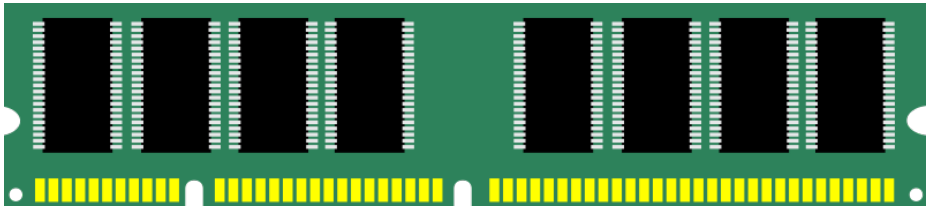
Checkpoint 2, Physical Layouts

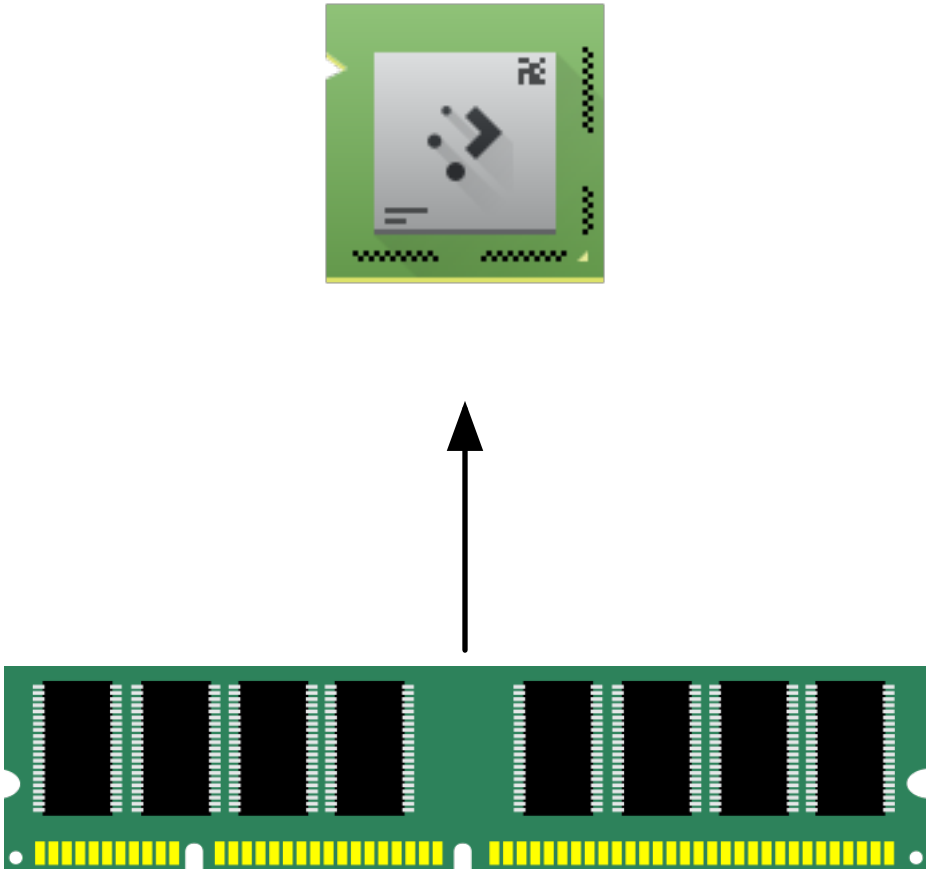
CSE 4/562: Database Systems | Lecture 6

DB. Sys.: T.C.B.: Ch. 13.1-13.7, 15.7, 16.7

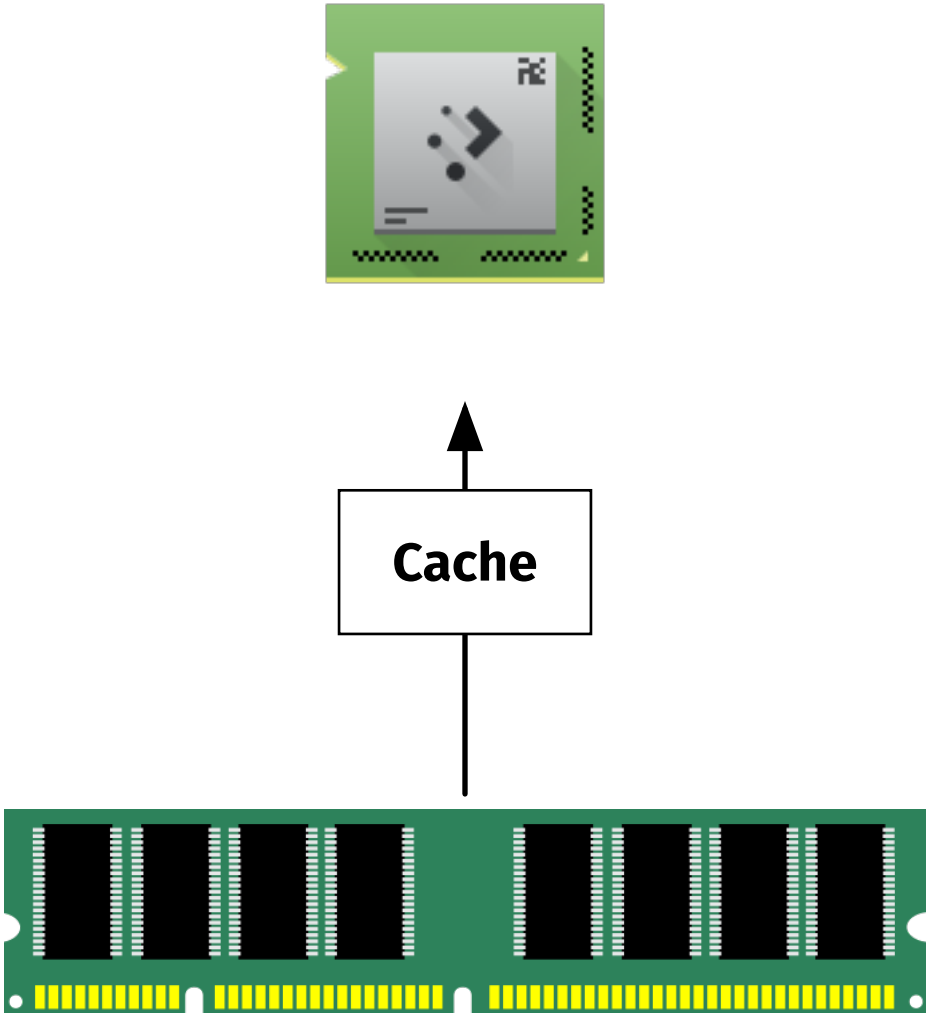
Memory Access



Memory Access

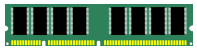


Memory Access

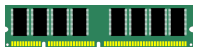


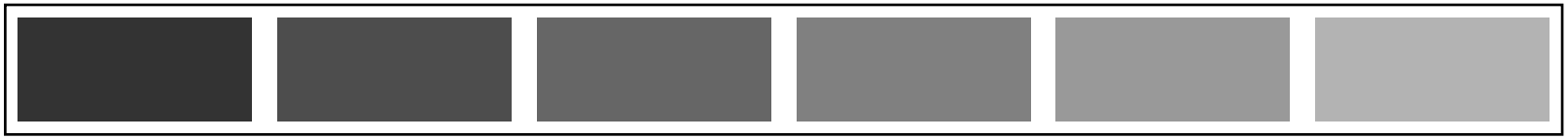
Memory Access

Cache



Cache



