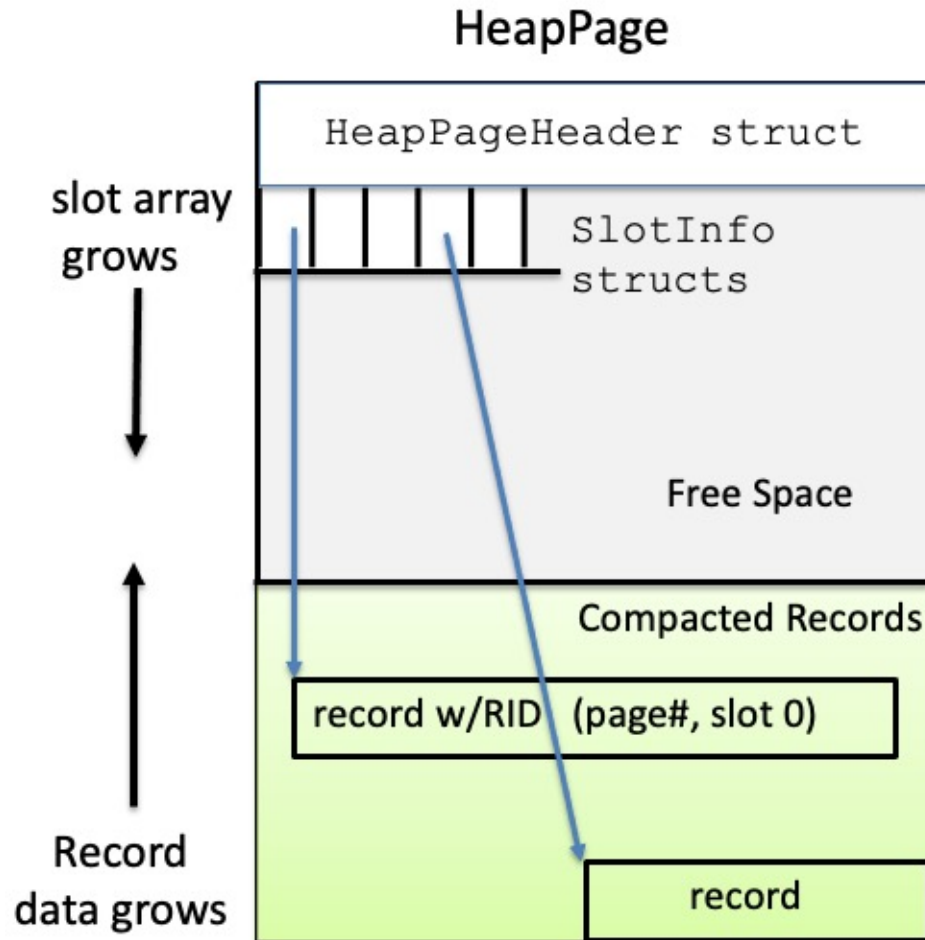


SwatDB HeapPage Lab: Using type-recasting to mapping Heap Page structure on top of raw Page space (char/byte array)

Tia Newhall and Ameet Soni
Swarthmore College

SwatDB HeapPage Structure



for Variable-length records

PAGE_SIZE chunk of space:

- A PAGE_SIZE array of bytes/chars
- Page is unit of storage in the system.
- Every type of Page is exactly the same size. Heap Page, Index Page, Leaf Page, ...

HeapPage is derived from Page class

- Page class defines raw page-size space
- HeapPage maps structure onto raw data

```
class HeapPage : public Page { ... }
```

```
class Page {
```

```
    ...
```

