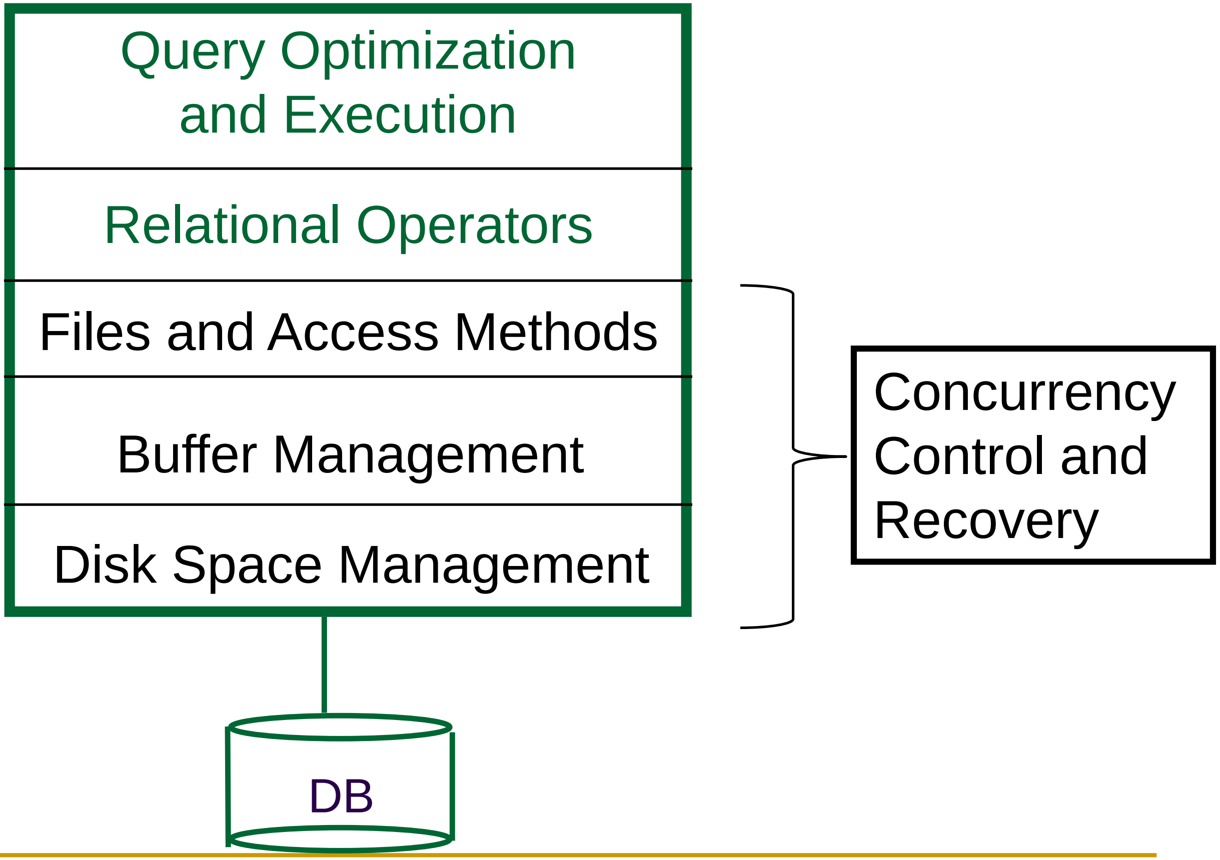
Storing Data: Disks and Files

SUN YAT-SEN UNIVERSITY

Block diagram of a DBMS



Disks

- DBMS stores information on disks.

- Fixed unit of transfer
  - Read/write *disk blocks* or *pages* (8K)

    磁盘块　　页/数据页/磁盘页

- *Not* “random access” (vs. RAM)
  - Time to retrieve a block depends on location
  - Relative placement of blocks on disk has major impact on DBMS performance!