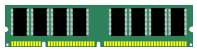


Memory Access

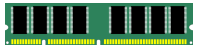
Cache

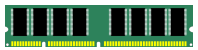
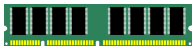
Memory Access

Cache

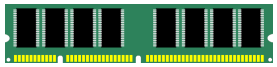


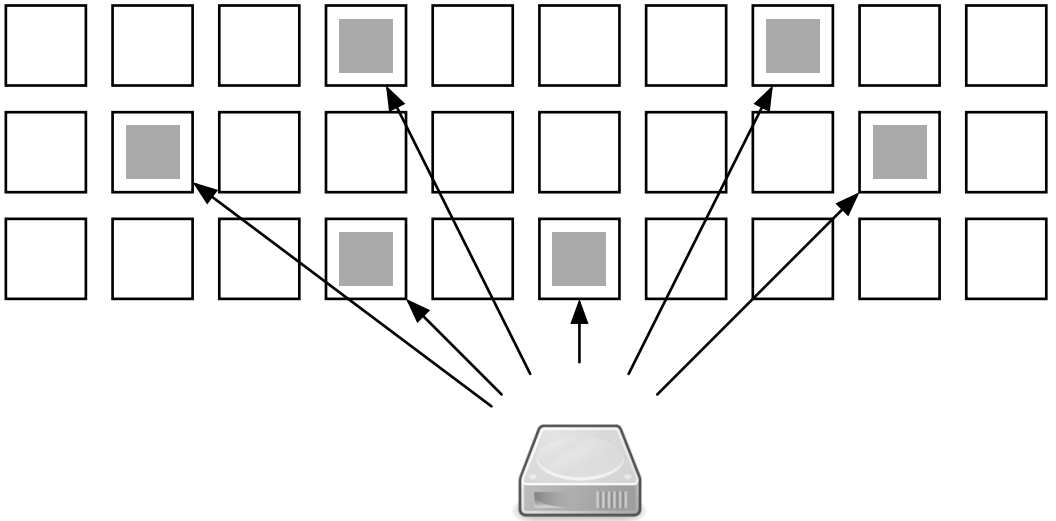
Memory Access



- **Read** : $\text{PageID} \rightarrow$ 
- **Write** : $\text{PageID},$  $\rightarrow$
- **Allocate** : $\rightarrow \text{PageID}$

