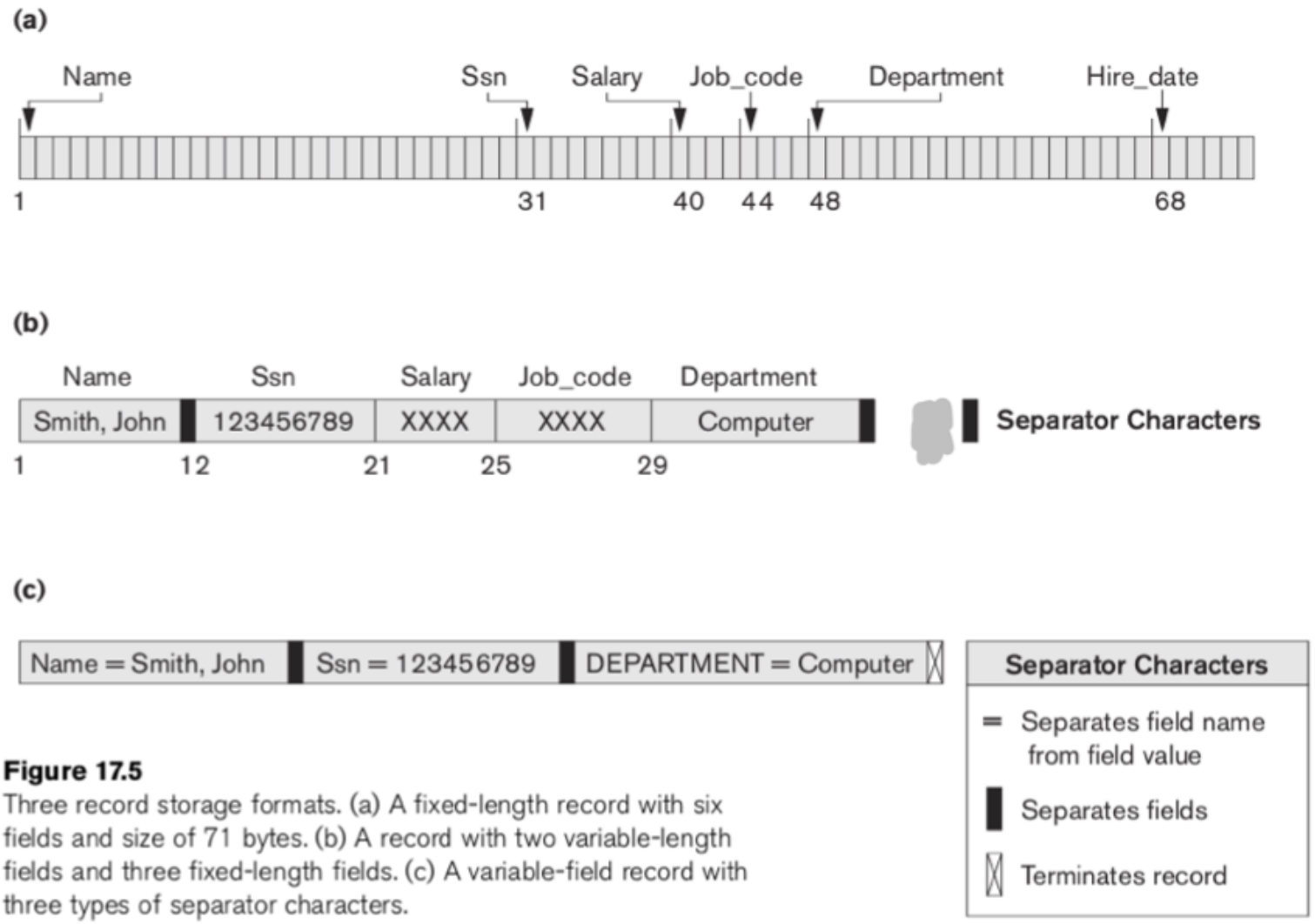


Figure 17.5
Three record storage formats. (a) A fixed-length record with six fields and size of 71 bytes. (b) A record with two variable-length fields and three fixed-length fields. (c) A variable-field record with three types of separator characters.

