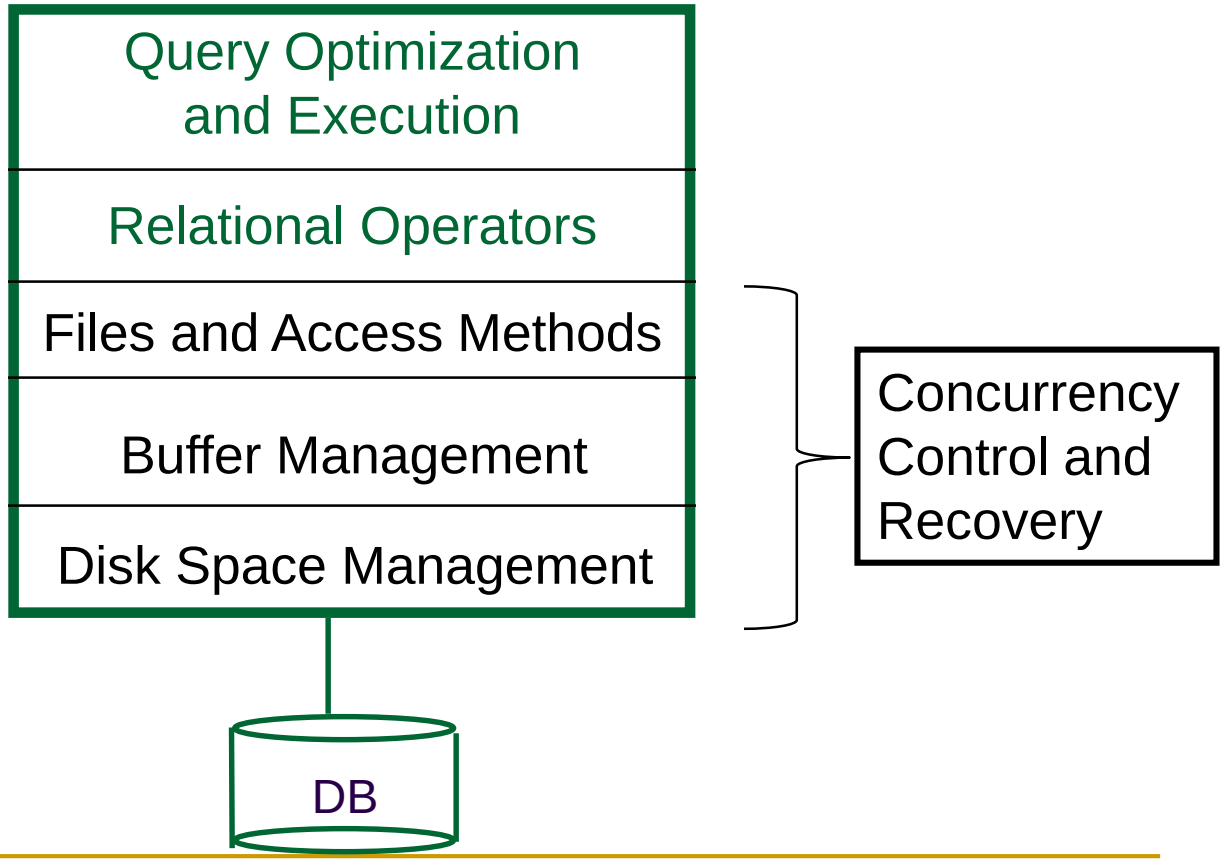
Storing Data: Disks and Files

SUN YAT-SEN UNIVERSITY

Block diagram of a DBMS



Disks

- DBMS stores information on disks.
  - Tapes are also used.
- Major implications for DBMS design!
  - READ: transfer data from disk to main memory (RAM).
  - WRITE: transfer data from RAM to disk.
  - Both high-cost(高开销)
    - Can/should plan carefully!

Why Not Store Everything in Main Memory?

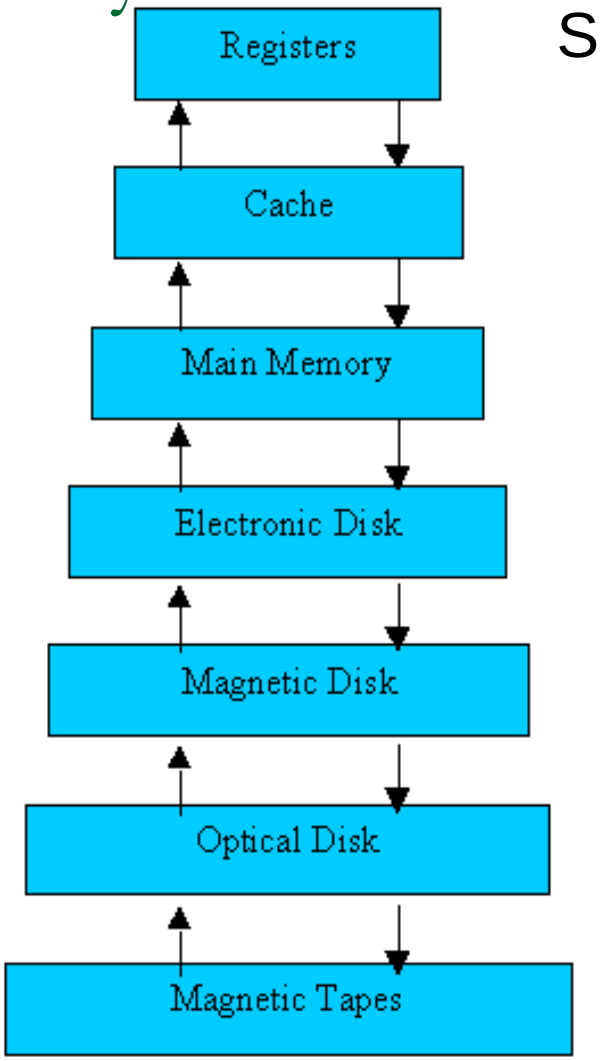
- *Costs too much(成本太高).* For ~$1000, PC Connection will sell you either
  - ~80GB of RAM (unrealistic)
  - ~400GB of Flash USB keys (unrealistic)
  - ~180GB of Flash solid-state disk (serious)
  - ~7.7TB of disk (serious)
- *Main memory is volatile(易失的).*
  - Want data to persist between runs. (Obviously!)

The Storage Hierarchy

–Main memory (RAM) for currently used data.

–Disk for main database (secondary storage-二级存储介质).

–Tapes for archive (tertiary storage-三级存储介质).



