

# Storing Data: Disks and Files

## Chapter 9

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 1

---

---

---

---

---

---

---

---

# DBMS Organization

```

    graph TD
      A[SQL Query / Application] --> B[Query Evaluation Engine]
      B --> C[Files & Access Methods]
      C --> D[Buffer Management]
      D --> E[Disk Space Management]
      E --> F[(Disk)]
      subgraph This_lecture [This lecture]
        C
        D
        E
      end
  
```

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 2

---

---

---

---

---

---

---

---

# The Memory Hierarchy

Performance of Microprocessors and Memory improving faster than disks and tapes

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 3

---

---

---

---

---

---

---

---

### Why Not Store Everything in Main Memory?

- *Too expensive*: RAM: \$90 per GB, Disk: \$0.25 per GB
- *Main memory is volatile*: Want data to persist between runs
- Typical storage hierarchy:
  - Main memory (RAM) for currently used data
  - Disk for the main database (secondary storage)
    - Read and write data between disk and main memory
    - Both are high-cost operations, relative to in-memory operations, so must be planned carefully!
  - Tapes for archiving older versions of data (tertiary storage)
    - Sequential access devices

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 4

---

---

---

---

---

---

---

---

### Disks

- Secondary storage device of choice.
- Main advantage over tapes: *random access* vs. *sequential*.
- Data is stored and retrieved in units called *disk blocks* or *pages*.
- Unlike RAM, time to retrieve a disk page varies depending upon location on disk.
  - Therefore, relative placement of pages on disk has major impact on DBMS performance!

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 5

---

---

---

---

---

---

---

---

### Magnetic Disks

Set of tracks with same diameter called a *cylinder*

Data stored in *blocks*. Size of block is a multiple of sector size.

Only one disk head reads or writes at a time.

The diagram shows a vertical stack of three platters. A central spindle is visible. A disk arm extends from the left, holding a disk head that is positioned over the platters. Labels include: Disk head, Tracks, Sector, Platters, and Disk arm. A double-headed arrow labeled 'Arm movement' indicates the horizontal movement of the disk arm.

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 6

---

---

---

---

---

---

---

---

## Performance Implications

- Data must be in memory for DMBS to use.
- Unit of transfer is a block. Whole block must be transferred. Reading or writing a disk block is called an I/O.
- Disk geometry affects access time
  - **Seek time**: time to move disk head to appropriate track
  - **Rotational delay**: time waiting for block to move under disk head
  - **Transfer time**: time to read or write block once head is positioned

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 7

---

---

---

---

---

---

---

---

## Arranging Blocks on Disk

Access time = seek time + rotational delay + transfer time

- **GOAL**: Minimize seek time and rotational delay
- 'Next' block concept:
  - blocks on same track, followed by
  - blocks on same cylinder, followed by
  - blocks on adjacent cylinder
- Arranging blocks so they are read and written *sequentially* is important to reducing time spent doing disk I/O

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 8

---

---

---

---

---

---

---

---

## Disk Space Manager

- Manages space on disk; **Page**: unit of data
  - Mapped to a disk block
  - Higher level calls: allocate/de-allocate, read/write pages
    - Insulates upper levels from the underlying hardware & OS
- Efficiently lay out pages on disk
- Bookkeeping for disk blocks (free or in-use)
  - **Linked-list of free blocks** OR
  - **Bitmap: 1 bit per block**
- Can we use OS files?
  - DB Disk manager can be built using OS files
    - Portability and technical limitations
  - Many DBMSs do their own disk management

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 9

---

---

---

---

---

---

---

---

## Buffer Management in a DBMS

- Data must be in RAM for DBMS to operate on it!
  - Can't keep all the DBMS pages in main memory
- Buffer Manager: Efficiently uses main memory
  - Memory divided into **buffer frames**: slots for holding disk pages

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 10

---

---

---

---

---

---

---

---

---

---

---

---

## Buffer Manager

- Bookkeeping per frame:
  - pin count**: # of active users of the frame
    - Pinning – “I’m using this page”
    - Unpinning – “I’m no longer using this page”
  - dirty bit**: indicates if page has been modified since it was brought in from disk

