STORING DATA: DISK AND FILES

CS 564- Fall 2016

ACKs: *Dan Suciu, Jignesh Patel, AnHai Doan*

MANAGING DISK SPACE

- The disk space is organized into files
- Files are made up of pages
- Pages contain records

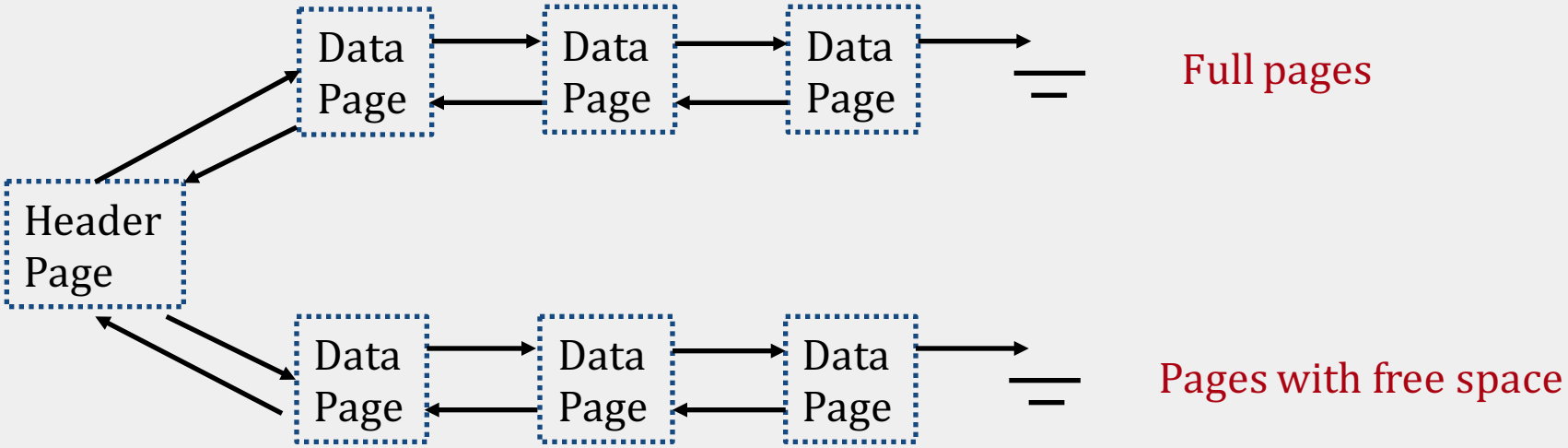
File Organization

UNORDERED (HEAP) FILES

- Contains the records in no particular order
- As file grows/shrinks, disk pages are allocated/deallocated
- To support record level operations, we must keep track of:
  - the pages in a file: page id (*pid*)
  - free space on pages
  - the records on a page: record id (*rid*)