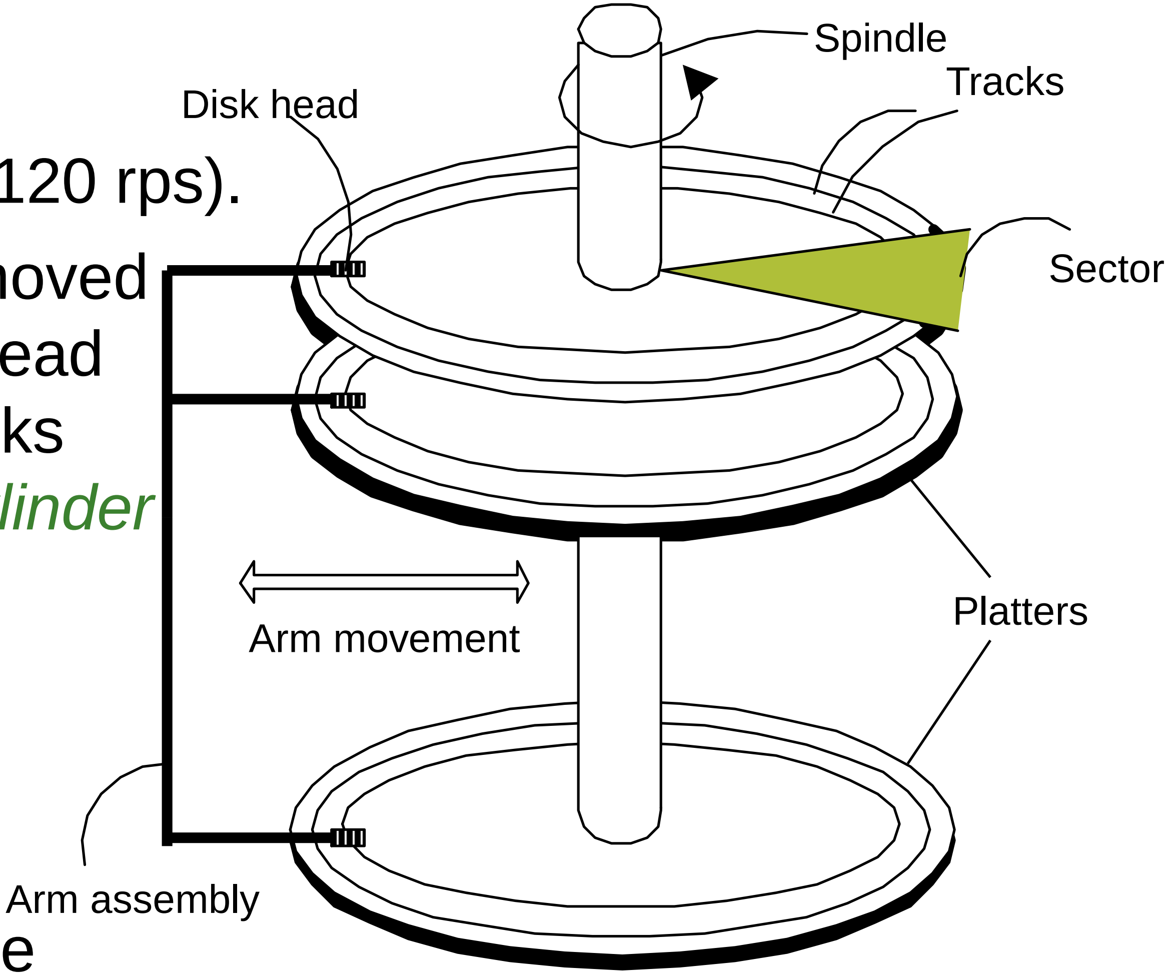
Components of a Disk

The platters spin (say, 120 rps).

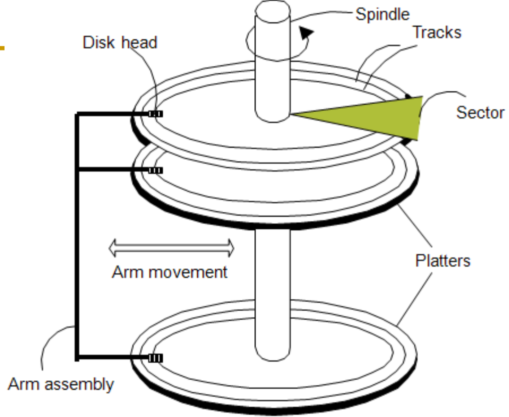
The arm assembly is moved in or out to position a head on a desired track. Tracks under heads make a *cylinder* (imaginary!).

Only one head reads/writes at any one time.

- *Block size* is a multiple of *sector size* (which is fixed).

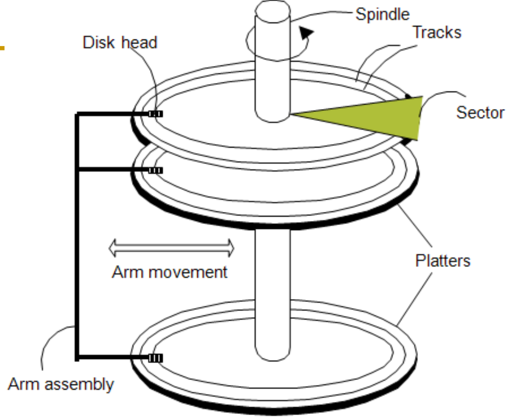


Accessing a Disk Page



- Time to access (read/write -- 存取时间) a disk block:
  - *seek time* (寻道时间-moving arms to position disk head on track)
  - *rotational delay* (旋转延迟-waiting for block to rotate under head)
  - *transfer time* (传输时间-actually moving data to/from disk surface)
- Seek time and rotational delay dominate.
  - Seek time varies from 0 to 10msec . – 平均时间
  - Rotational delay varies from 0 to 3msec
  - Transfer rate around 0.2msec per 8K block
- Key to lower I/O cost: reduce seek/rotation delays! Hardware vs. software solutions?

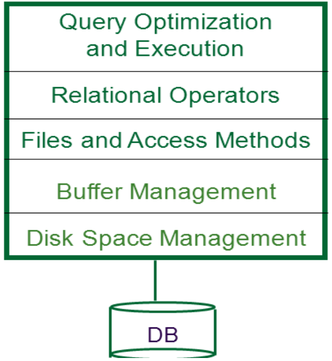
Arranging Pages on Disk



- \`*Next*’ block concept:
  - blocks on same track, followed by
  - blocks on same cylinder, followed by
  - blocks on adjacent cylinder

- Blocks in a file should be arranged sequentially on disk (by \`next’), to minimize seek and rotational delay.
  - For a sequential scan, *pre-fetching (预读取)* several pages at a time is a big win!