Buffer Manager

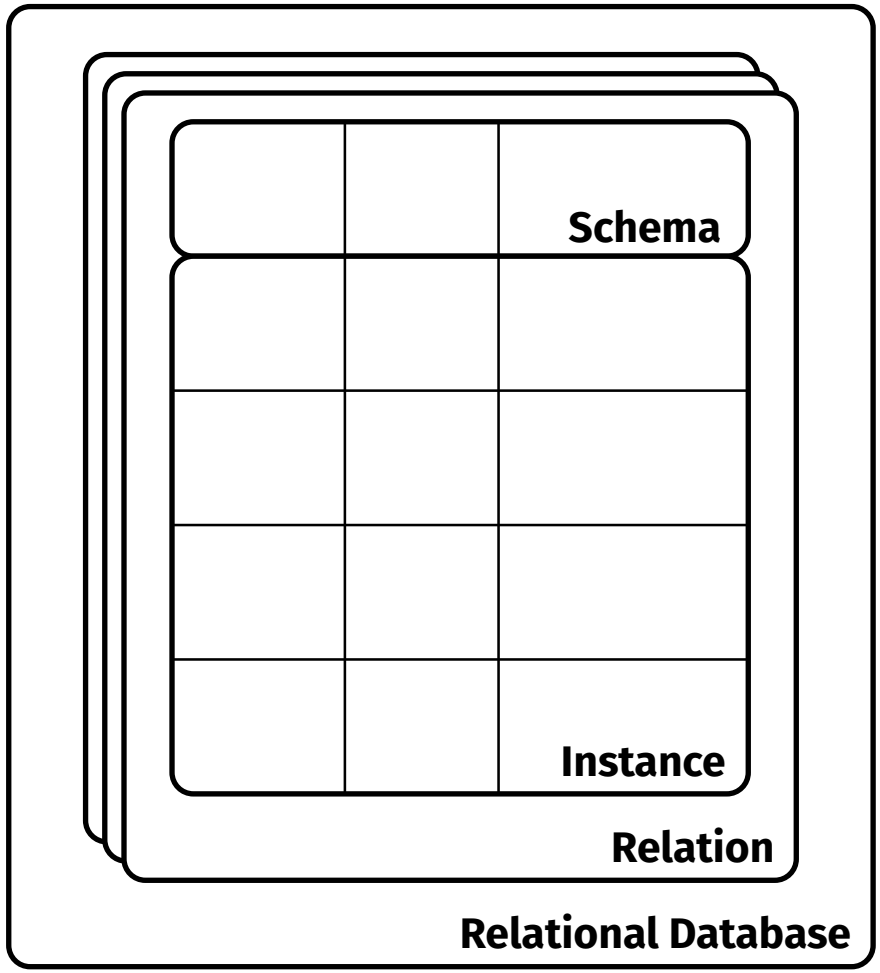


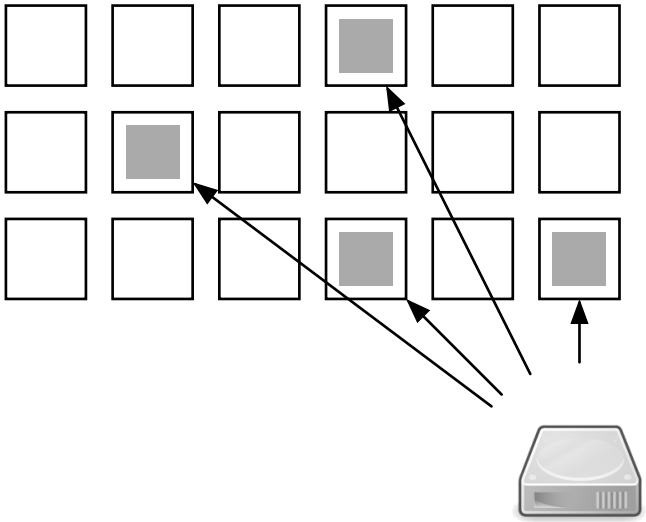


- **Pin:** $\text{PageID} \rightarrow \square$
- **UnPin:** $\text{PageID} \rightarrow$

$|R|$?

Logical vs Physical Representations of Data





Pages != Rows

Today

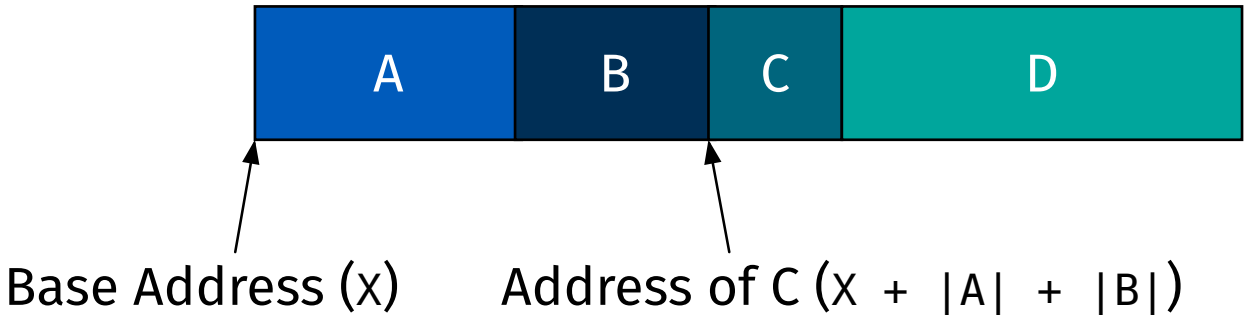
1. How do we represent an attribute value in a region of memory?
2. How do we represent a record in a region of memory?
3. How do we organize records in a page?