Smaller, Faster

Bigger, Slower

Source: Operating Systems Concepts 5th Edition

Disks

- Still the secondary storage device of choice.
- Main advantage over tape:
  - *random access* vs. *sequential*.
- 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!

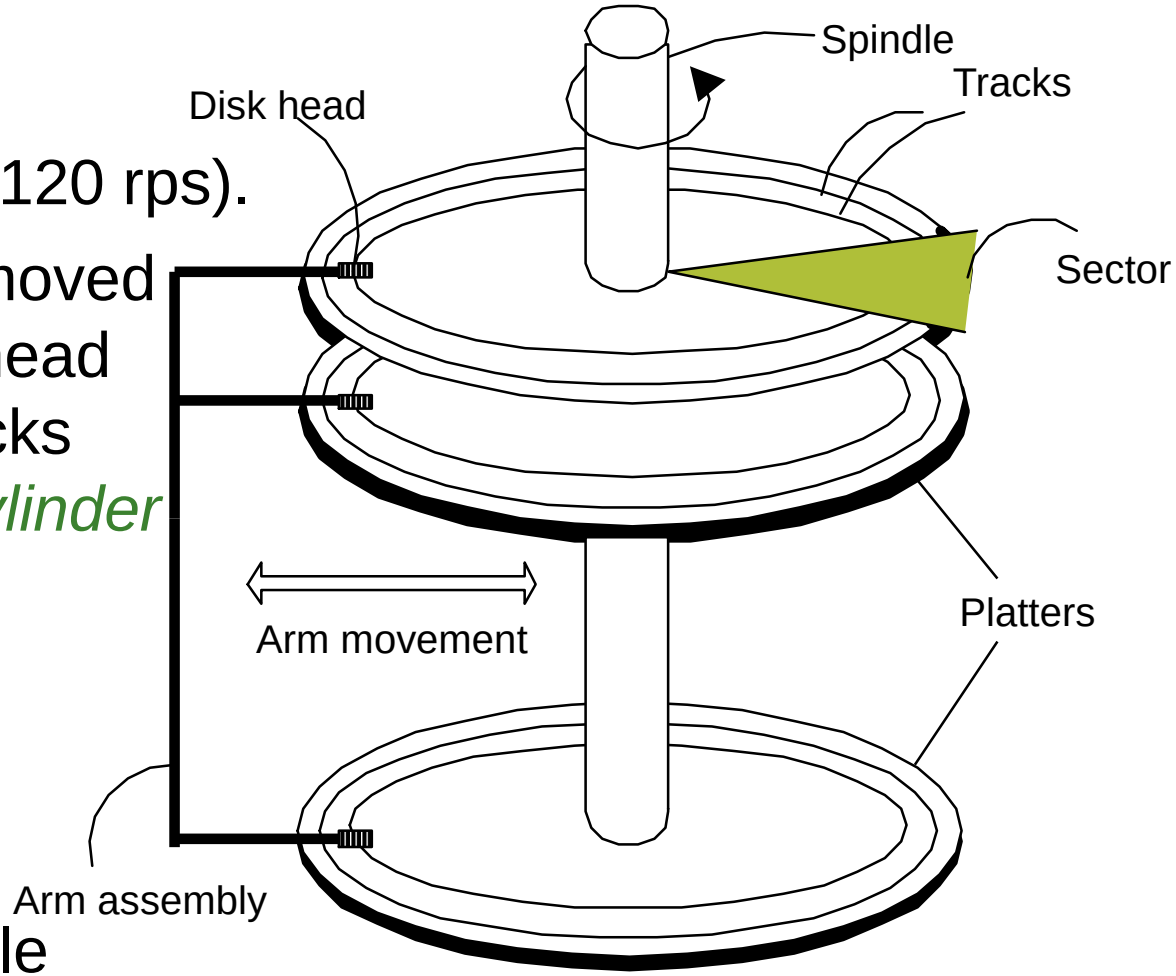
Components of a Disk

The platters spin (say, 120 rps).

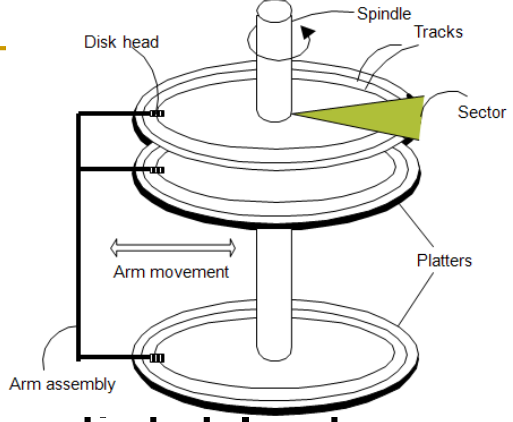
The arm assembly is moved in or out to position a head on a desired track. Tracks under heads make a *cylinder* (imaginary!).

Only one head reads/writes at any one time.

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

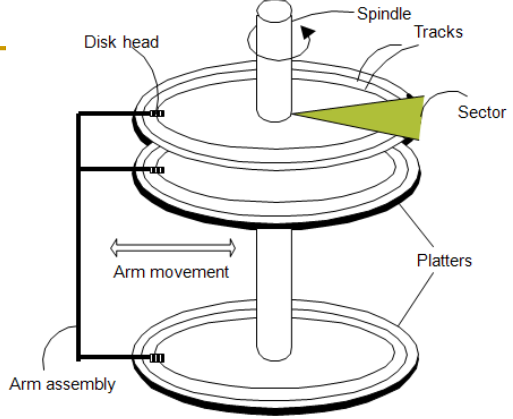


Accessing a Disk Page



- 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?

