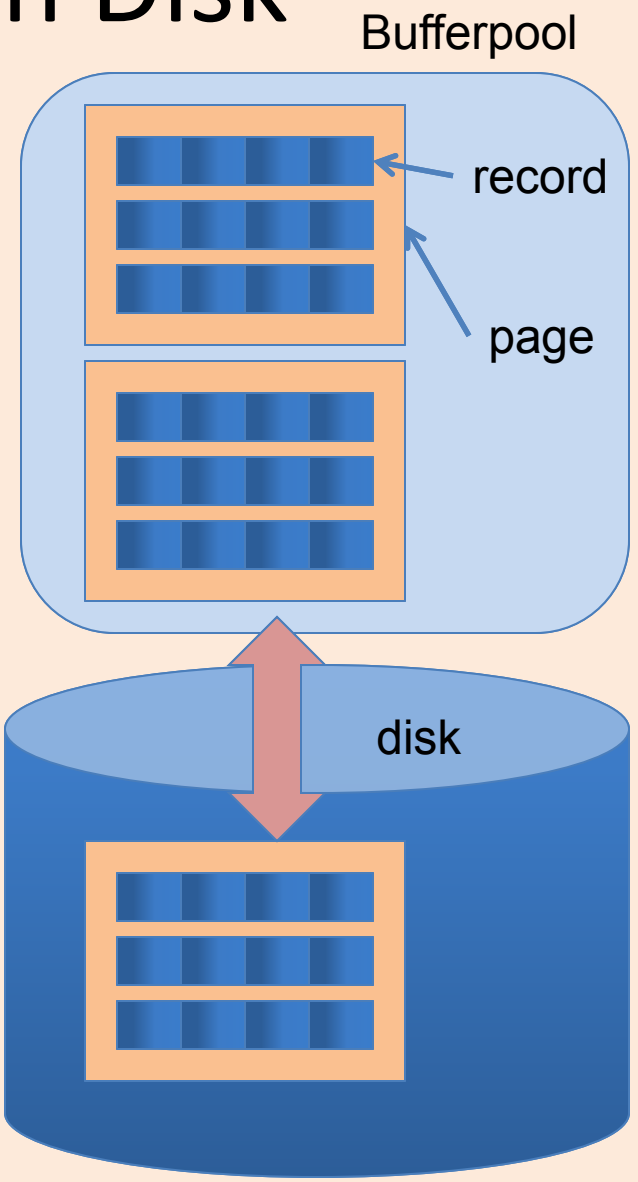
# ICS 421 Spring 2010
# Storage

Asst. Prof. Lipyeow Lim
Information & Computer Science Department
University of Hawaii at Manoa