Attributes in Memory

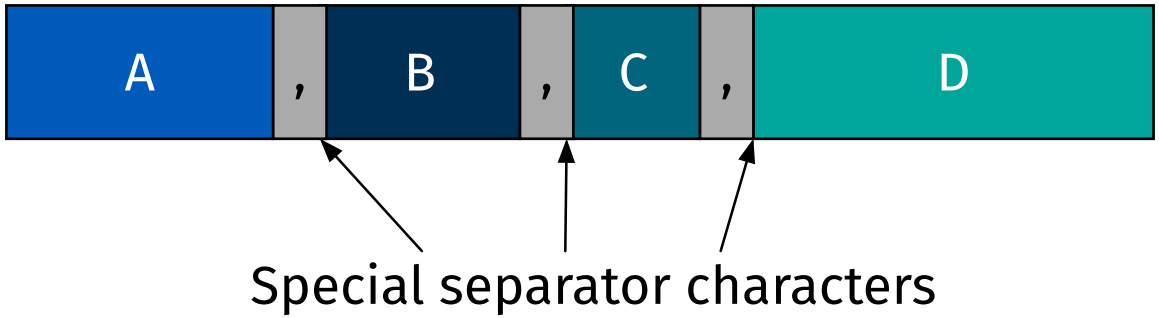
Data: $[B_1]\,[B_2]$, ..., $[B_N]$

1. Integers (fixed size)
   - `INT1`: B_1
   - `INT2`: $B_1 + 256 \cdot B_2$
   - `INT4`: $B_1 + 256 \cdot B_2 + 256^2 \cdot B_3 + 256^3 \cdot B_4$
2. Float (fixed size)
   - `FLOAT`: $B_1, \ldots, B_4$: 1 bit sign, 8 bits exponent, 23 bits value
3. Strings (variable size)
   - `CHAR(N)`: '$B_1 B_2 \ldots B_N$'
   - `string`: '$B_1 B_2 \ldots B_N$`\0`'

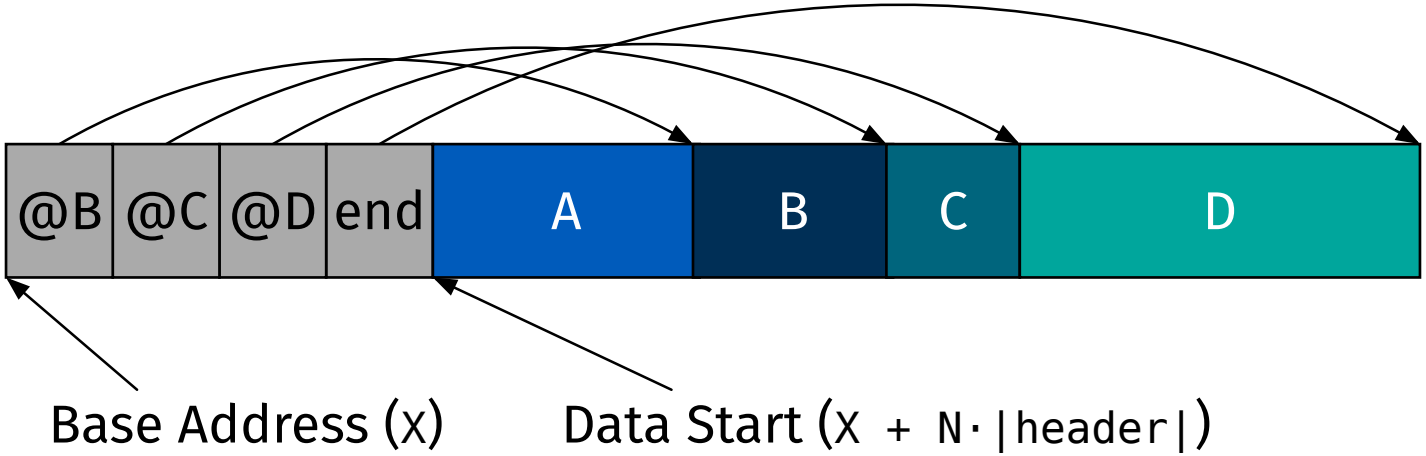
Records: Fixed Layout



Records: Delimited Layout



Records: Header-Organized Layout



Fixed

Constant size fields.

- Field i at byte $\sum_{j<i} |\text{Field}_i|$
- Field length is known

Delimited

Character separates fields.

- Find field i by finding the ith occurrence of the character.
- Find length of field i by finding the $i+1$th occurrence.