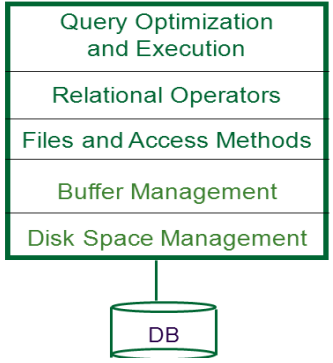
Arranging Pages on Disk



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

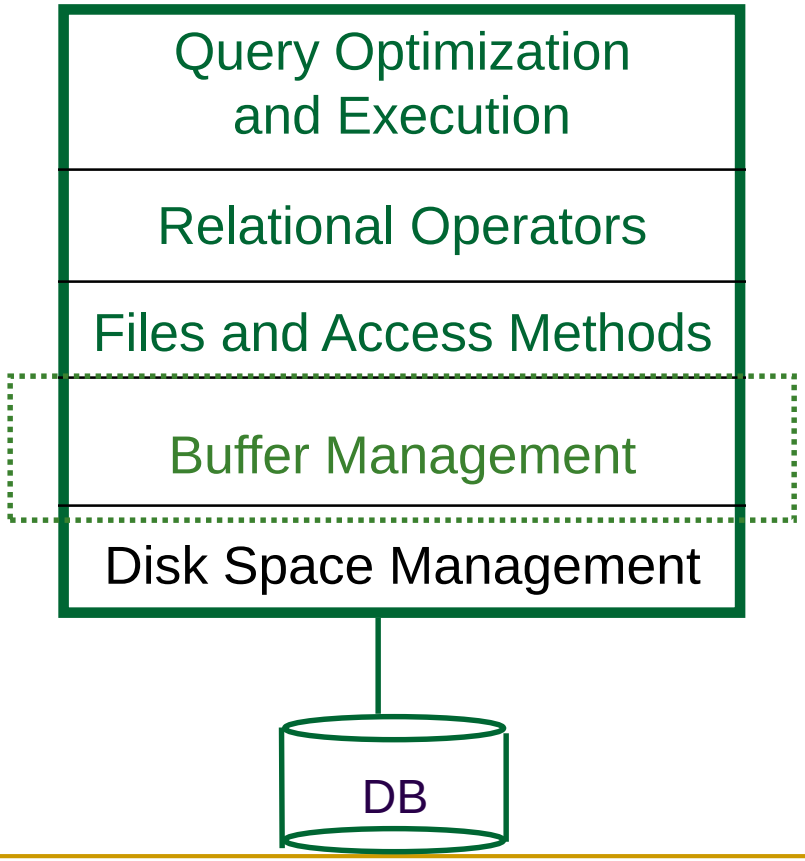
- Blocks in a file should be arranged sequentially （连续） on disk (by `next'), to minimize seek and rotational delay.
  - For a sequential scan, *pre-fetching (预读取)* several pages at a time is a big win!

Disk Space Management



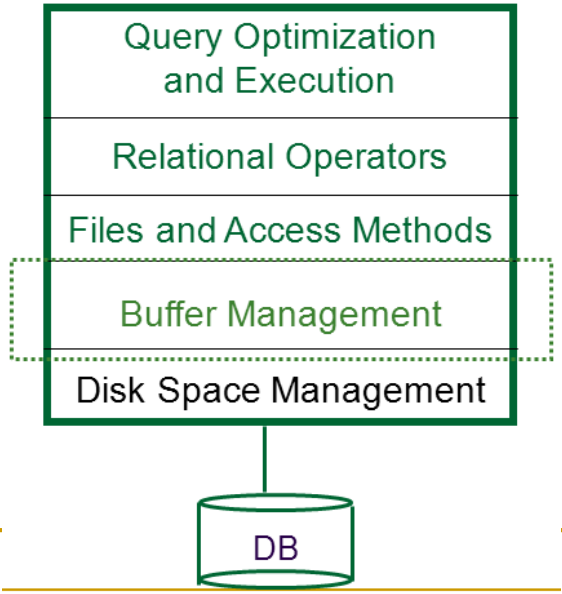
- Lowest layer of DBMS, manages space on disk
- 接口: Higher levels call upon this layer to:
  - allocate/de-allocate a page-分配和释放磁盘页
  - read/write a page
- 期望性能:Request for a *sequence* of pages best satisfied by pages stored sequentially on disk!
  - Responsibility of disk space manager.
  - Higher 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

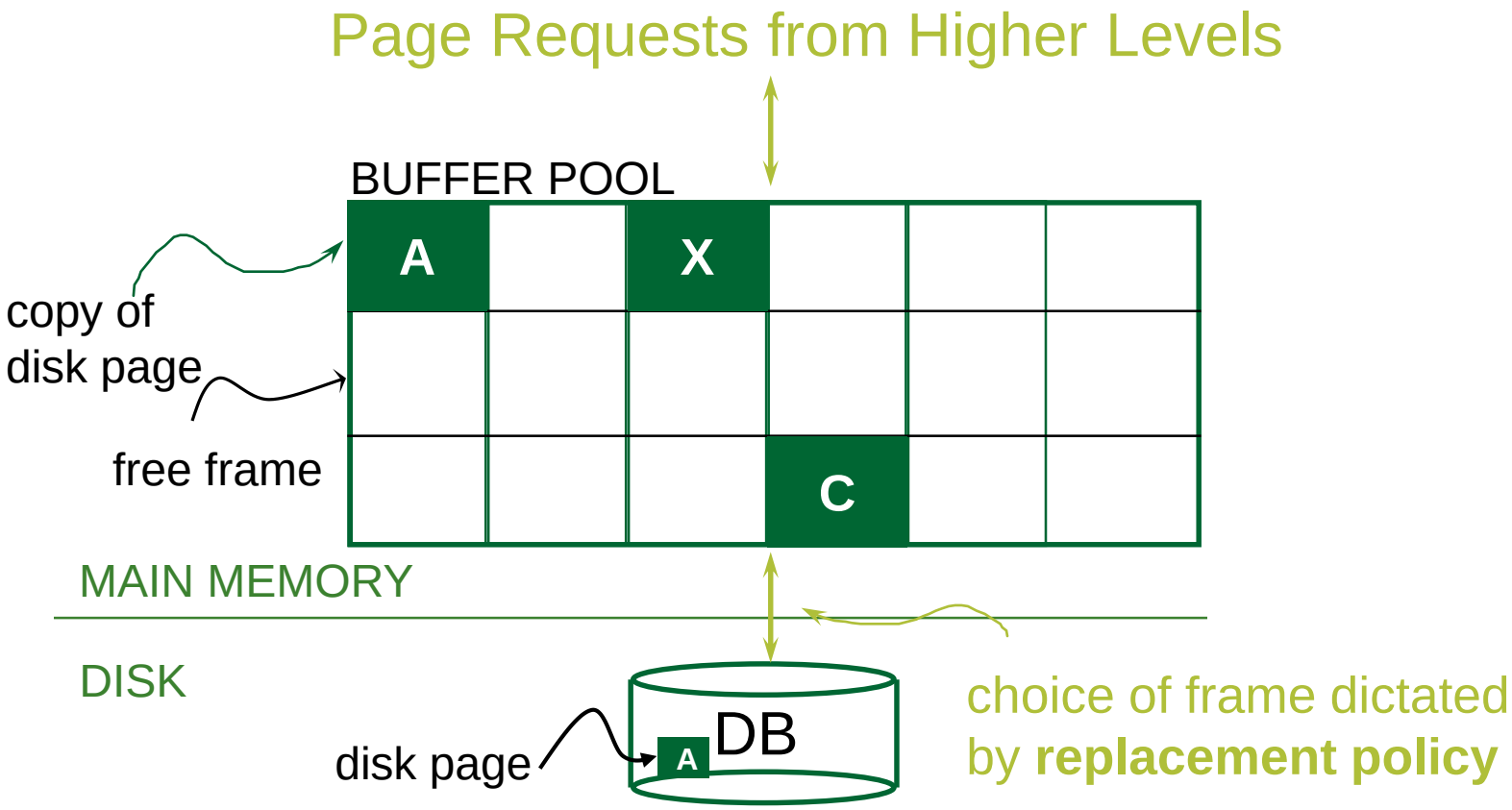

Query Optimization
and Execution
Relational Operators
Files and Access Methods
Buffer Management
Disk Space Management
DB