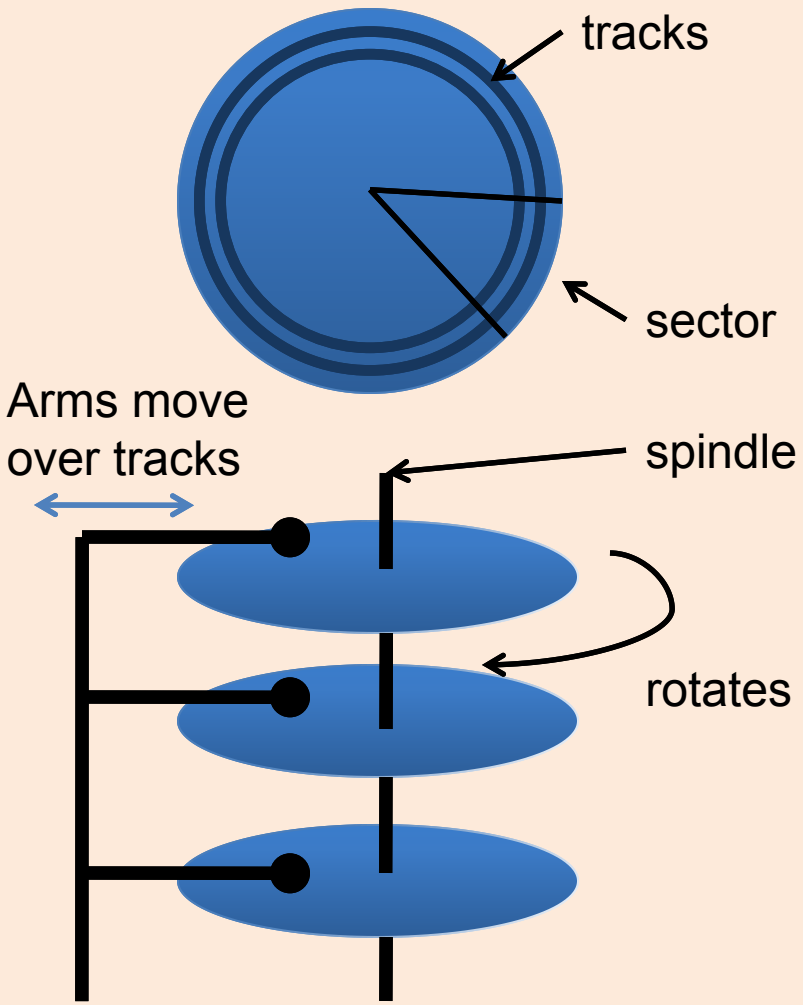
# Relational Tables on Disk

- **Record** -- a tuple or row of a relational table
- **RIDs** – record identifiers that uniquely identify a record across memory and disk
- **Page** – a collection of records that is the unit of transfer between memory and disk
- **Bufferpool** – a piece of memory used to cache data and index pages.
- **Buffer Manager** – a component of a DBMS that manages the pages in memory
- **Disk Space Manager** – a component of a DBMS that manages pages on disk



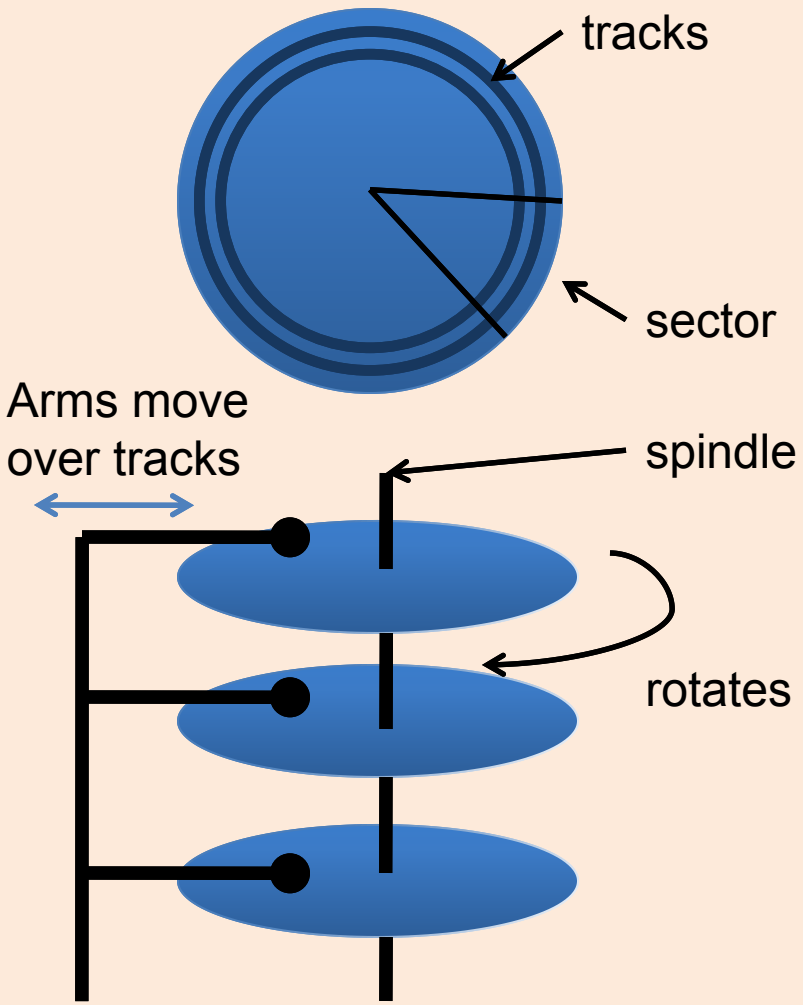
# Magnetic Disks

- A disk or platter contains multiple concentric rings called **tracks**.
- Tracks of a fixed diameter of a spindle of disks form a **cylinder**.
- Each track is divided into fixed sized **sectors** (ie. “arcs”).
- Data stored in units of disk **blocks** (in multiples of sectors)
- An array of **disk heads** moves as a single unit.
- **Seek time**: time to move disk heads over the required track
- **Rotational delay**: time for desired sector to rotate under the disk head.
- **Transfer time**: time to actually read/write the data