Disk Space Management



Query Optimization and Execution

Relational Operators

Files and Access Methods

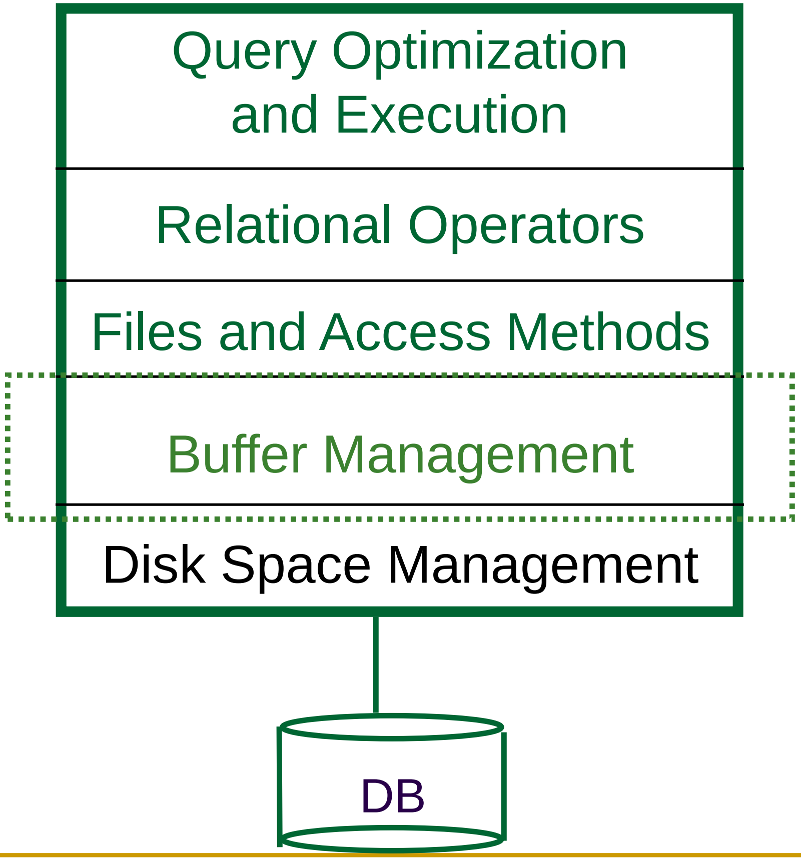
Buffer Management

Disk Space Management

DB

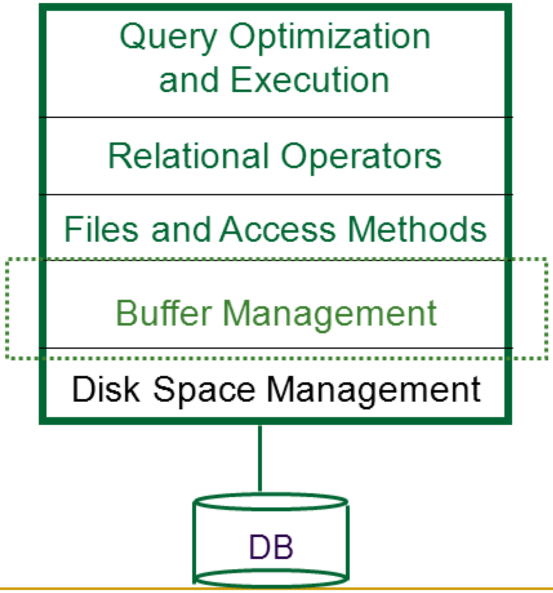
- Lowest layer of DBMS, manages space on disk
- 接口: Higher levels call upon this layer to:
  - allocate/de-allocate a page
  - read/write a page
- 期望性能:Request for *a sequence* of pages best satisfied by pages stored sequentially on disk!
  - Responsibility of disk space manager.
  - Higher levels don’t know how this is done, or how free space is managed.
  - They may make performance assumptions!
    - Hence disk space manager should do a decent job.

Context


Query Optimization
and Execution
Relational Operators
Files and Access Methods
Buffer Management
Disk Space Management
DB