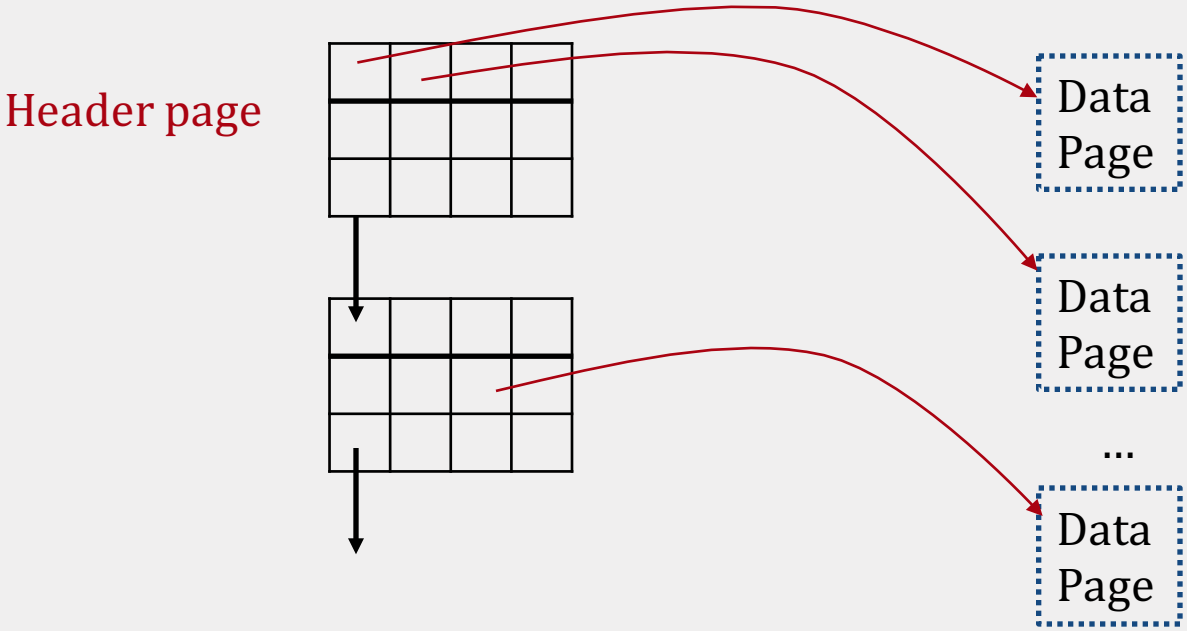
Heap File As Linked List

- (heap file name, header page id) stored somewhere
- Each page has 2 pointers + data
- Pages in the free space list have “some” free space



Heap File As Page Directory

- Each entry for a page keeps track of:
  - is the page free or full?
  - how many free bytes are?
- We can now locate pages for new tuples faster!



Page Organization

FILES OF RECORDS

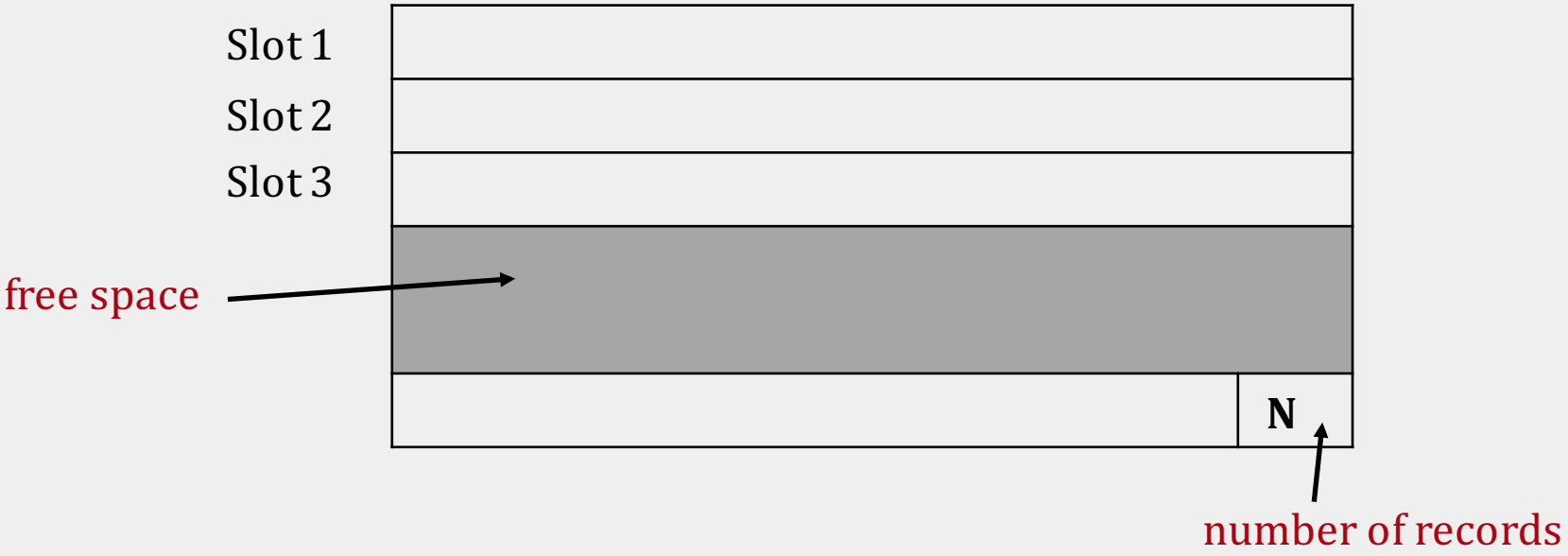
- Page or block is OK for I/O, but higher levels operate on records, and files of records
- File operations:
  - insert/delete/modify record
  - read a record (specified using the record id)
  - scan all records (possibly with some conditions on the records to be retrieved)