Header

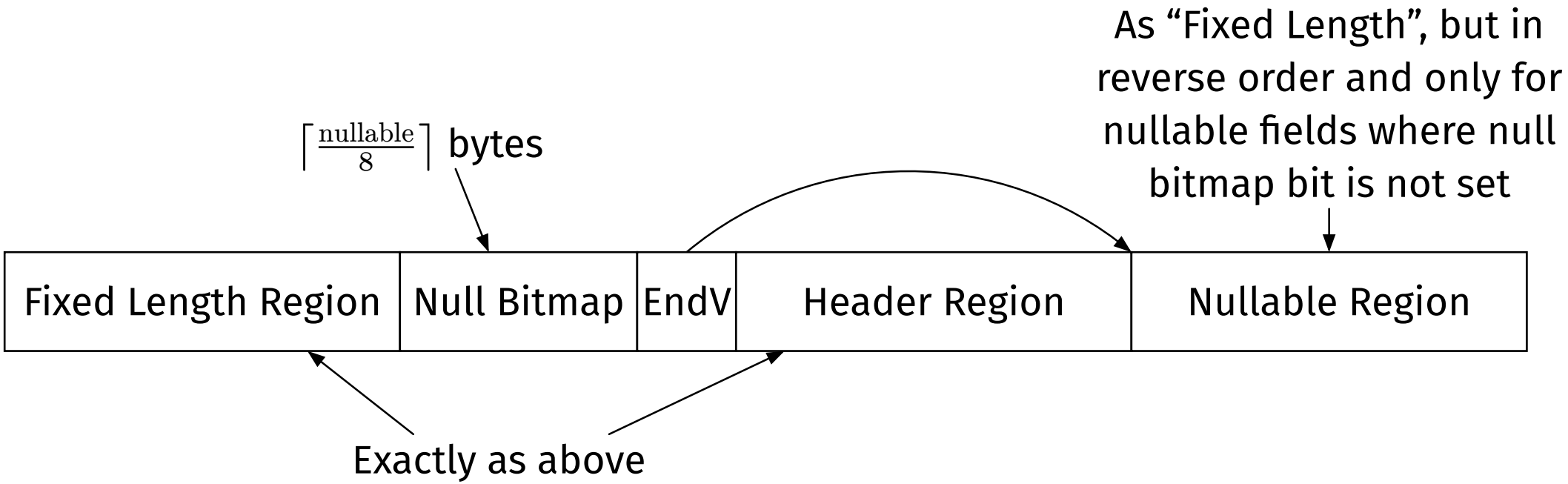
Fixed size header points to the start of each field.-

- Field 0 at $N \cdot |\texttt{header}|$; Offset to field i at $i \cdot |\texttt{header}|$
- Compute field i's length by comparing $i \cdot |\texttt{header}|$ and $(i+1) \cdot |\texttt{header}|$

4 byte integers must start at a multiple of 4

How do you store a record consisting of a 2 byte integer and a 4 byte integer?

Checkpoint 2 Layout



Checkpoint 2 Layout Examples


```
CREATE TABLE R(
  A int
  B int not null,
  C varchar(20)
  D long int not null,
  E varchar(20) not null
  F float
)
```

1. <A: 10, B: 15, C: "foo", D: 20, E: "bar", 12.8>
2. <A: NULL, B: 15, C: NULL, D: 20, E: "bar", 12.8>
3. <A: NULL, B: 15, C: NULL, D: 20, E: "bar", NULL>

Page Layouts

Overview: We treat each page as an array of records.

Each record is identified by its PageID and the position in the array (Slot ID)