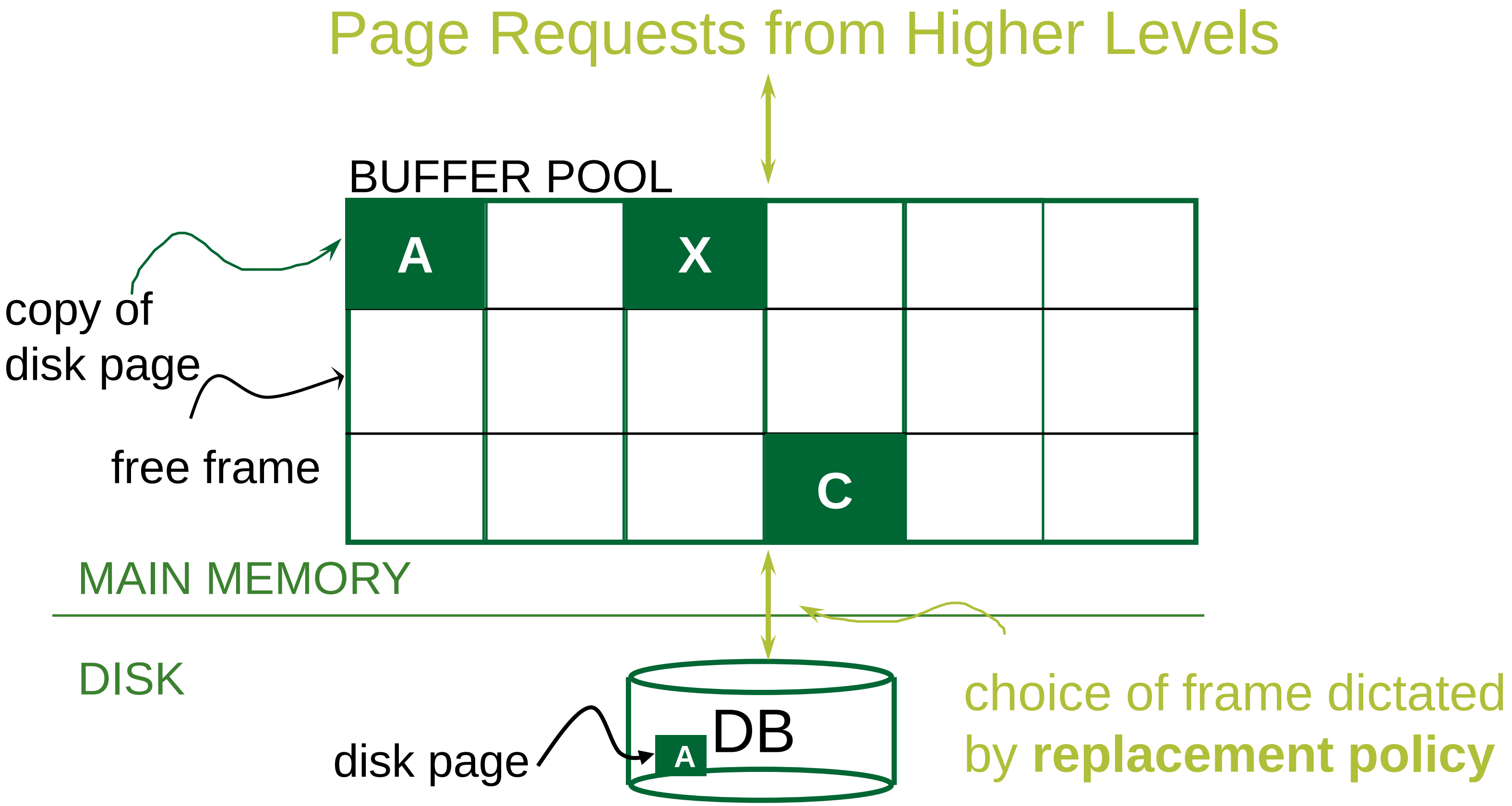
Buffer Management in a DBMS
缓冲区管理

- *Data must be in RAM for DBMS to operate on it!*
- *BufMgr hides the fact that not all data is in RAM*



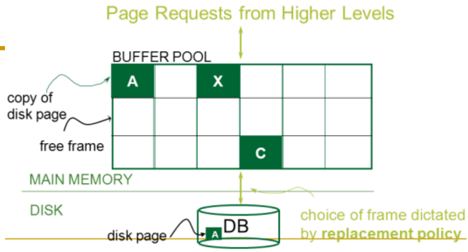
Buffer Management in a DBMS

缓冲区管理—how?


Page Requests from Higher Levels
BUFFER POOL
A
X
C
copy of disk page
free frame
MAIN MEMORY
DISK
disk page
A
DB
choice of frame dictated by **replacement policy**

划分帧→(空闲帧/内容帧) → 副本→ 请求→替换策略

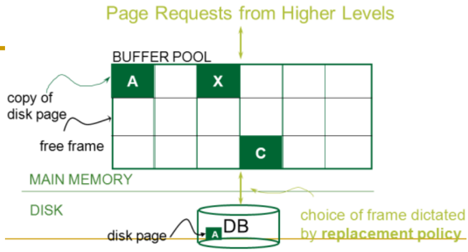
When a Page is Requested ...


Page Requests from Higher Levels
BUFFER POOL
A
X
copy of disk page
free frame
C
MAIN MEMORY
DISK
disk page
A
DB
choice of frame dictated by replacement policy

- Buffer pool information table contains: *<frame#, pageid, pin_count, dirty>*

1. If requested page is not in pool:
    a. Choose a frame for *replacement.*
       Only "un-pinned" pages are candidates!
    b. If frame "dirty", write current page to disk
    c. Read requested page into frame
2. *Pin* the page and return its address.