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*

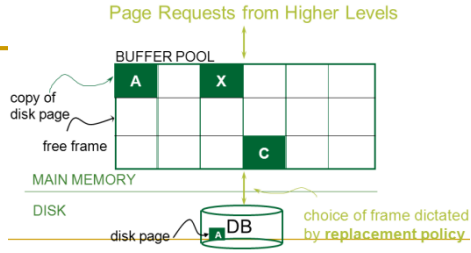


Buffer Management in a DBMS
缓冲区管理—how?



划分**帧**→（空**闲帧**/内容帧） → 副本→ **请**求→替**换**策略

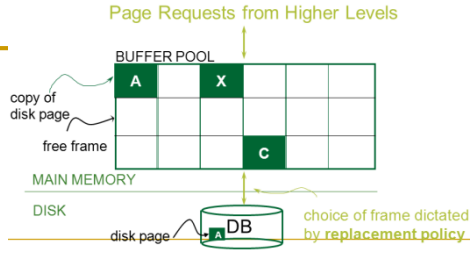
When a Page is Requested ...



- 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

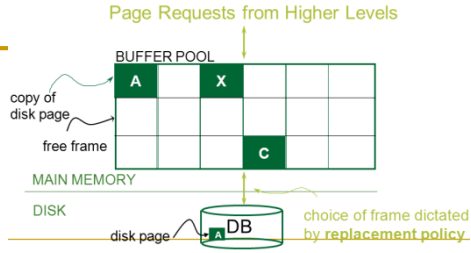


- Requestor of page must eventually:
  1. *unpin* it -- 解钉　pin_count--
  2. indicate whether page was modified via *dirty* bit.

  <frame#, pageid, pin_count, dirty>

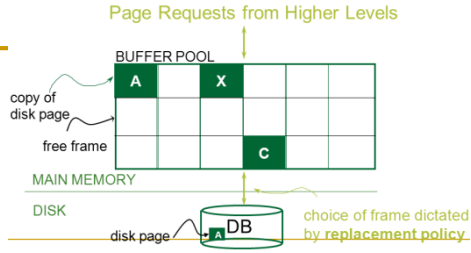
- Page in pool may be requested many times,
  - a *pin _count* is used.
  - To pin a page(钉住页): pin_count++
  - A page is a candidate for replacement iff *pin_count* == 0 (*"unpinned"*)

Buffer Replacement Policy



- Frame is chosen for replacement by a *replacement policy(替换策略):*
  - Least-recently-used (LRU), MRU-最近最常使用, Clock, …
- Policy can have big impact on #I/O's;
  - Depends on the *access pattern*.访问模式

LRU Replacement Policy
最近最少使用策略



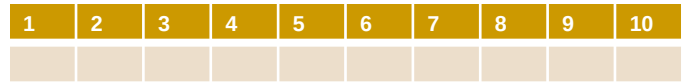
<frame#, pageid, pin_count, dirty>

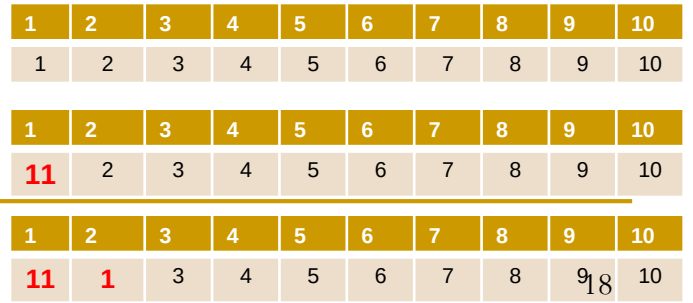
- *Least Recently Used (LRU)*
  - track time each frame last *unpinned* (end of use),
  - by using a queue of pointers to frames with *pin_count*= 0.
  - replace the frame which has the earliest unpinned time
- Very common policy: intuitive and simple
  - Works well for repeated accesses to popular pages

Problem of LRU

- *Problem: Sequential flooding* --连续扩散
  - LRU + repeated sequential scans.(多次顺序扫描)

- An illustrative situation:
  - suppose a buffer pool has 10 frames,
  - and the file to be scanned has 11 pages;
  - then using LRU, every scan of the file will result in reading every page of the file.





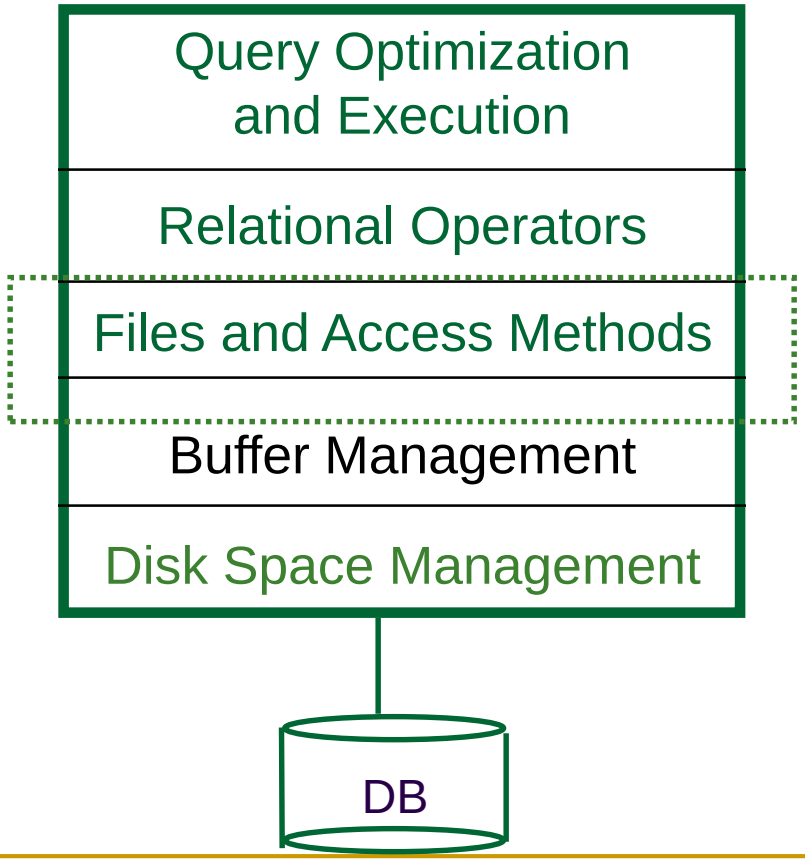
解决方案：MRU-最近最常使用

DBMS vs. OS File System

OS does disk space & buffer mgmt: why not let OS manage these tasks?