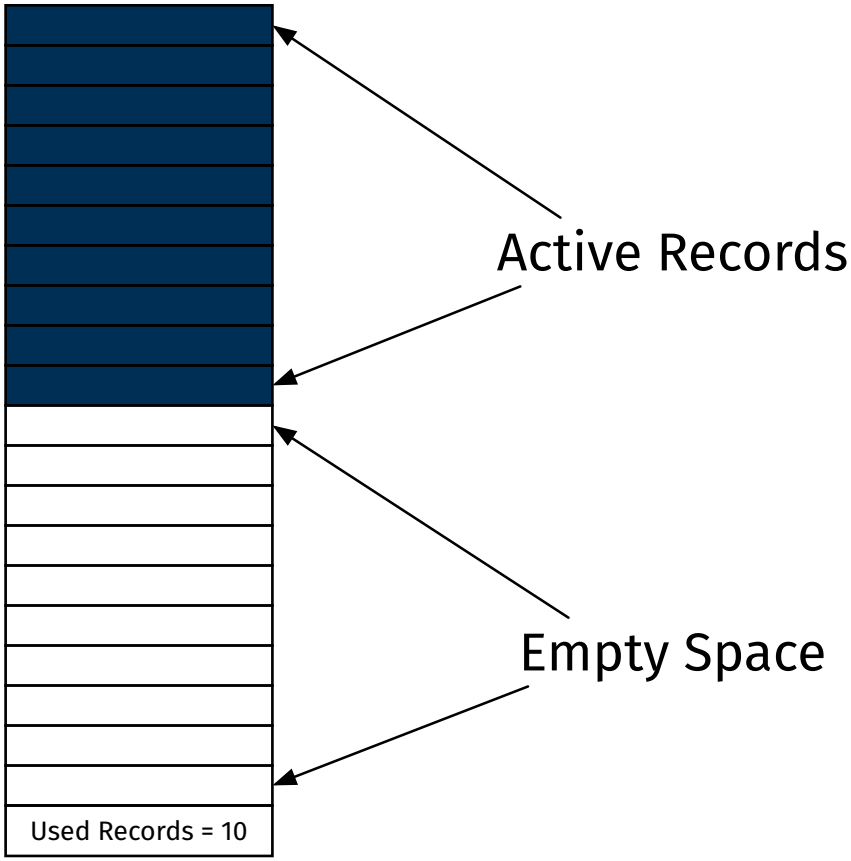
API

1. Find an empty entry
2. Insert a record
3. Remove a record
4. Update a record

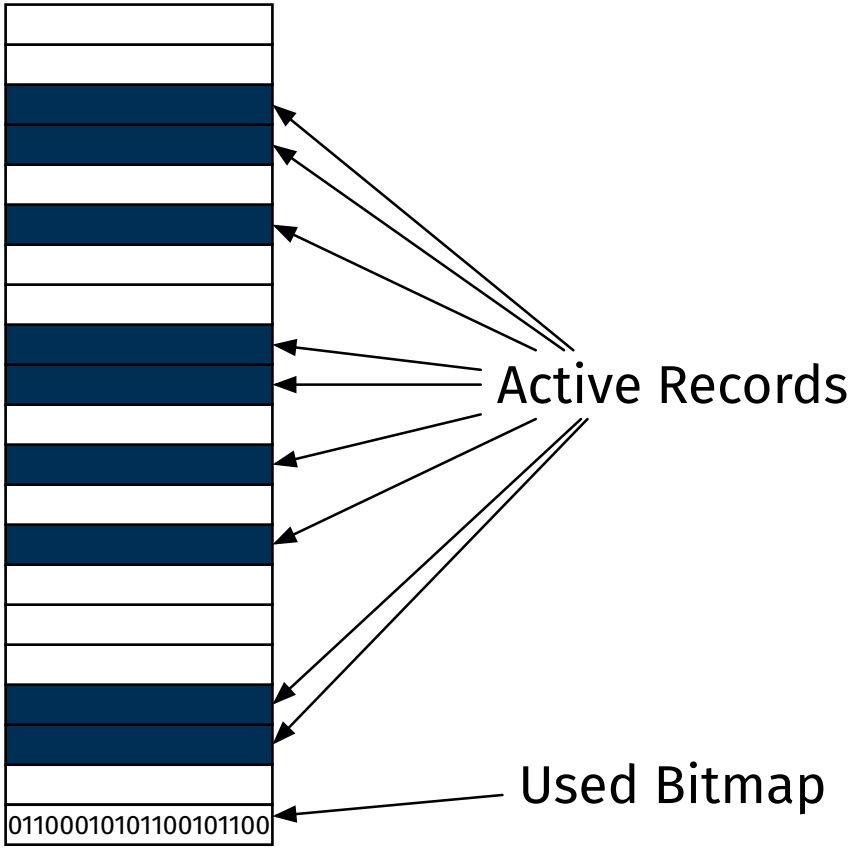
Approaches

- **Fixed Size Records**: Subdivide the page into an array of records
- **Variable Size Records**: Similar to Header organized records