More on Buffer Management

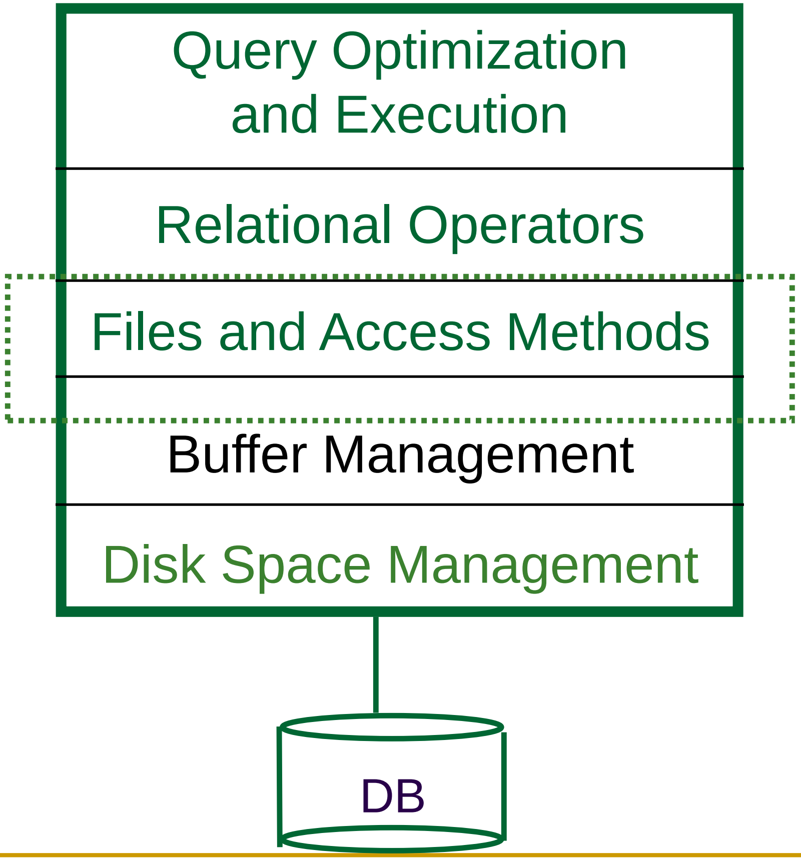

Page Requests from Higher Levels
BUFFER POOL
A
X
C
copy of disk page
free frame
MAIN MEMORY
DISK
disk page
A
DB
choice of frame dictated by **replacement policy**

- Requestor of page must eventually:
  1. *unpin* it -- 解钉 pin_count--
  2. indicate whether page was modified via *dirty* bit.

  <frame#, pageid, pin_count, dirty>

- Page in pool may be requested many times,
  - a *pin _count* is used.
  - To pin a page(钉住页): pin_count++
  - A page is a candidate for replacement iff *pin_count* == 0 (“*unpinned*”)

Context


Query Optimization
and Execution
Relational Operators
Files and Access Methods
Buffer Management
Disk Space Management
DB

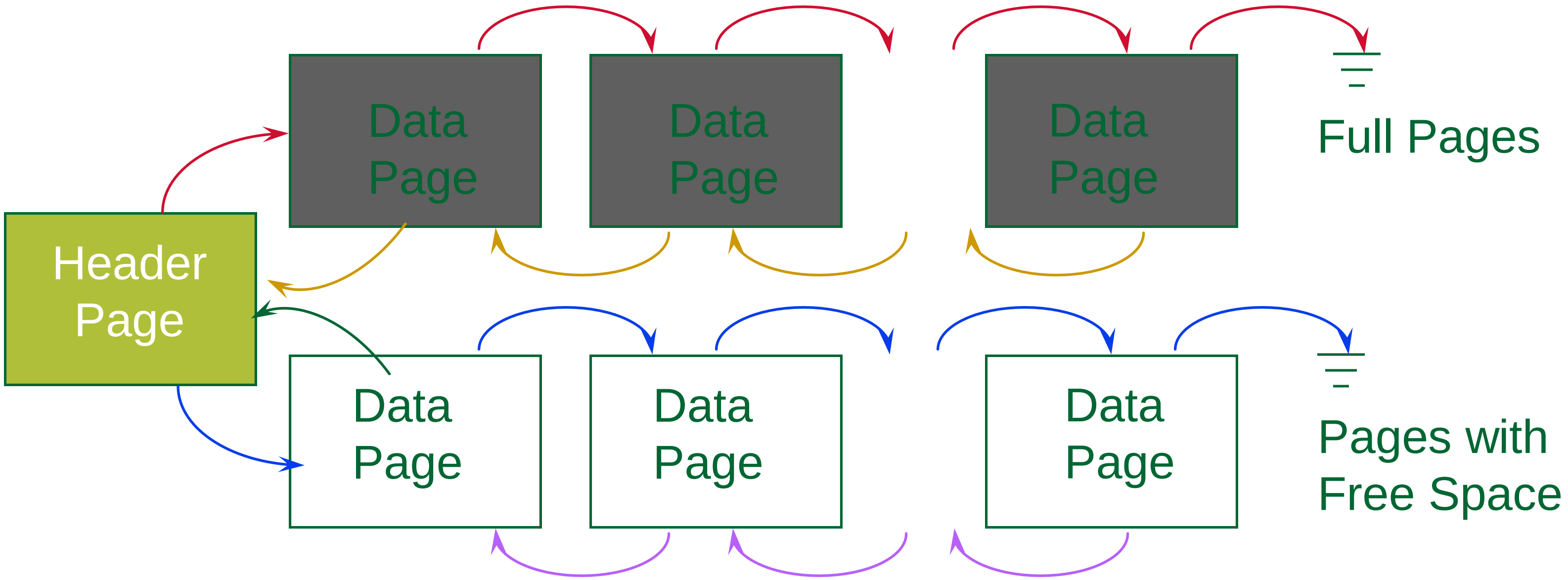
Files of Records (记录文件)

- Blocks are the interface for 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
  - fetch a particular record (specified using *record id*)
  - scan all records (possibly with some conditions on the records to be retrieved)