Placing files on disk

- It is conceivable to store all variable length fields using a maximum number possible number of fixed bytes and including a way of showing that a value has not been included by a NULL. This is a waste of storage space.
- To determine where a variable length field ends, a special separator character is introduced.
- Formatting a file of records with optional fields can be done in the following ways:
  - if the number of fields that appear in the record are small and the number of fields defined for the record large, then each record's field can be defined by a `<field-name, field-value>` pair.
    Three separating characters are used:
    - separation of field and value pairs (=)
    - separation of fields (black block)
    - separation of records (white block with cross)
  - When considering repeating fields, a separator character is required as before for the fields and one is required to separate individual values in the field.

Placing files on disk

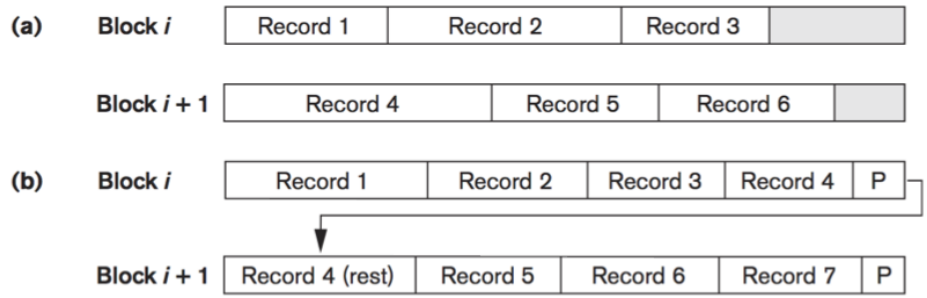
Techniques for placing file records on disk

- Record Blocking
- Allocating Blocks to Disk
- File Headers

Placing files on disk - Record Blocking

- A block is the unit of data transferred between the disk and memory, therefore records are allocated to the disk in blocks.
- If the record size is less than the block size, each block will contain multiple records. That is, for a fixed-length record size of R bytes and a block size of B, the blocking factor (bfr) is $floor(B/R)$. Each block will have unused space of $B - (bfr * R)$ bytes.
- To use this unused space, part of a record is stored in one block and the rest in another with a pointer at the end of the first block showing where the remainder is stored.
- Keeping records in blocks is an **unspanned** organisation of records while allowing records to be stored across blocks is referred to as **spanned** record organisation.

Figure 17.6
Types of record organization.
(a) Unspanned.
(b) Spanned.



Placing files on disk - Allocating Blocks to Disk

- There are several ways to allocate blocks of a file to disk:
  - **contiguous allocation**, allocate file blocks consecutively
  - **linked allocation**, each file block contains a pointer to the next file block
  - **combination of contiguous and linked**, file blocks are allocated to clusters and clusters are linked with pointers
  - **indexed allocation**, one or more index blocks contain pointers to file blocks

Placing files on disk - File Headers

- File headers or file descriptors contain information about the files needed by the system program to access the file records.
- The file header contains information to calculate the disk addresses of file blocks as well as record descriptions.
- To search for a record, one or more blocks are loaded into main memory buffers and the desired records are searched for in the buffers. This search could be linear if the address of the block is unknown.

Operations on Files

- Operations are grouped into retrieval and update operations. Retrieval operations do not change the data. Update operations change the file by inserting or deleting records.
- Search operations on files are based on similar conditions as are used in SQL SELECT statements.
  - Simple selection conditions are mostly used as is, while complex selection conditions are broken down by the DBMS into simple conditions.
  - The records are retrieved using this simple condition and then checked to see if they adhere to the complex condition or not.
  - The search process is linear.

Operations on Files

- The following are high-level operations for access file records as DBMS actual operations are vendor dependent. All, except for Open and Close are record-at-a-time operations.
  - Open
  - Reset
  - Find/Locate
  - Read/Get
  - FindNext
  - Delete
  - Modify
  - Insert
  - Close
- Modification of the records in the buffer are eventually committed to disk.
- Additional *set-at-a-time* operations include:
  - FindAll
  - Find/Locate n
  - FindOrdered
  - Reorganise

What is to come?

- How to organise records on disk
  - Sequential - Unordered (Heap) and Ordered and their search methods
  - Hashed - Dynamic, Extendible, and Linear
  - B-trees - used more with indexing
- Parallelisation