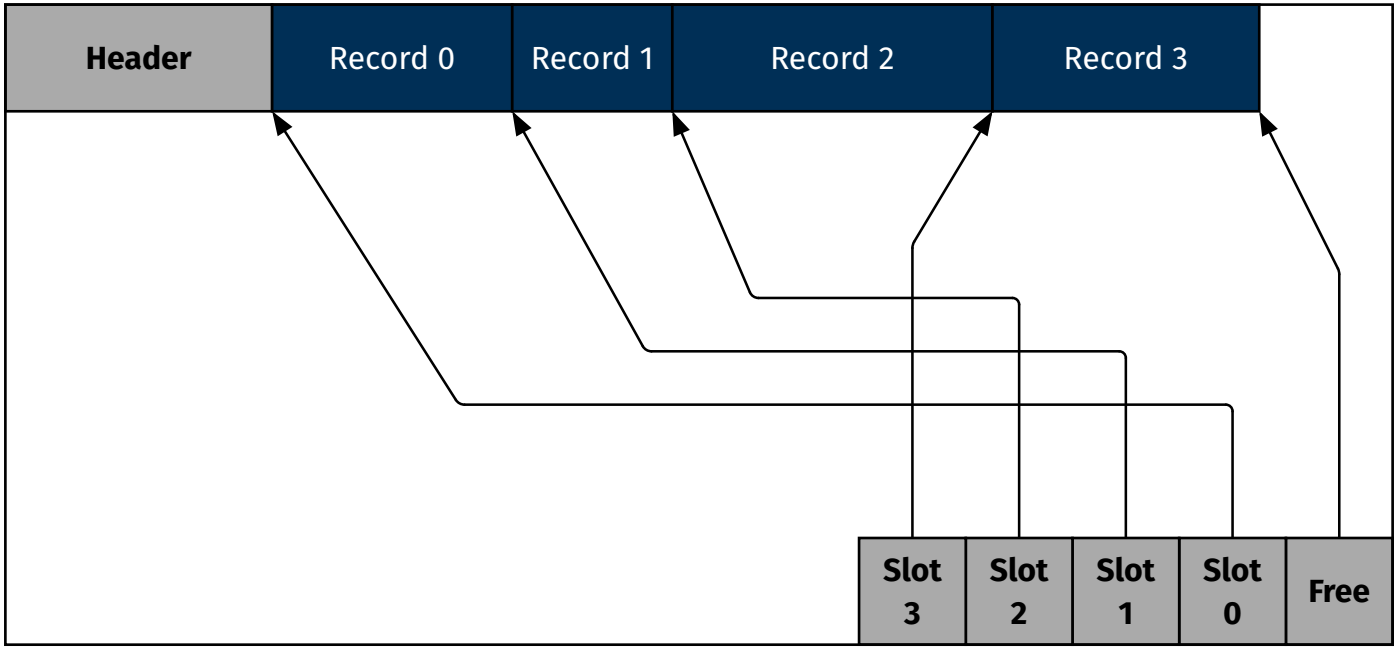
Fixed Layouts: Packed



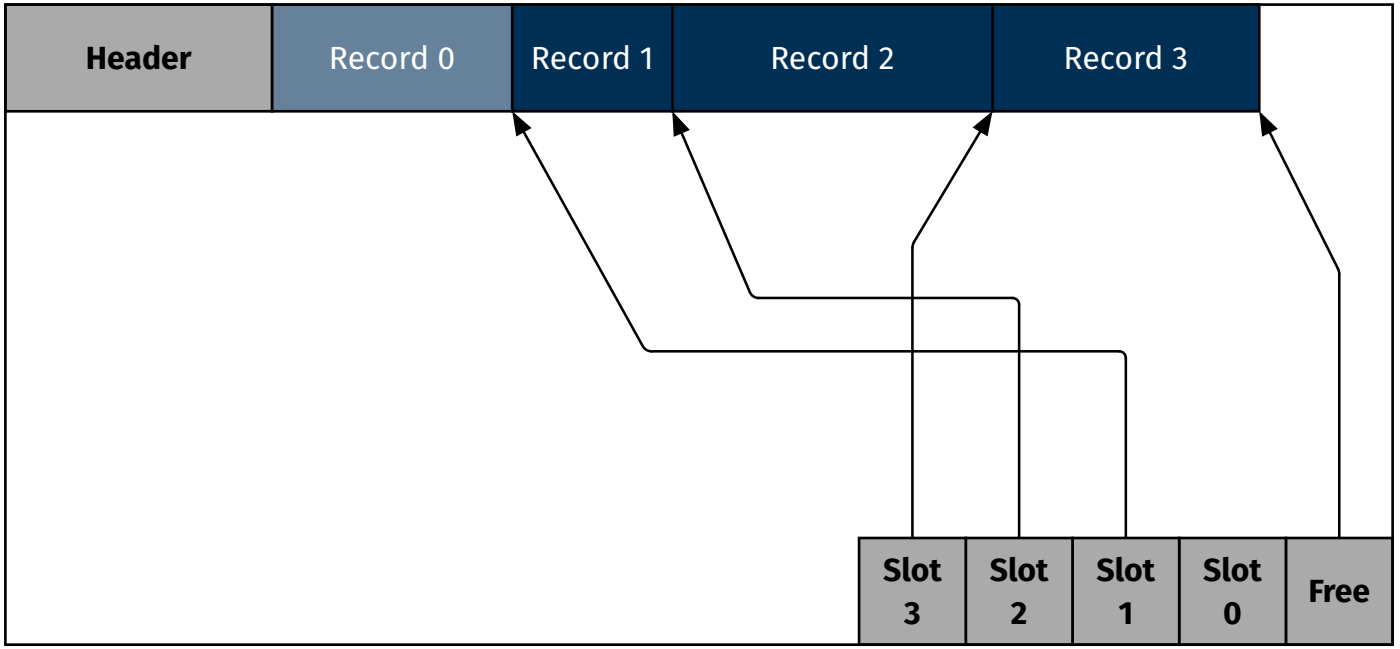
Fixed Layouts: Unpacked



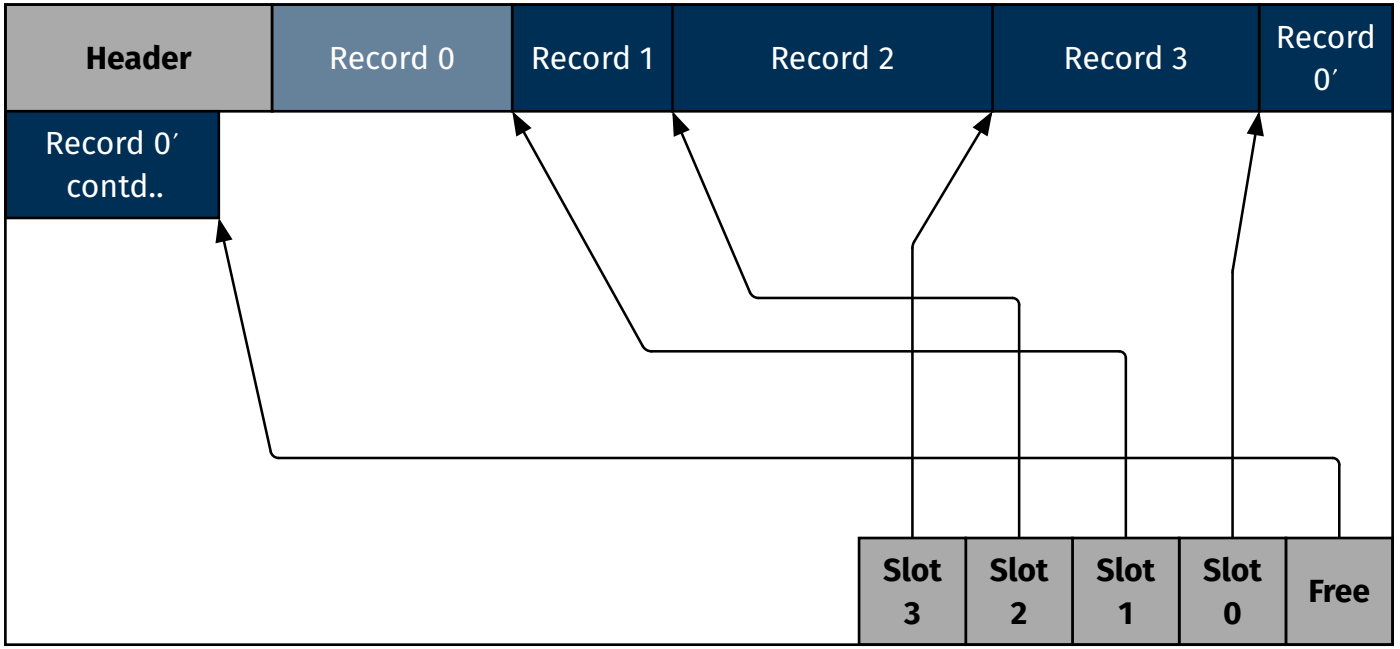
Variable-Size Layout



Variable-Size Layout



Variable-Size Layout



Page compaction typically happens lazilly.

When a page is “full”, we compact and then retry the insert.

Heap File

A collection of pages organized into a linked list.

Checkpoint 2

Overview

Implement (at least in part)...

1. Key parts of record layouts (`Schema.cpp`)
2. Page layouts (`VarLenDataPage.cpp`)
3. Collections of pages (Tables) (`Table.cpp`)
4. SQL to RA (`InterpretSQL.cpp`)