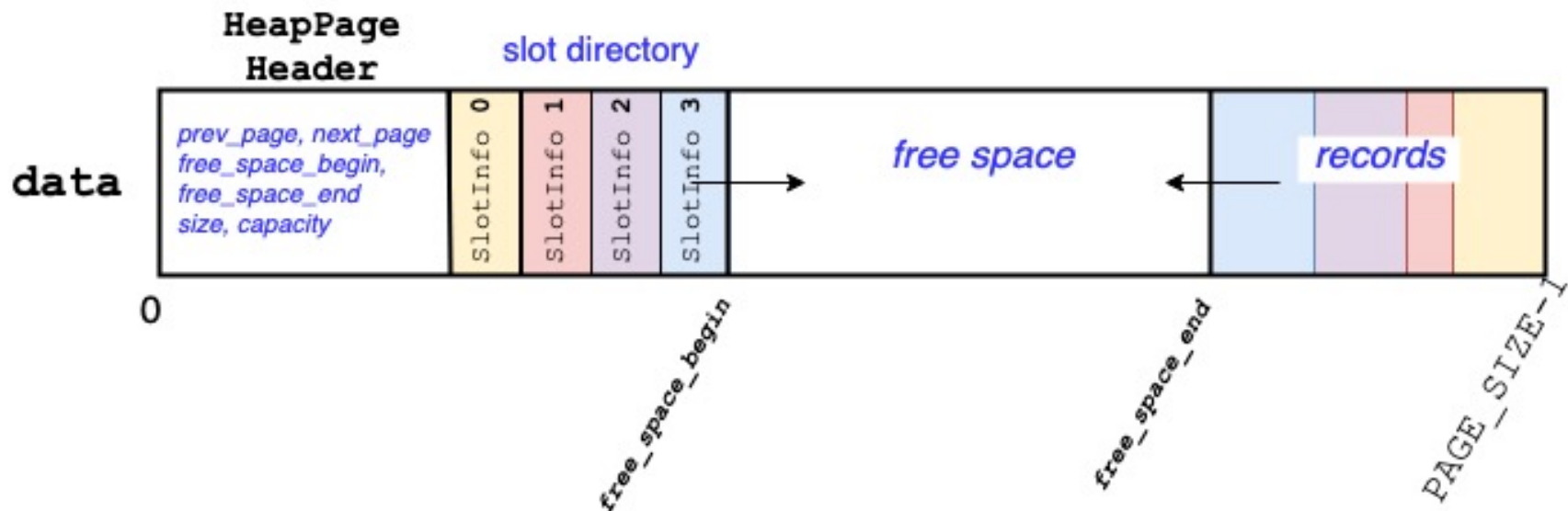
```
    private:
```

```
        char data[PAGE_SIZE]; // raw page-size data
```

```
};
```



HeapPage maps heap page-specific structure onto the raw page-size space defined by Page class
a structured type-specific view on top of array of bytes (char)



Map Heap Page header information on top of the first bytes of the page data

```
struct HeapPageHeader {  
    PageNum prev_page;  
    PageNum next_page;  
    std::unit32_t free_space_begin;  
    std::unit32_t free_space_end;  
    std::unit32_t size; // slot dir  
    std::unit32_t capacity; // slot dir  
};
```

HeapPage Header

data



Map HeapPageHeader information on top of the first bytes of the page data



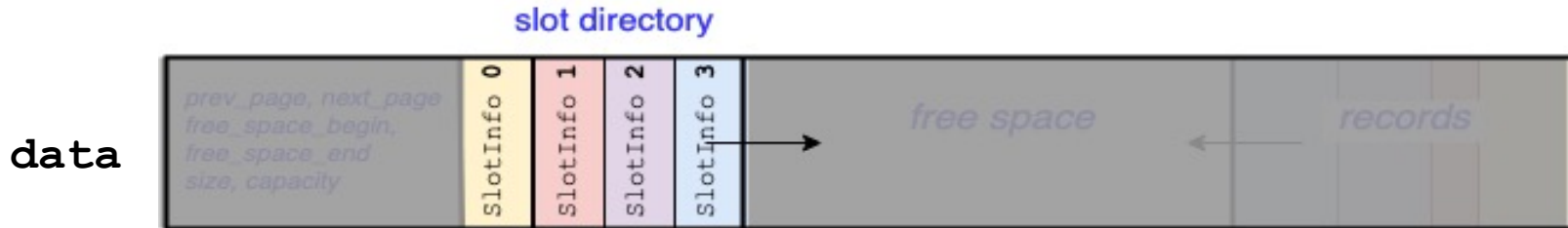
Map: re-cast memory address as type you want to map onto it
(instead of an address of char, re-cast as address of HeapPageHeader)

```
HeapPageHeader* HeapPage::_getPageHeader() {  
    // re-cast data as a pointer to a HeapPageHeader  
    return (HeapPageHeader*) data; // &data[0]  
}
```

Call: `HeapPageHeader *hdr = _getPageHeader();`

Use: `hdr->size = 10; //sets offset of size field into data array to 10`

Map slot directory entries after header on data



Map: re-cast memory address as type you want to map onto it

```
SlotInfo *HeapPage::_getSlotDirectory() {  
    // re-cast address in data after HeapPageHeader as  
    // pointer to SlotInfo struct (base address slot dir)  
    return (SlotInfo *) (data + sizeof(HeapPageHeader));  
}
```

`data + sizeof(HeapPageHeader) :`

Pointer Arithmetic: adds to address value based on type
computes the address at `sizeof(HeapPageHeader)` number of
bytes/chars from the base address of data.