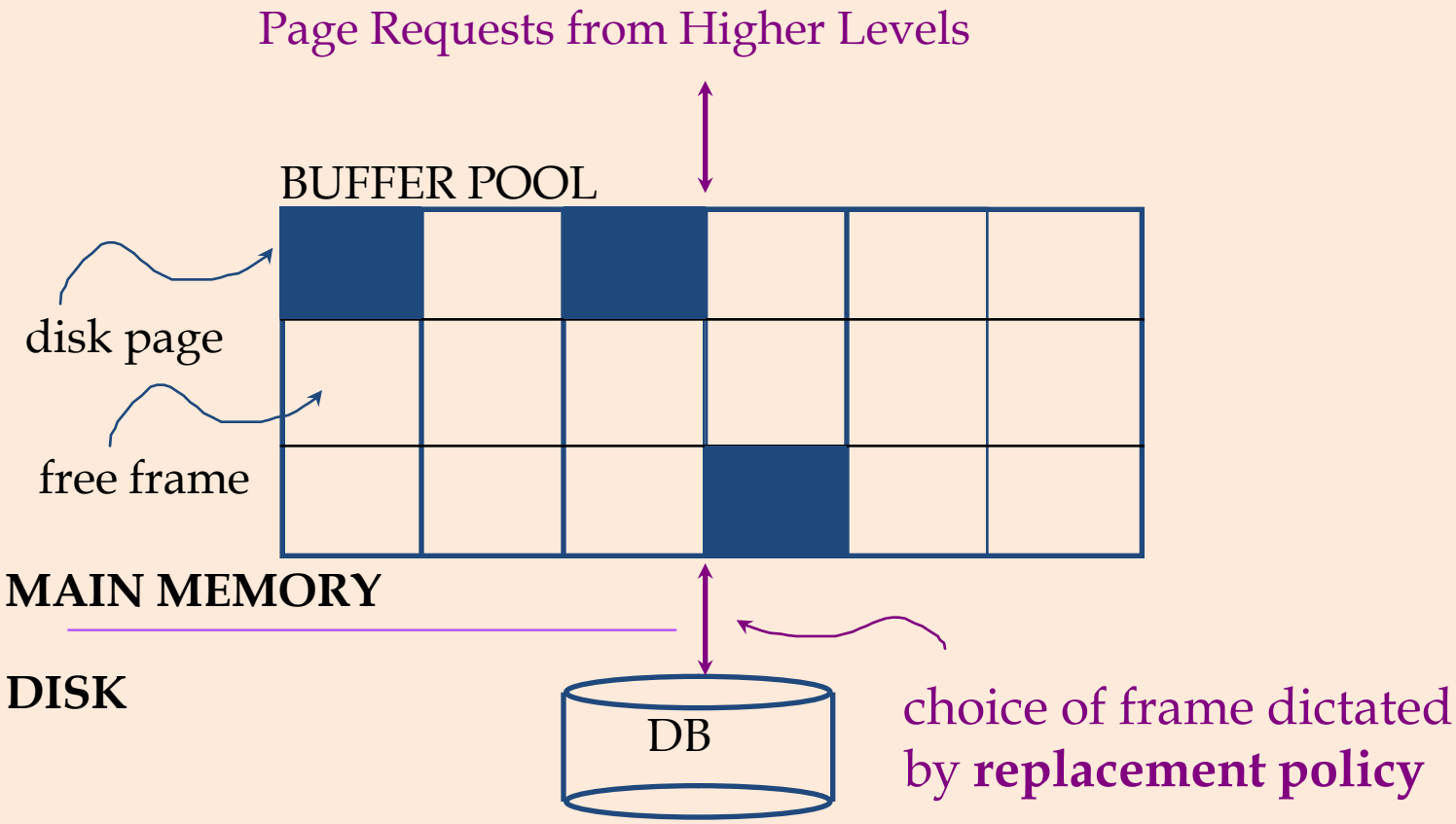


# Accessing Data on Disk

- **Seek time**: time to move disk heads over the required track ≈8.5msec
- **Rotational delay**: time for desired sector to rotate under the disk head.
  - Assume uniform distribution, on average time for half a rotation ≈4 msec
- **Transfer time**: time to actually read/write the data ≈7200rpm
- Seek time and rotational delay dominate
  - Key to lower I/O cost: reduce seek/rotation delays!
  - Pre-fetching



# Buffer Management in a DBMS




- Data must be in RAM for DBMS to operate on it!
- Table of <frame#, pageid> pairs is maintained.

# Page Requests

- If requested page is not in pool
  - If buffer pool is full
    - Choose frame for replacement
    - If frame is dirty, write to disk
  - Read requested page into empty frame
  - Pin the page and return its address
- Requestor of page
  - Unpins the page
  - Set dirty bit if modified
- Page in pool may be requested many times,
  - A *pin count* is used.
  - A page is a candidate for replacement iff *pin count* = 0.