PAGE FORMATS

- A page is collection of records
- Slotted page format
  - A page is a collection of slots
  - Each slot contains a record
- ***rid*** = *<page id, slot number>*
- There are many slotted page organizations
- We need to have support for:
  - search, insert, delete records on a page

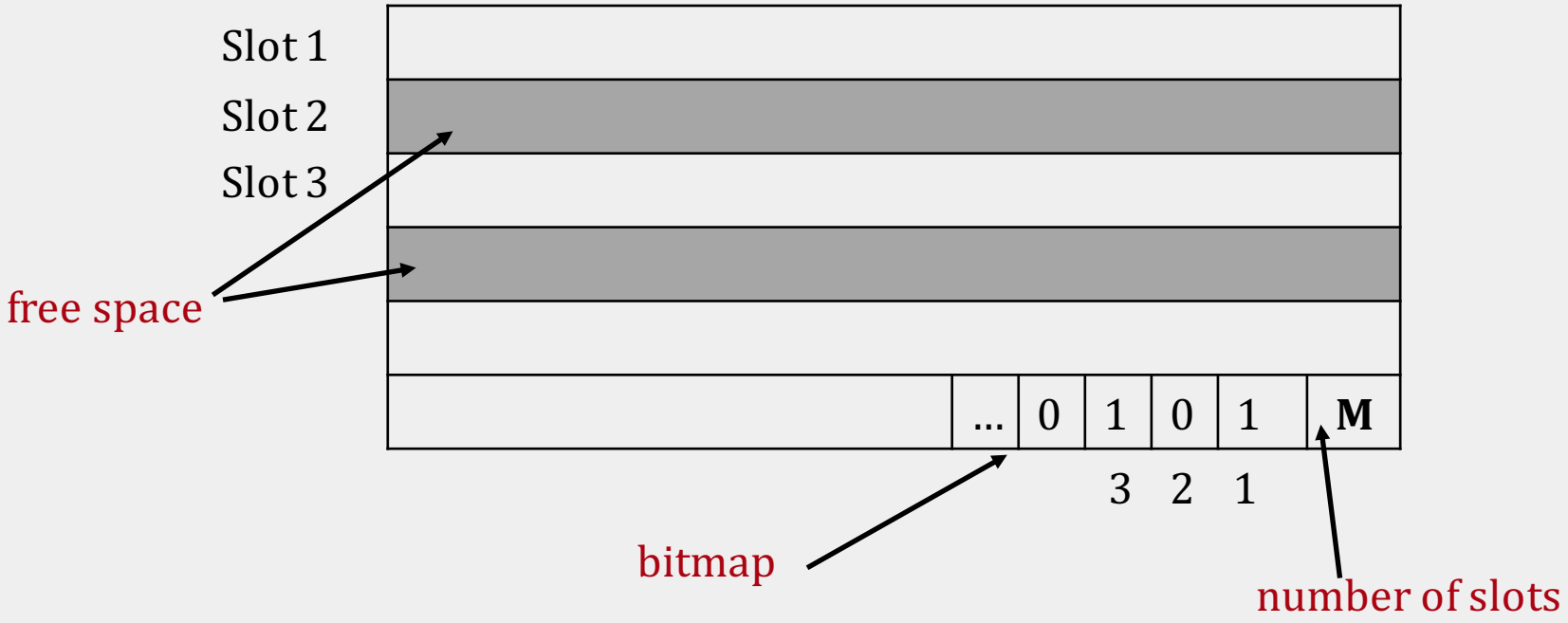
FIXED LENGTH RECORDS (1)

- *packed* organization: *N* records are always stored in the first *N* slots
- problem when there are references to records!