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 11

---

---

---

---

---

---

---

---

---

---

---

---

## Buffer Manager

- When a page is requested:
  - If page is in buffer pool
    - Increment the frame’s pin count
    - Return main memory address of the frame
  - Else
    - Choose a frame for replacement
      - Use replacement policy
      - Only choose a frame with pin count == 0
    - If frame is dirty, write old page to disk
    - Read requested page into chosen frame
      - Set pin count = 1 (“pin the page”)
      - Set dirty bit = false

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 12

---

---

---

---

---

---

---

---

---

---

---

---

## Buffer Replacement Policy

- Chose a frame for replacement
  - Least-recently-used (LRU), Clock, MRU etc.
- LRU: queue of pointers to "empty" frames
  - Add to end of queue, grab frames from front of queue
- Clock: variant of LRU, but lower overhead
- Policy can have big impact on # of I/O's; depends on the *access pattern*.
- *Sequential flooding*: Nasty situation caused by LRU + repeated sequential scans.
  - # buffer frames < # pages in file
    - Use MRU

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 13

---

---

---

---

---

---

---

---

## Files of Records (tuples)

- Pages are great for storage system, but DBMS views data as a collection of records
- Important questions:
  - How is a record organized / implemented?
  - How are records laid out on pages?
  - How are pages organized into files?

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 14

---

---

---

---

---

---

---

---

## Record Formats

- Two kinds:
  - Fixed-length records
    - Each field has a fixed length, and the number of fields is fixed
  - Variable-length records
    - Some of the fields are of variable length
    - E.g., variable-length strings

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 15

---

---

---

---

---

---

---

---

## Buffer Management in DBMS vs. OS

- Obvious similarities between DBMS buffer management, virtual memory (OS)
- Why not use the OS?
  - DBMS can better predict access patterns (limited set of operations)
    - Adjust replacement policy
    - **Pre-fetch** pages based on predictable access patterns
  - Portability issues
  - DBMS recovery requires the ability to force a page to disk

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 16

---

---

---

---

---

---

---

---

## Record Formats: Fixed Length

- All records are the same length
- Information about field types same for all records in a file; stored in *system catalogs*.
- What about null values?

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 17

---

---

---

---

---

---

---

---

## Record Formats: Variable Length

Two alternative formats:

- Second alternative offers direct access to *i*th field
  - Efficient storage of nulls
  - Small directory overhead.
- Issues with growing records!
  - Need to shift subsequent values
  - Record may exceed size of page

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 18

---

---

---

---

---

---

---

---

## Page Formats

- Page contains a collection of records
- Think of page as a collection of *slots*, each of which contains a record
- Record ID (RID):  $\langle \text{page id, slot \#} \rangle$
- Many different slotted page organizations.
  - Must support search, insert, delete records

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 19

---

---

---

---

---

---

---

---

## Page Formats: Fixed Length Records

Record id =  $\langle \text{page id, slot \#} \rangle$

First alternative: moving records changes rid

may not be acceptable.

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 20

---

---

---

---

---

---

---

---

## Page Formats: Variable Length Records

**Delete a record?**

Slot Entry: Offset, reflen

- Directory grows backwards!
- Move records on same page
  - rid unchanged
- Good for fixed-length records too.

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 21

---

---

---

---

---

---

---

---

## Unordered (Heap) Files

- Simplest file structure contains records in no particular order.
- Page contains a set of records
- As file grows and shrinks, disk pages are allocated and de-allocated.
- Many alternatives for storing file

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 22

---

---

---

---

---

---

---

---

## Heap File Implemented as a List

- (heap file name, header page id) recorded in a known location
- Each page contains two pointers plus data: Pointer = Page ID (pid)
- Pages in the free space list have "some" free space
- What happens with variable length records?
- Fetch a record with rid

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 23

---

---

---

---

---

---

---

---

## Heap File Using a Page Directory

- Entry for a page:
  - Free/full
  - Number of free bytes
- Can locate pages for new tuples faster!

Directory itself is implemented as a linked list of pages

2/3/09 EECS 484: Database Management Systems, Kristen LeFevre 24

---

---

---

---

---

---

---

---