- Concurrency Control & recovery may entail additional I/O when a frame is chosen for replacement. (*Write-Ahead Log* protocol; more later.)

# Buffer Replacement Policy

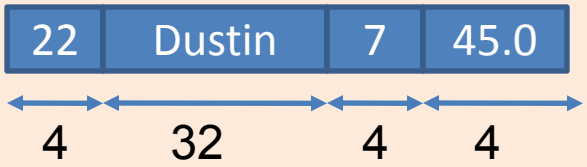
- Frame is chosen for replacement by a *replacement policy:*
  - Least-recently-used (LRU), Clock, FIFO, MRU etc.
- Policy can have big impact on # of I/O's; depends on the *access pattern*.
- *Sequential flooding*: Nasty situation caused by LRU + repeated sequential scans.
  - # buffer frames < # pages in file means each page request causes an I/O. MRU much better in this situation (but not in all situations, of course).

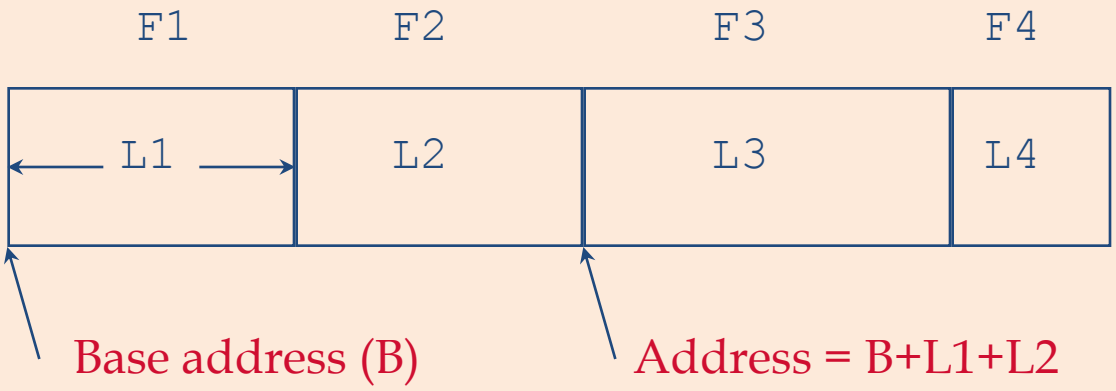
# DBMS vs OS

- OS does disk space & buffer mgmt: why not let OS manage these tasks?
- Differences in OS support: portability issues
- Some limitations, e.g., files can't span disks.
- Buffer management in DBMS requires ability to:
  - pin a page in buffer pool, force a page to disk (important for implementing CC & recovery),
  - adjust *replacement policy,* and pre-fetch pages based on access patterns in typical DB operations.