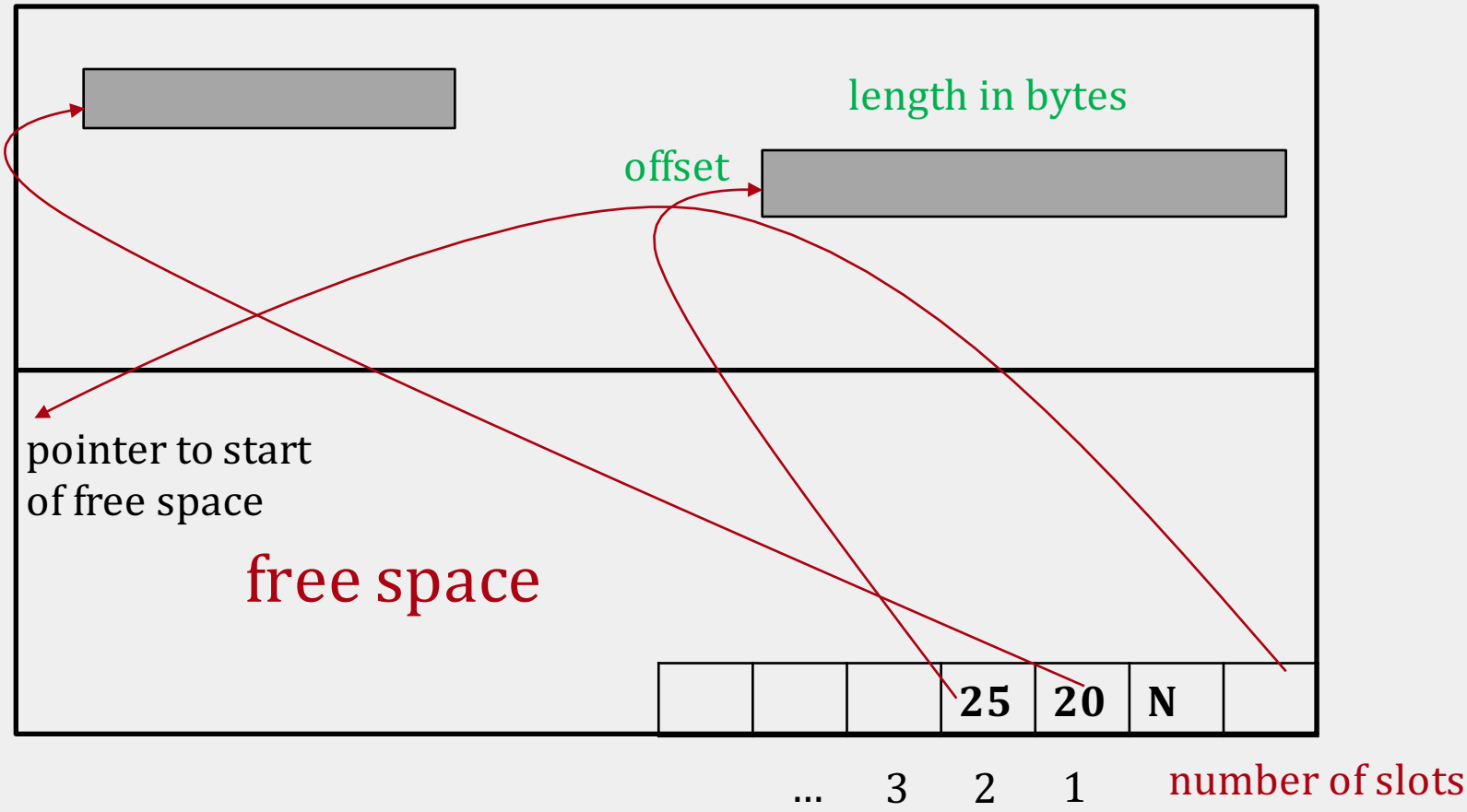


FIXED LENGTH RECORDS (2)