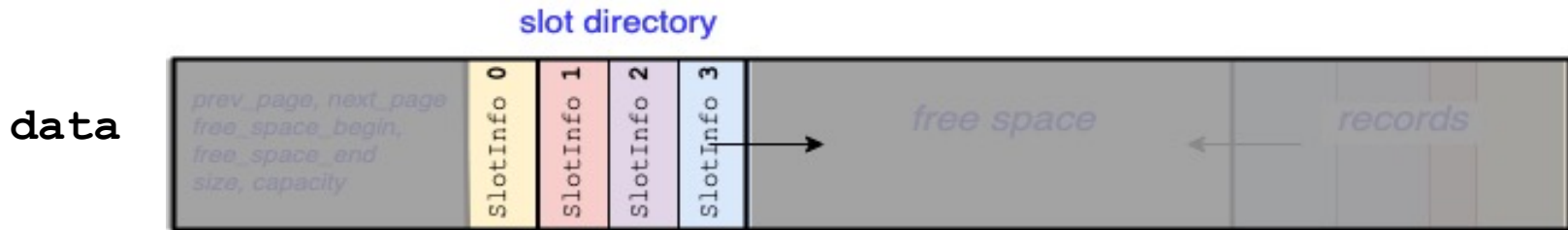
ex. `data + 3` is `(&data[3])`

ex. `x = sizeof(HeapPageHeader);`

`data + x` (is `&data[x]`)

(note: data is a char array, so sizeof giving number of bytes works with char * arith)

Map slot directory entries after header on data

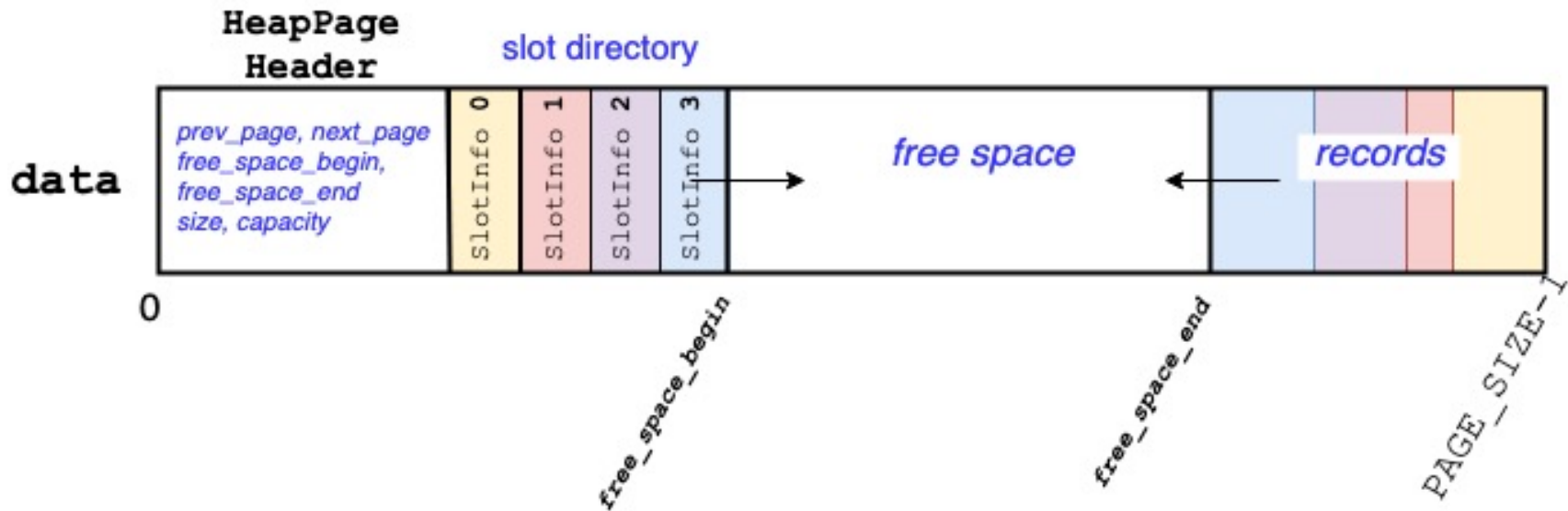


Map: re-cast memory address as type you want to map onto it

```
SlotInfo *HeapPage::_getSlotDirectory() {  
    // re-cast address in data after HeapPageHeader as  
    // pointer to SlotInfo struct (base address slot dir)  
    return (SlotInfo *) (data + sizeof(HeapPageHeader));  
}
```

Call: `SlotInfo *slot_dir = _getSlotDirecotory();`

Use: `slot_dir[3].offset = 400; // set offset field of entry 3 in slot array`



```
HeapPageHeader *hdr = _getPageHeader();
hdr->size = 10; // modify size of slot directory
hdr->free_space_end -= 24; // modify free space end on page
```

```
SlotInfo *slot_dir = _getSlotDirectory();
slot_dir[3].offset=400; // set the offset field of slot directory entry 3
```

`slot_dir` is base address of an array of SlotInfo structs (slot directory)
`slot_dir[3].offset` is offset field in the 3rd bucket of the `slot_dir`
 (at some address from the start of data array)