# Record Formats: Fixed Length

| sid int | sname char(32) | rating int | age real |
|---|---|---|---|
| 22 | Dustin | 7 | 45.0 |

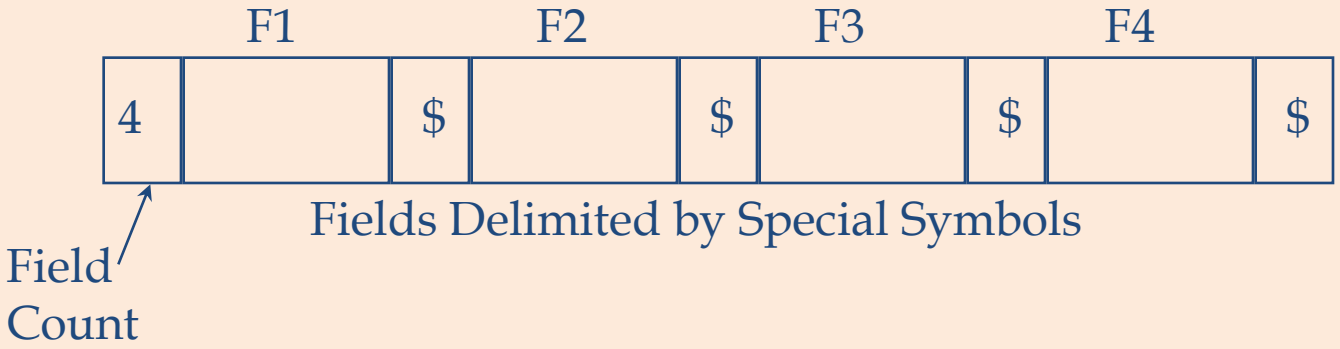


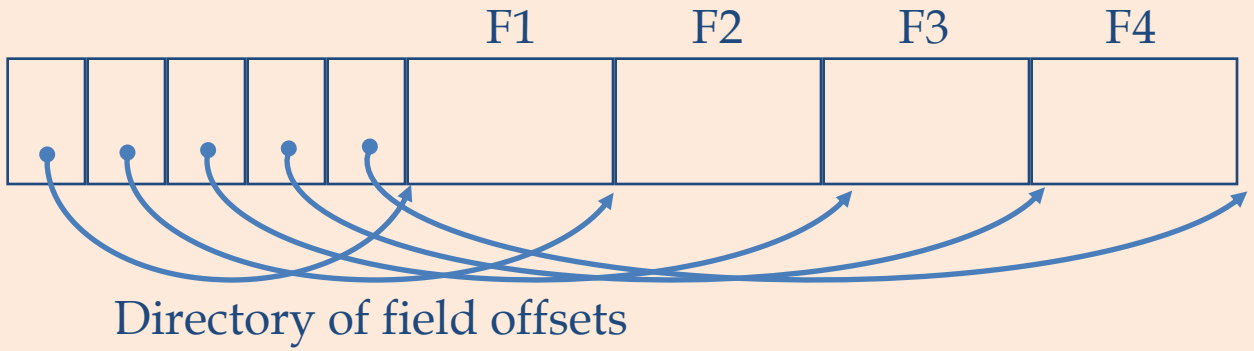

- Information about field types same for all records in a file; stored in *system catalogs.*
- Finding *i'th* field does not require sequential scan of record.

# Record Formats: Variable Length

Option 1:

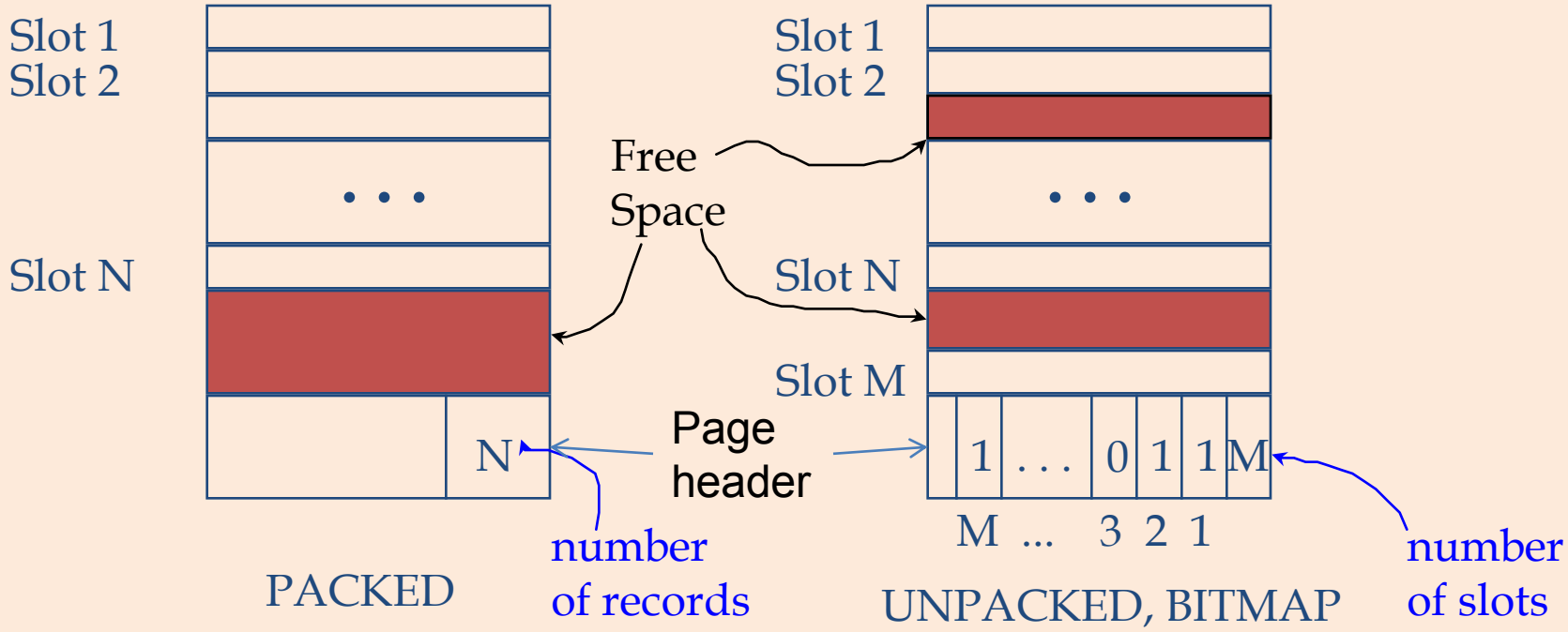



Option 2:



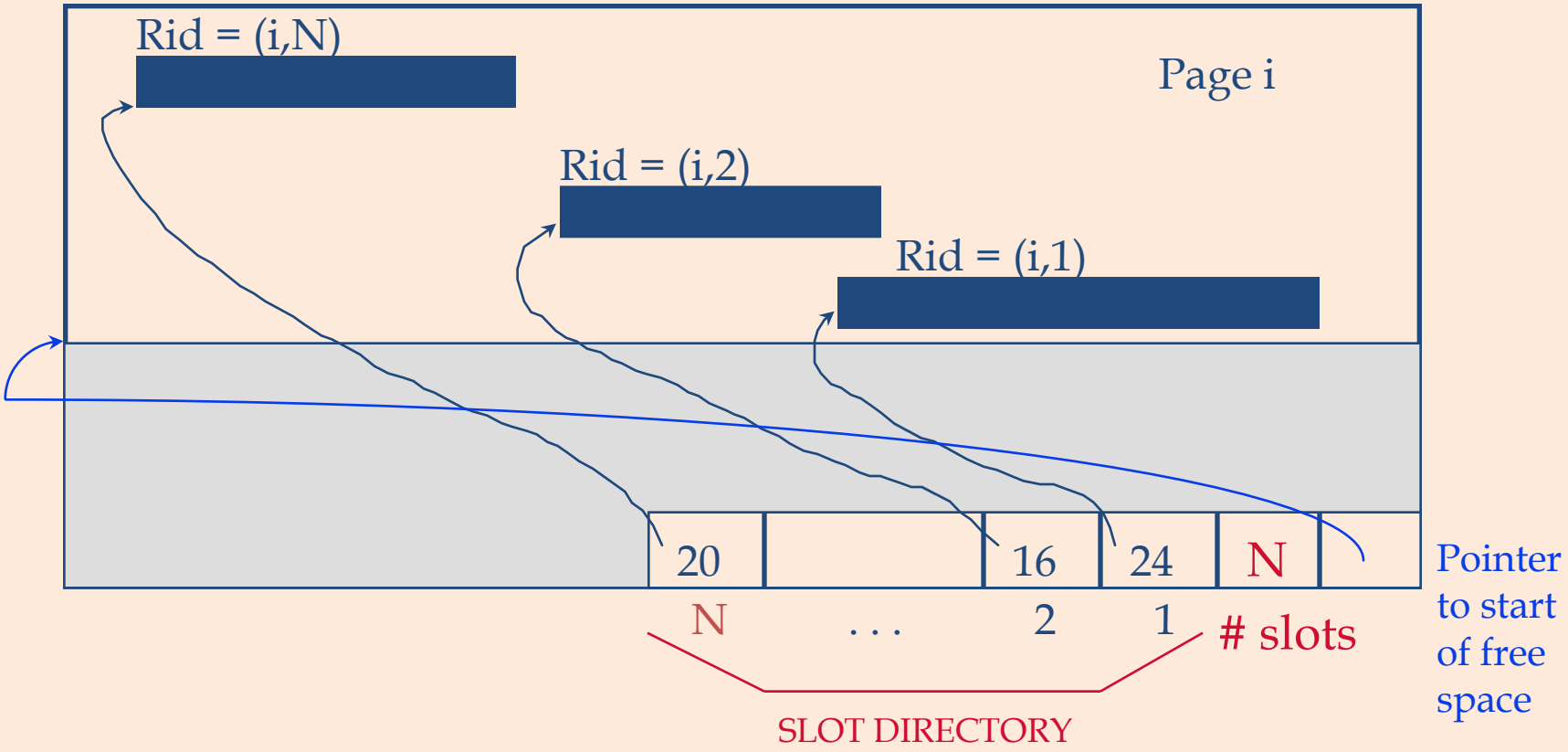

- Option 2 offers direct access to i-th field, efficient storage of nulls with small directory overhead

# Page Formats: Fixed Length Records




- *Record id = <page id, slot #>. In first alternative, moving records for free space management changes rid; may not be acceptable.*

# Page Formats: Variable Length Records




- Can move records on page without changing rid; so, attractive for fixed-length records too