- A DBMS can often predict(预测) page reference patterns more accurately than an OS.
  - Most page references are generated by higher-level operations such as sequential scan.
  - adjust *replacement policy,* and pre-fetch pages based on page reference patterns in typical DB operations.
- A DBMS also requires the ability to explicitly *force* a page to disk. --强制写
  - For realizing Write-Ahead Log protocol

Context


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

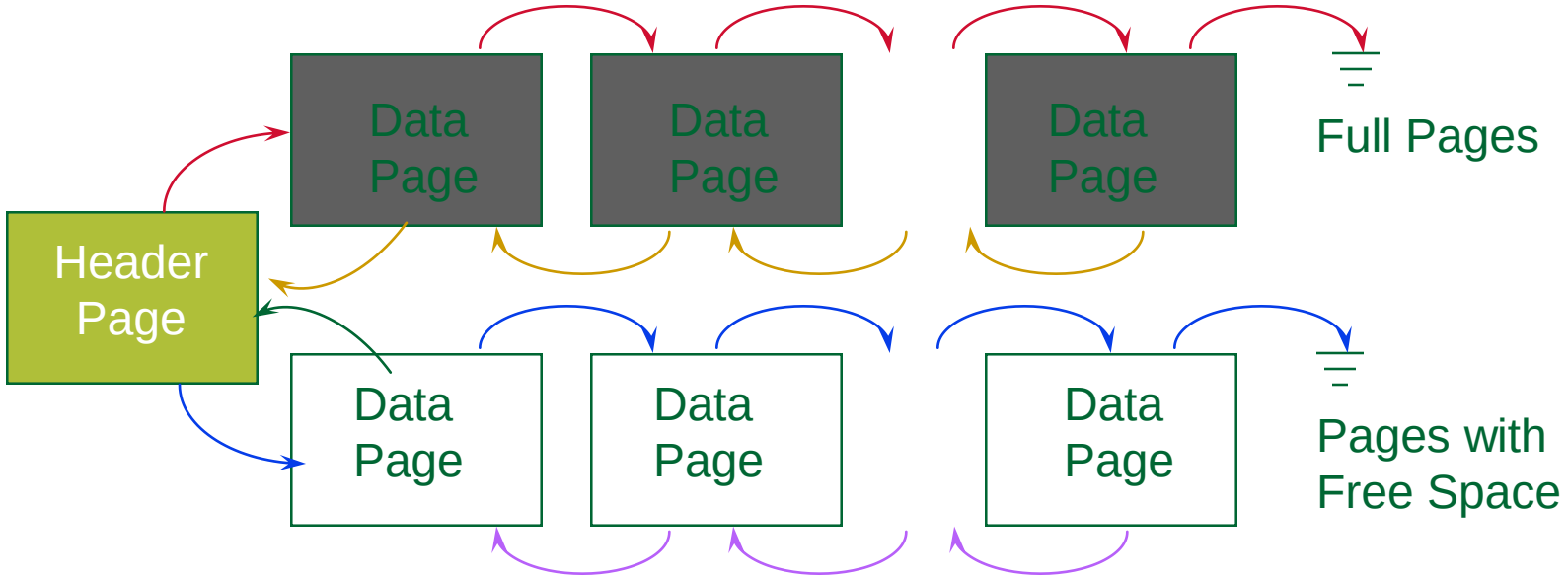
Files of Records (记录文件)

- 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

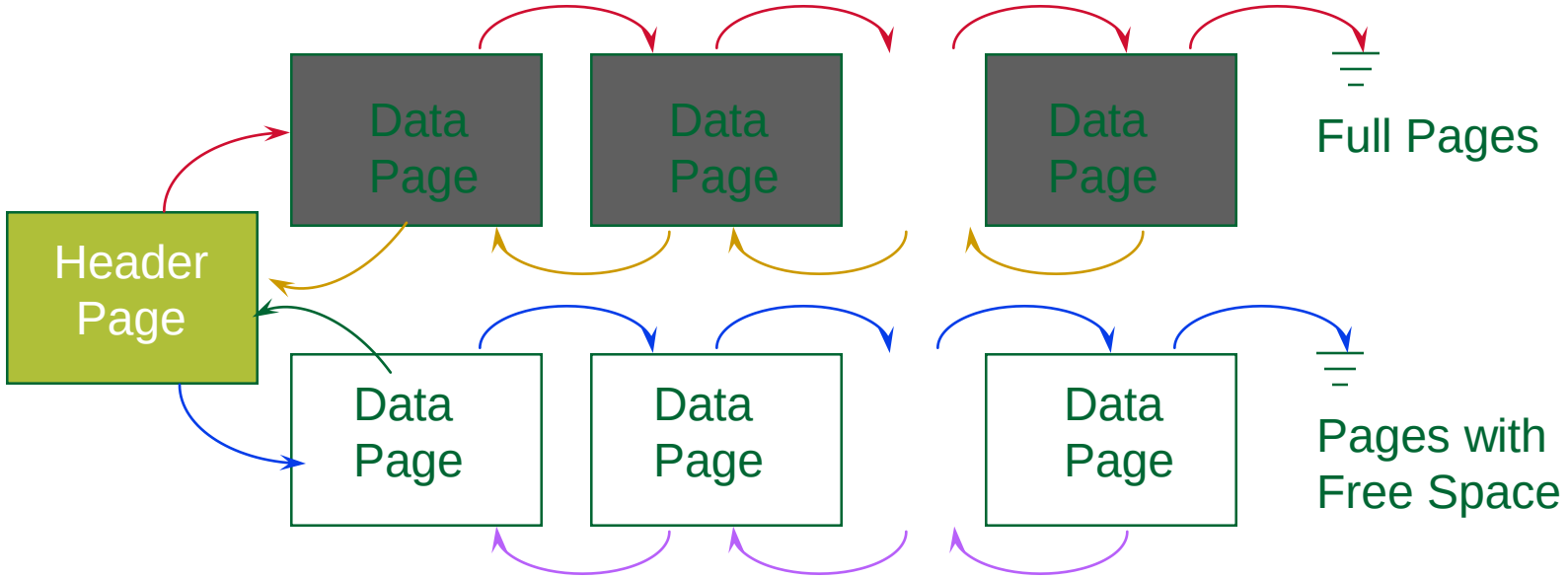
- Collection of records in no particular order.
  - As file shrinks/grows, disk pages (de)allocated
- To support record level operations, we must:
  - keep track of the *pages* in a file
  - keep track of *free space* on pages
  - keep track of the *records* on a page
- There are many alternatives for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