- *unpacked* organization: use a bitmap to locate records in the page



VARIABLE LENGTH RECORDS



VARIABLE LENGTH RECORDS

- Deletion:
  - offset is set to -1
- Insertion:
  - use any available slot
  - if no space is available, reorganize
- *rid* remains unchanged when we move the record (since it is defined by the slot number)

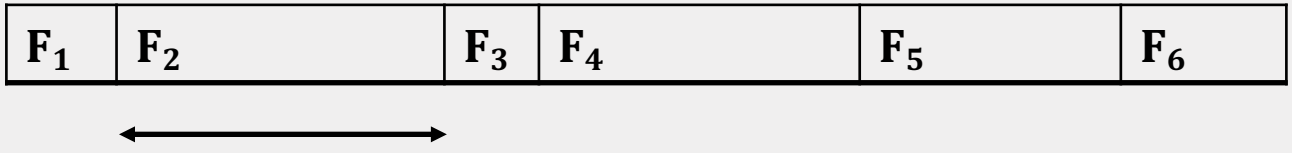
RECORD FORMAT

- How do we organize the field within a record?
  - fixed length
  - variable length

- Information common to all records of a given type is kept in the system catalog:
  - number of fields
  - field type

RECORD FORMAT: FIXED LENGTH

- All records have the same length and same number of fields
- The address of any field can be computed from info in the system catalog!



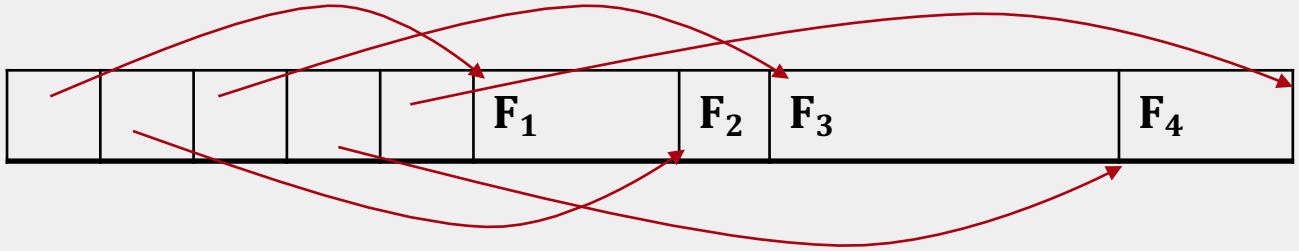
L_2 = length of field F_2

RECORD FORMAT: VARIABLE LENGTH (1)

- store fields consecutively
- use delimiters to denote the end of a field
- need a scan of the whole record to locate a field

RECORD FORMAT: VARIABLE LENGTH (2)

- store fields consecutively
- use an array of integer offsets in the beginning



BONUS: COLUMN STORES

- Consider a table:
  - **Foo** (a INTEGER, b INTEGER, c VARCHAR(255))
- and the query:
  - `SELECT a FROM Foo WHERE a > 10`

- What could be the problem when we read using the previous record formats?

BONUS: COLUMN STORES

- We can instead store data vertically !
- Each column of a relation is stored in a different file (and can be compressed as well)

1234	45	Here goes a very long sentence 1
4657	2	Here goes a very long sentence 2
3578	45	Here goes a very long sentence 3

row-store

column-store

1234
4657
3578

45
2
45

Here goes a very long sentence 1
Here goes a very long sentence 2
Here goes a very long sentence 3