5. Filter, Project, Cartesian Product, Table Scan (`plan/*.cpp`, `execution/*.cpp`)

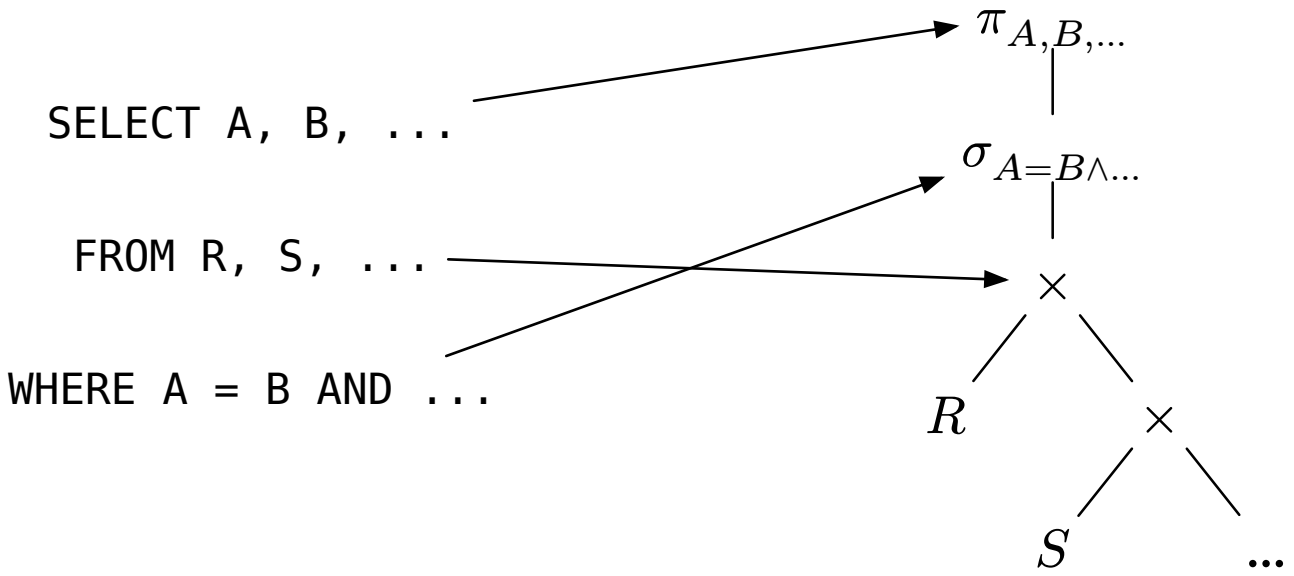
Record Layouts

See discussion of record layouts above

See discussion of page layouts above

See discussion of table layouts above. File header API provides next/prev pointers.

SQL to RA


SELECT A, B, ...
FROM R, S, ...
WHERE A = B AND ...
$\pi_{A,B,\ldots}$
$\sigma_{A=B\wedge\ldots}$
$\times$
R
$\times$
S
...

SQL Evaluation

See last week's lectures.

Reminders

- Checkpoint 2 posted; Autolab up soon.
- Checkpoint 1 solutions to be posted soon.
- Schedule code review meetings.