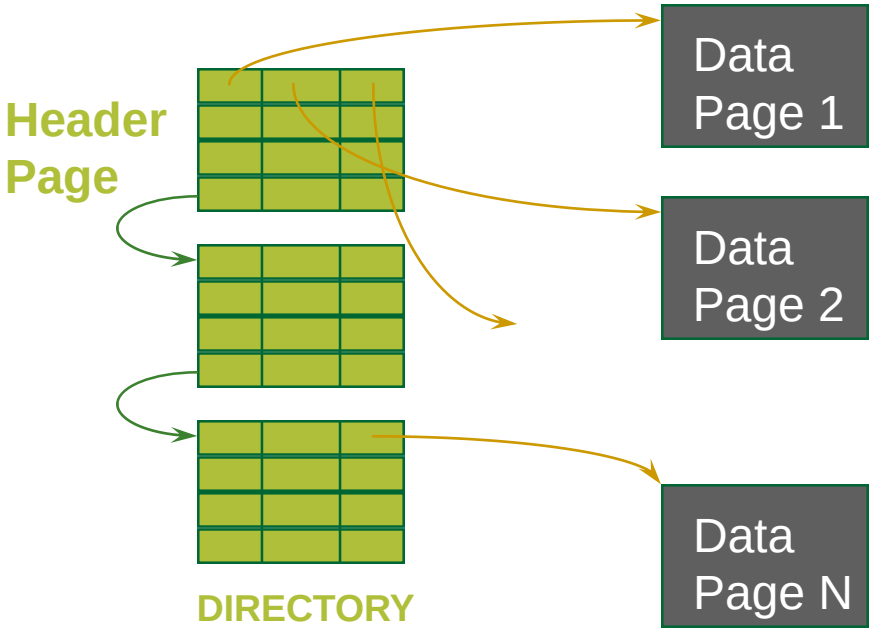
Heap File Implemented as a List (Cont.)



- One disadvantage
  - Virtually all pages will be on the free list if records are of variable length,

    i.e., every page may have some free bytes if we like to keep each record in a single page.

Heap File Using a Page Directory(页目录)

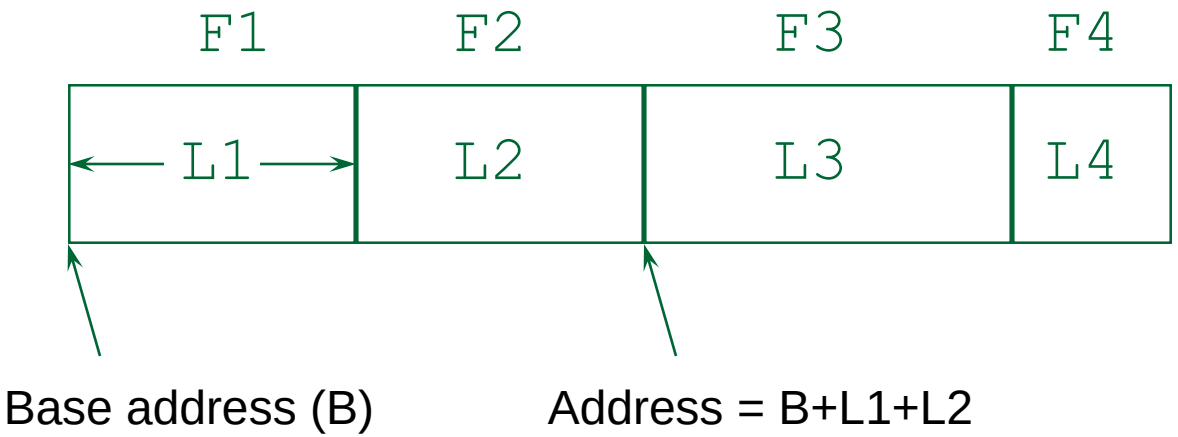


- The directory is itself a collection of pages;
  - each page can hold several entries.
  - The entry (目录项) for a page can include the number of free bytes on the page. <pointer,free_number>
- To insert a record, we can search the directory to determine which page has enough space to hold the record.

Indexes (索引， a sneak preview)

- A Heap file allows us to retrieve records:
  - by specifying the *rid* (record id), or
  - by scanning all records sequentially
- Sometimes, we want to retrieve records by specifying the *values in one or more fields*, e.g.,
  - Find all students in the “CS” department
  - Find all students with a gpa > 3
- Indexes are file structures that enable us to answer such value-based queries efficiently.

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

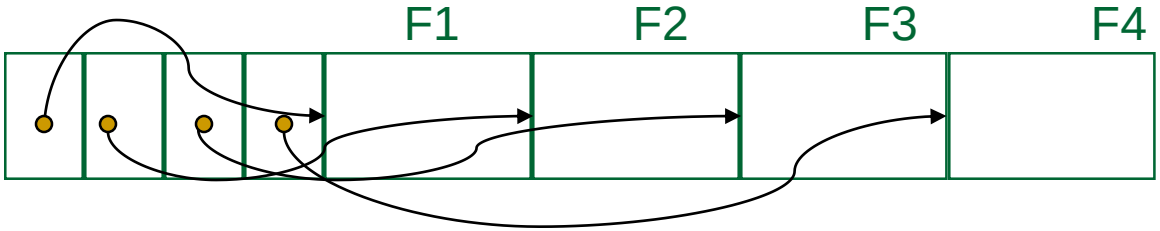


- Information about field types same for all records in a file; stored in *system catalogs.*
- Finding *i'th* field done via arithmetic.