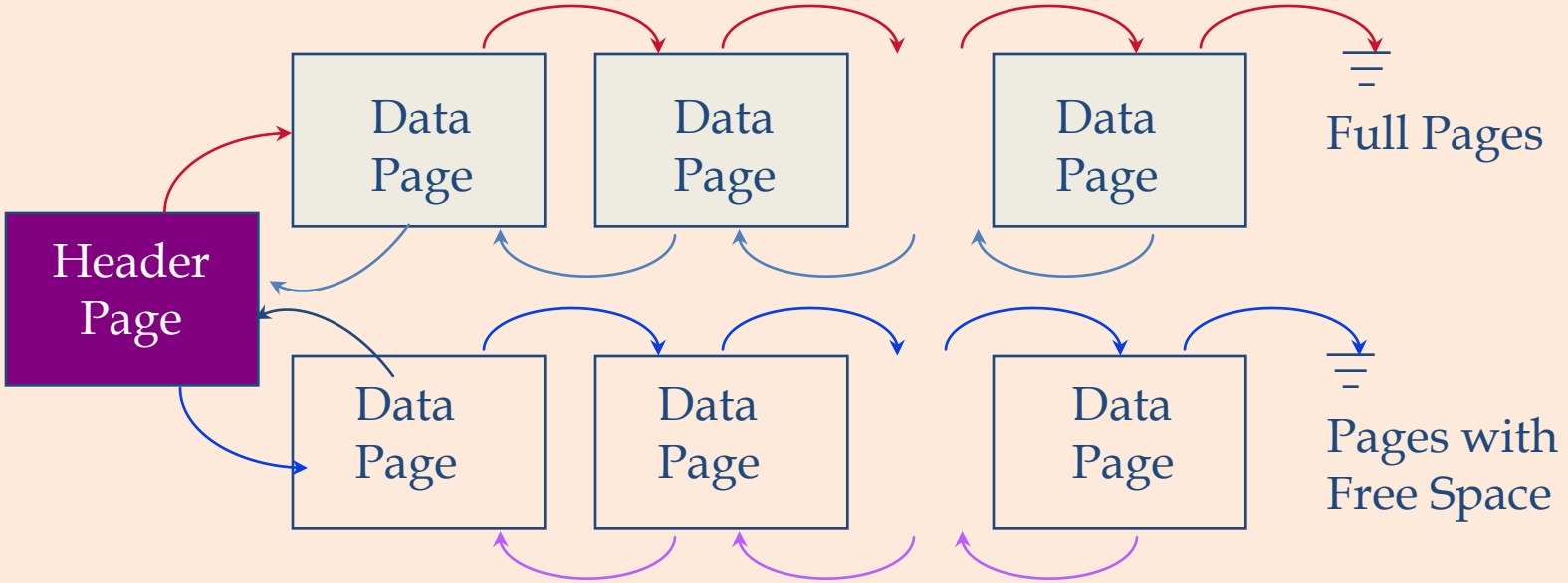
# Files of Records

- Page or block is OK when doing I/O, but higher levels of DBMS operate on *records*, and *files of records*.
- FILE: A collection of pages, each containing a collection of records. Must support:
  - insert/delete/modify record
  - read a particular record (specified using *record id*)
  - scan all records (possibly with some conditions on the records to be retrieved)

# Unordered Heap Files

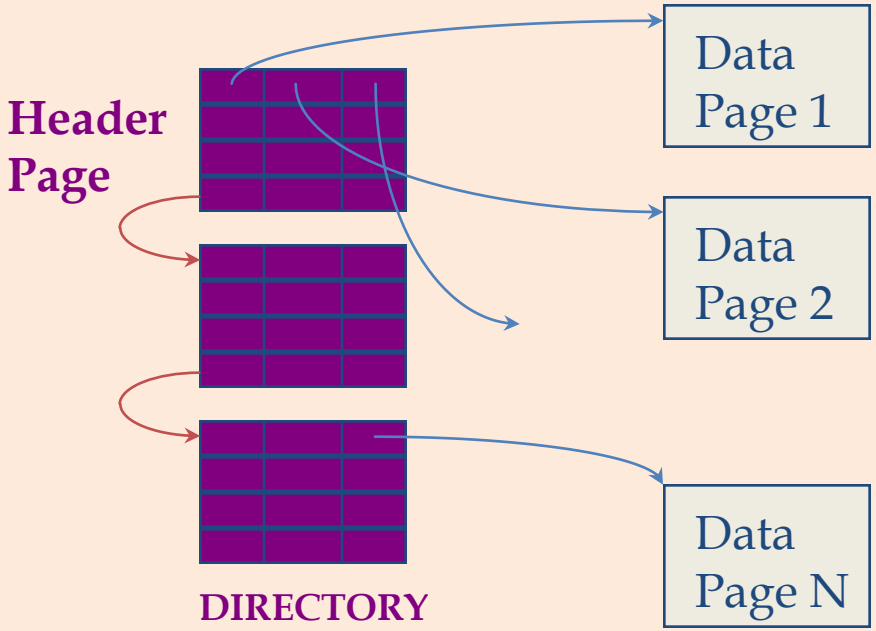
- Simplest file structure contains records in no particular order.
- As file grows and shrinks, disk pages are allocated and de-allocated.
- To support record level operations, we must:
  - keep track of the *pages* in a file
  - keep track of *free space* on pages
  - keep track of the *records* on a page
- How do we keeping track of these ?

# Heap File using a Linked List




- The header page id and Heap file name (corresponds to a table) must be stored someplace.
- Each page contains 2 `pointers' plus data.

# Heap File using a Page Directory




- The entry for a page can include the number of free bytes on the page.
- The directory is a collection of pages; linked list implementation is just one alternative.
  - *Much smaller than linked list of all HF pages*!