Unordered (Heap) Files --堆文件

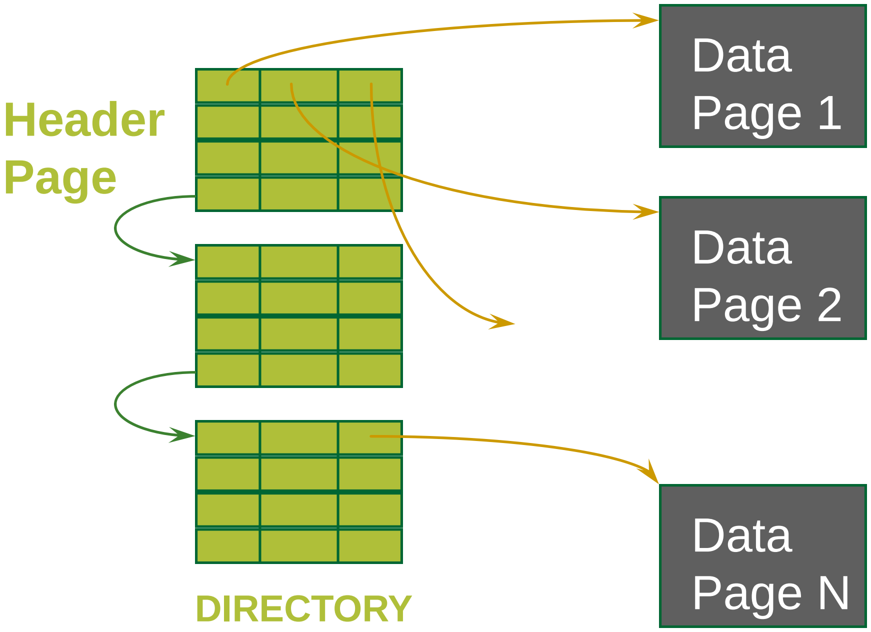
- Collection of records in no particular order.
  - As file shrinks/grows, disk pages (de)allocated
- To support record level operations, we must:
  - keep track of the *pages* in a file
  - keep track of *free space* on pages
  - keep track of the *records* on a page
- There are many alternatives for keeping track of this.
  - We'll consider two.

Heap File Implemented as a List (页链表)



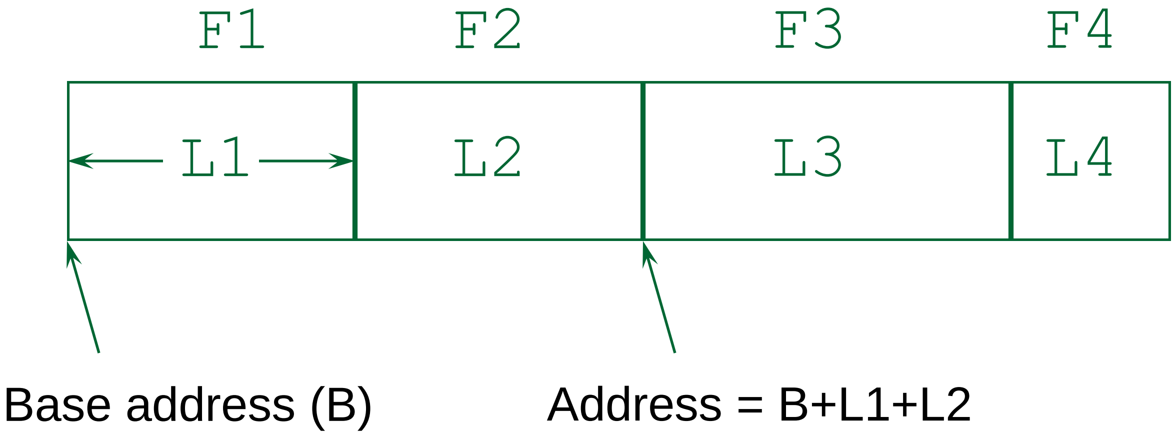
- The header page(首页) id and Heap file name must be stored someplace.
  - Database “catalog”
- Each page contains 2 `pointers’ plus data.

Heap File Using a Page Directory(页目录)



- The directory is itself a collection of pages;
  - each page can hold several entries.
  - The entry (目录项) for a page can include the number of free bytes on the page. <pointer,free_number>
- To insert a record, we can search the directory to determine which page has enough space to hold the record.

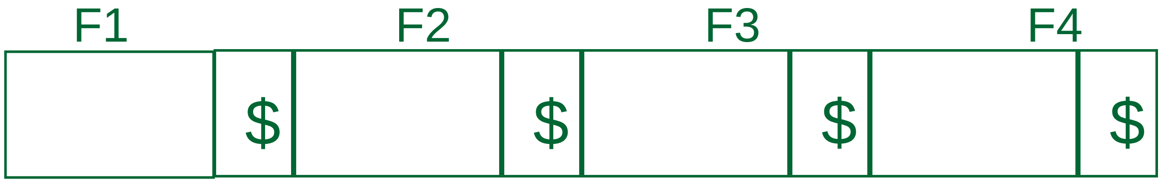
Record Formats(记录格式): Fixed Length(定长记录)



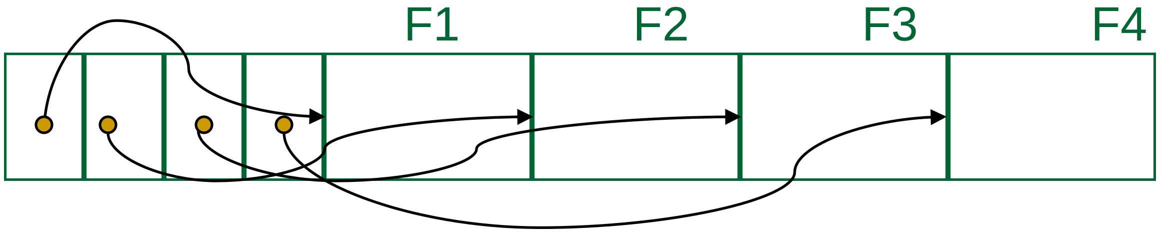
- Information about field types same for all records in a file; stored in *system catalogs.*
- Finding *i'th* field done via arithmetic.