Record Formats: Variable Length(变长记录)

- Two alternative formats (# fields is fixed):

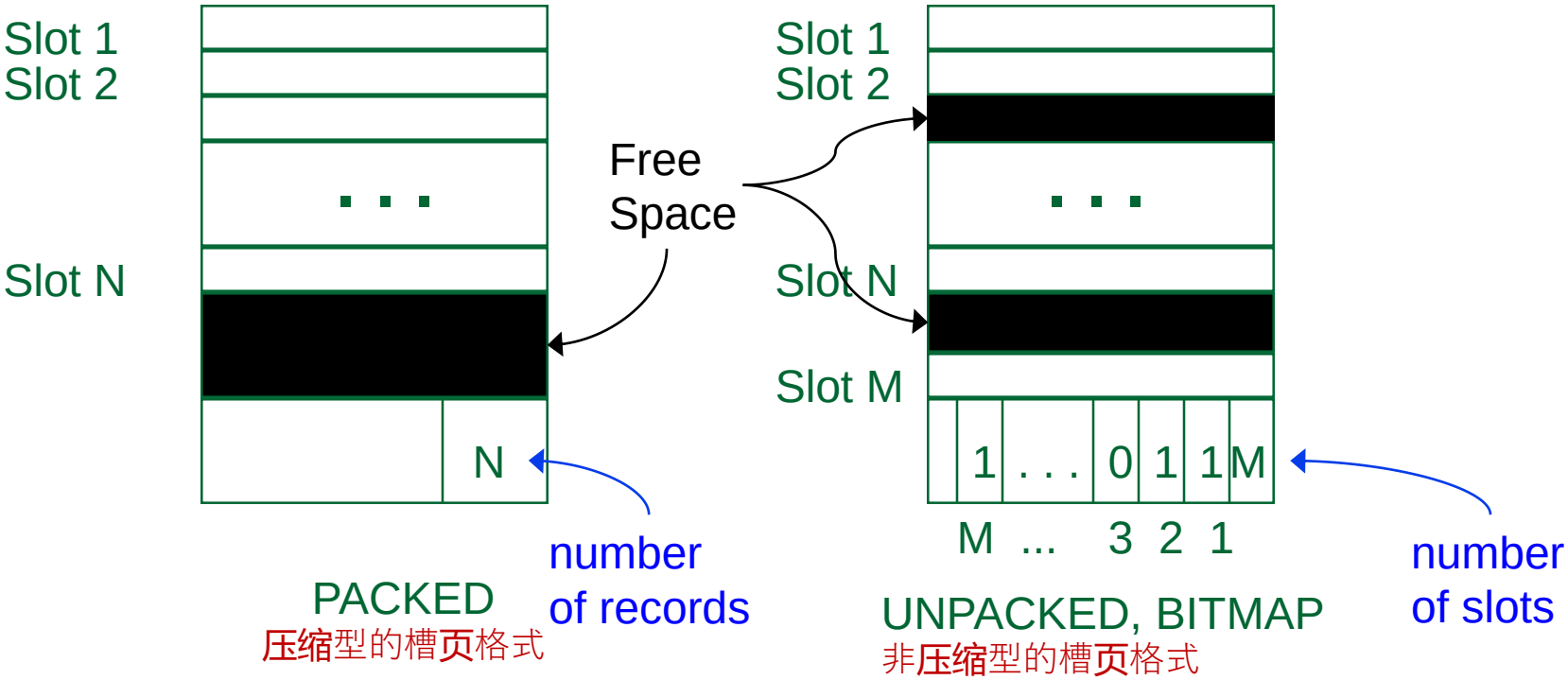
Fields Delimited by Special Symbols



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

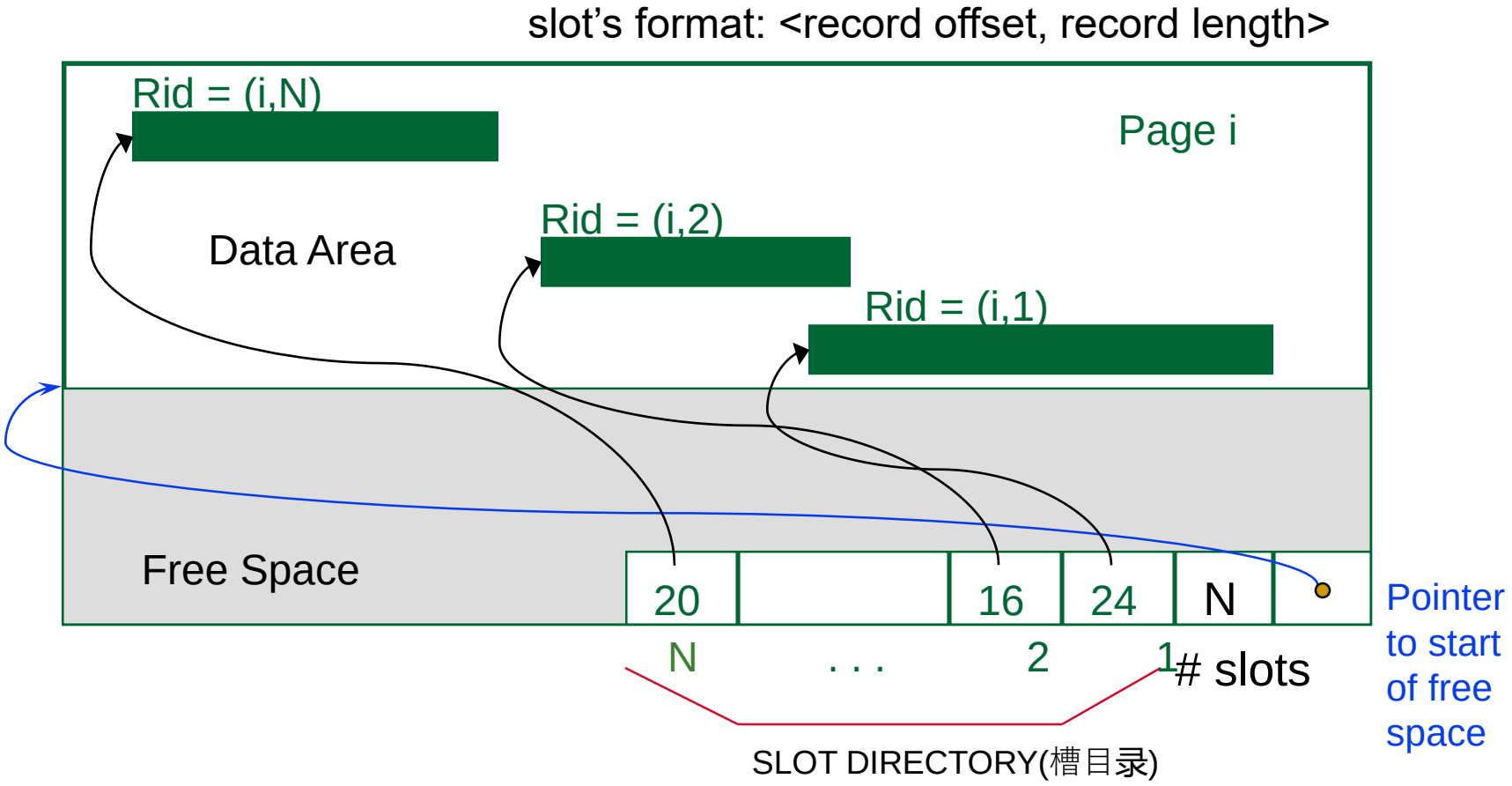


Record id = <page id, slot #>.

In first alternative,

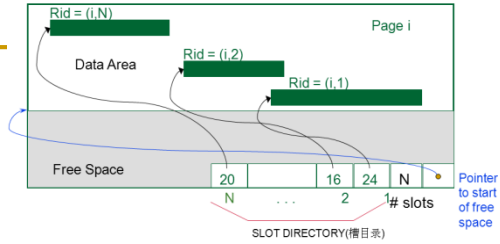
- *moving records for free space management changes rid;*
- *may not be acceptable.*

Page Formats: Variable Length Records

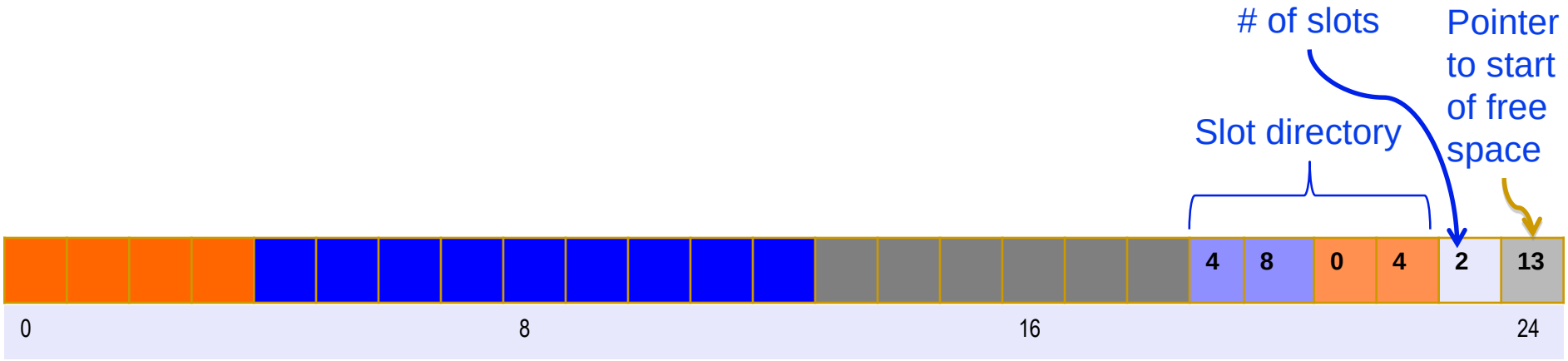


Can move records on page without changing rid; so, attractive for fixed-length records too.

Slotted page: a detailed view



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



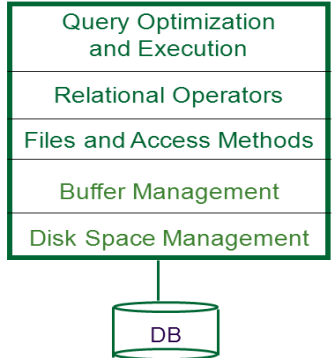
System Catalogs-系统目录

- For each relation:
  - name, file location, file structure (e.g., Heap file)
  - attribute name and type, for each attribute
  - index name, for each index
  - integrity constraints
- For each index:
  - structure (e.g., B+ tree) and search key fields
- For each view:
  - view name and definition
- Plus statistics, authorization, buffer pool size, etc.

Catalogs are themselves stored as relations!

Attr_Cat(attr_name, rel_name, type, position)

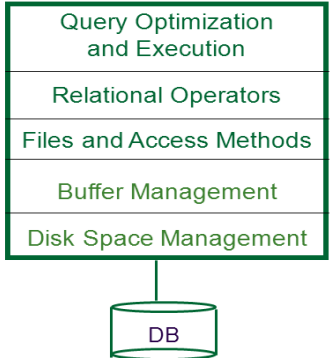
attr_name	rel_name	type	position
attr_name	Attribute_Cat	string	1
rel_name	Attribute_Cat	string	2
type	Attribute_Cat	string	3
position	Attribute_Cat	integer	4
sid	Students	string	1
name	Students	string	2
login	Students	string	3
age	Students	integer	4
gpa	Students	real	5
fid	Faculty	string	1
fname	Faculty	string	2
sal	Faculty	real	3



Summary

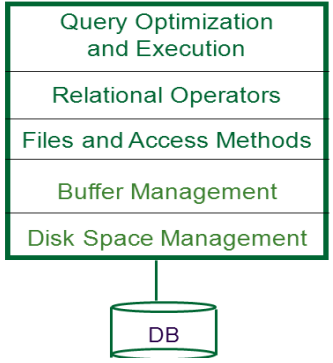
- Disks provide cheap, non-volatile storage.
  - Better random access than tape, worse than RAM
  - Arrange data to minimize *seek* and *rotation* delays.
    - Depends on workload!
- Buffer manager brings pages into RAM.
  - Page pinned in RAM until released by requestor.
  - Dirty pages written to disk when frame replaced (sometime after requestor unpins the page).
  - Choice of frame to replace based on *replacement policy.*
  - Tries to *pre-fetch* several pages at a time.

Summary (Contd.)



- DBMS vs. OS File Support
  - DBMS needs non-default features
  - Careful timing of writes, control over prefetch
- Variable length record format
  - Direct access to i'th field and null values.
- Slotted page format
  - Variable length records and intra-page reorg

Summary (Contd.)



- DBMS "File" tracks collection of pages, records within each.
  - Pages with free space identified using linked list or directory structure
- Indexes support efficient retrieval of records based on the values in some fields.
- Catalog relations store information about relations, indexes and views.
- 要求: 深刻理解 *Record id = <page id, slot #>*