Record Formats:
Variable Length(变长记录)

- Two alternative formats (# fields is fixed):



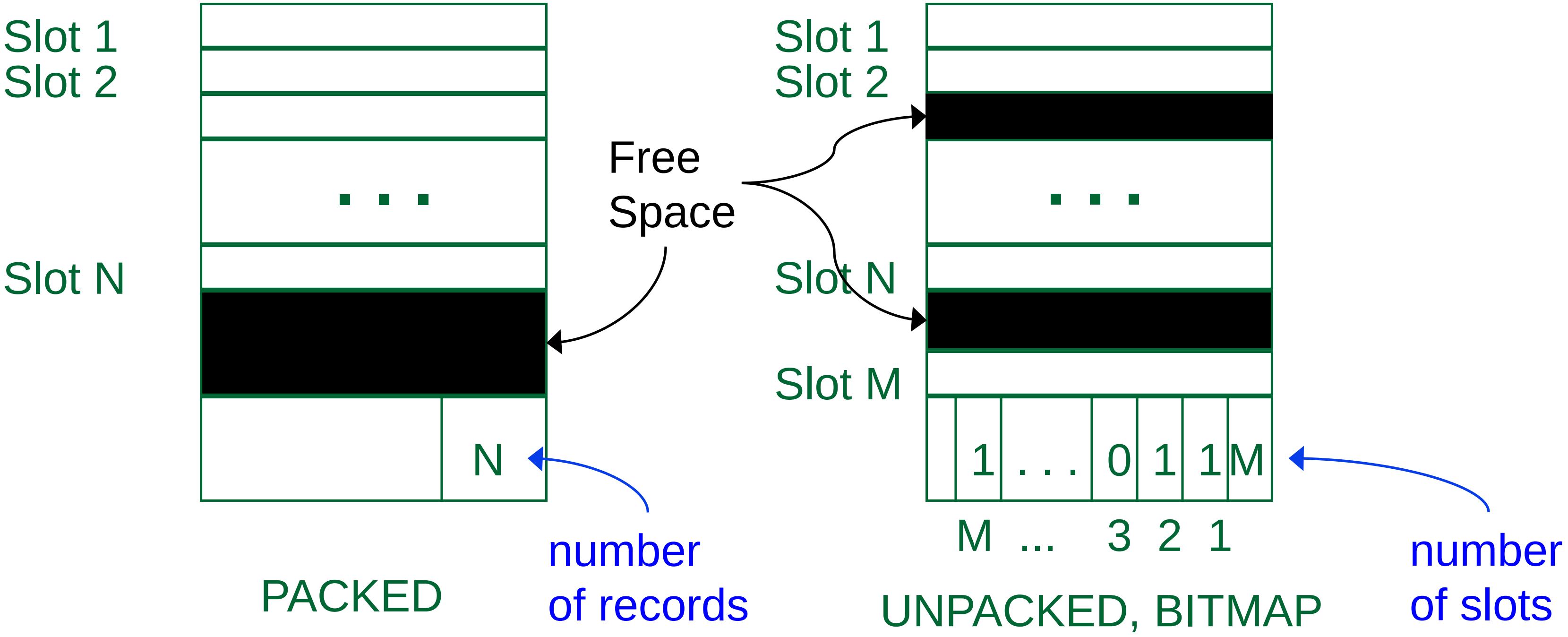
Fields Delimited by Special Symbols



Array of Field Offsets-字段偏移量数组

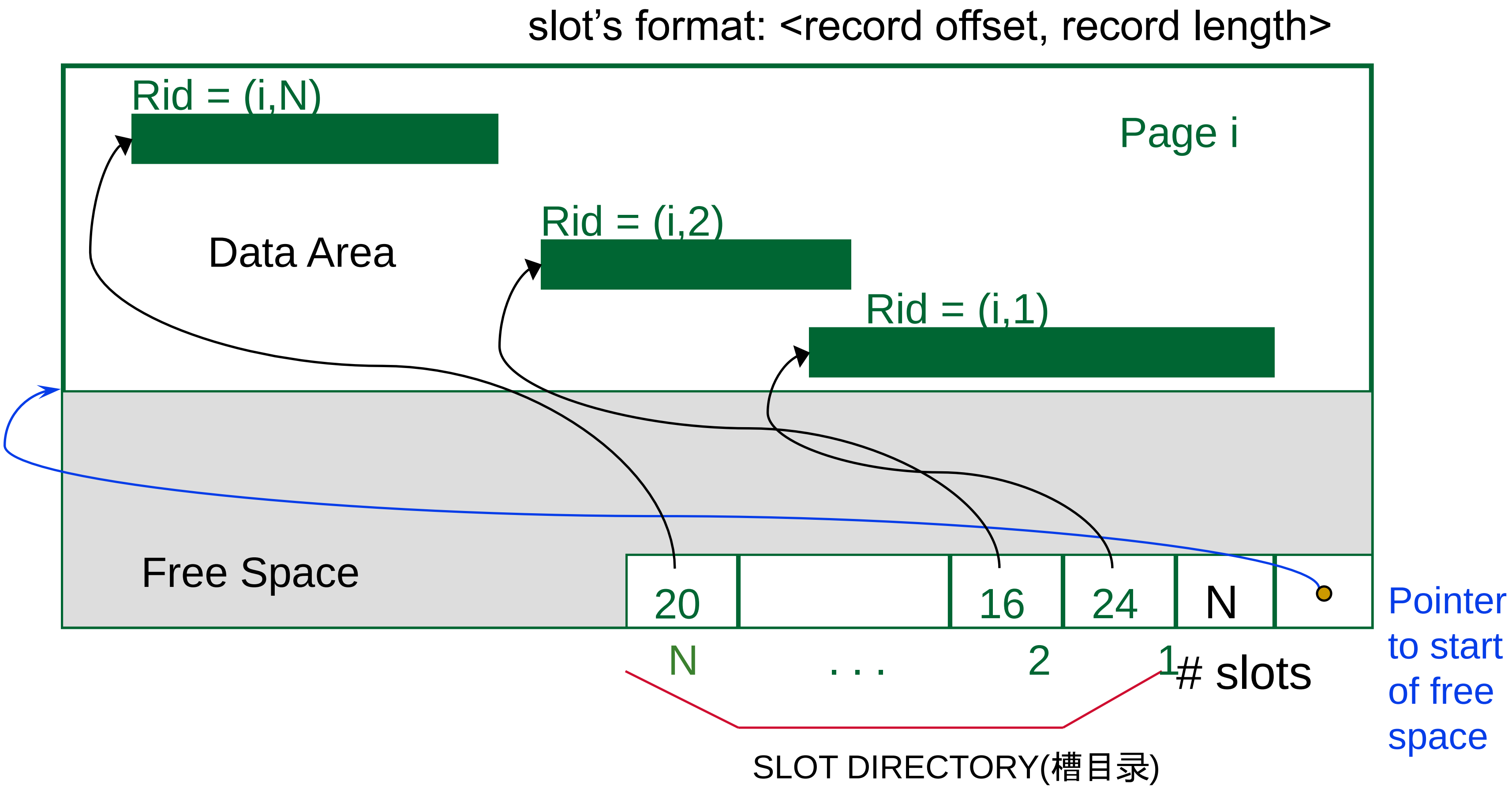
⌧ Second offers direct access to i’th field, efficient storage of *nulls* (special *don’t know* value); small directory overhead.

Page Formats(页格式): Fixed Length Records



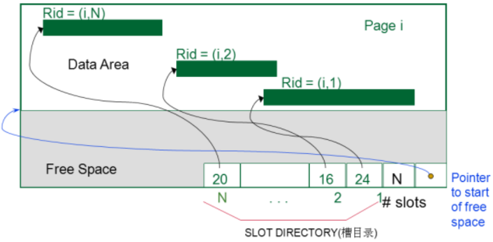
- Record id = <page id, slot #>.
- *In first alternative,*
  - *moving records for free space management changes rid;*
  - *may not be acceptable.*

Page Formats: Variable Length Records

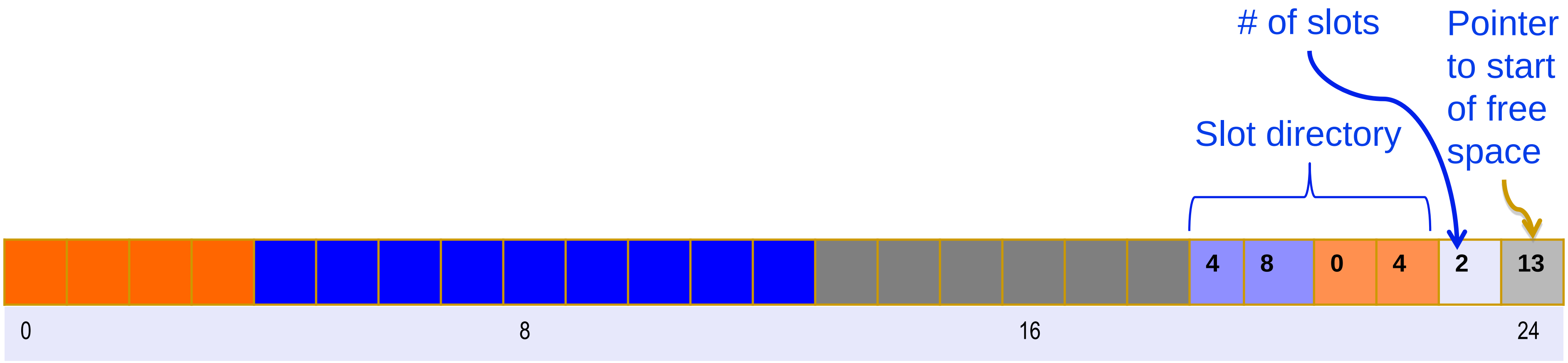


- *Can move records on page without changing rid; so, attractive for fixed-length records too.*

Slotted page: a detailed view



slot’s format: <record offset, record length>


# of slots
Pointer to start of free space
Slot directory
4
8
0
4
2
13
0
8
16
24