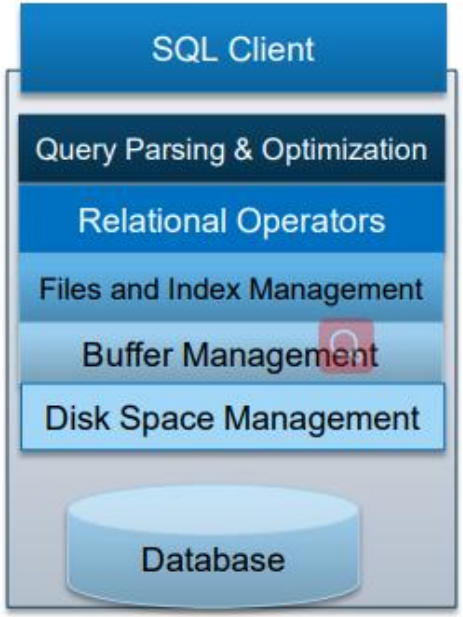
Discussion 2

Disks, Files, Buffers

Bin Wang

DBMS

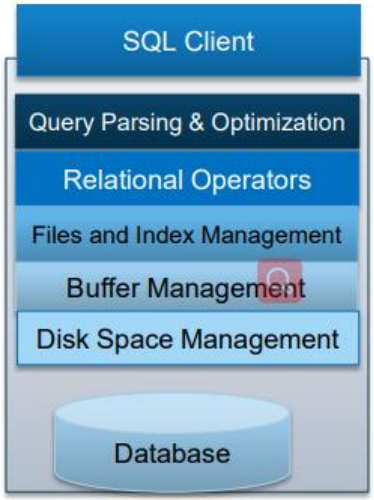
DBMS


SQL Client
Query Parsing & Optimization
Relational Operators
Files and Index Management
Buffer Management
Disk Space Management
Database

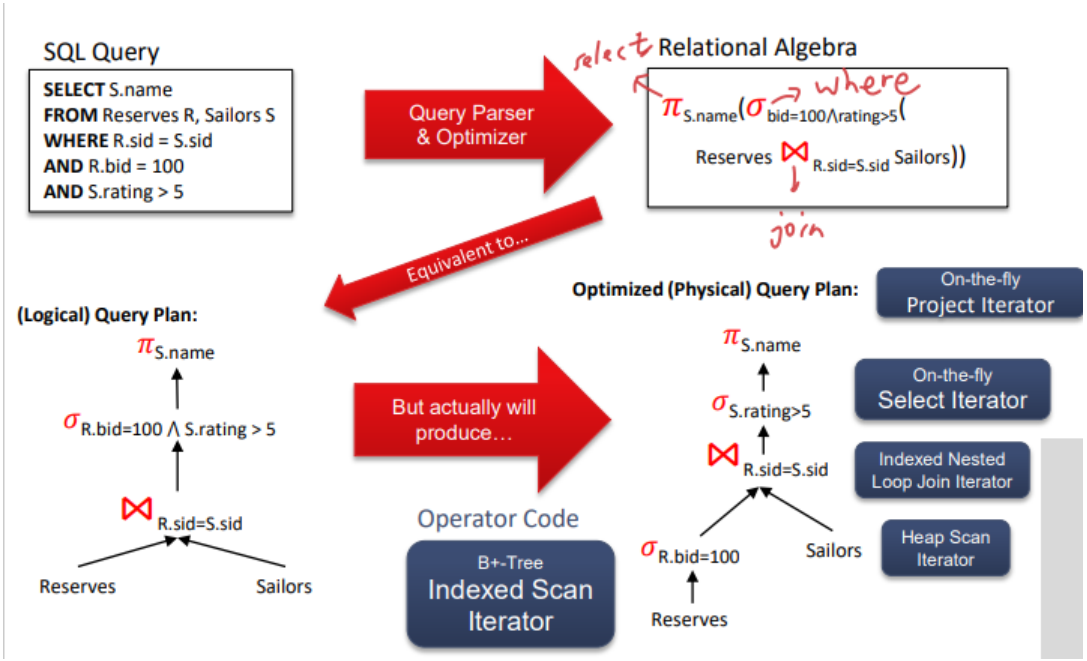
Query Parsing & Optimization

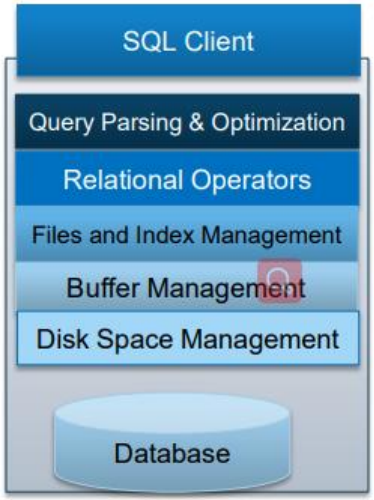
```
SELECT S.sid, S.sname, R.bid
FROM Sailors R, Reserves R
WHERE S.sid = R.sid and S.age > 30
GROUP BY age
```

- Check SQL you write
- Translate it into relational Operator

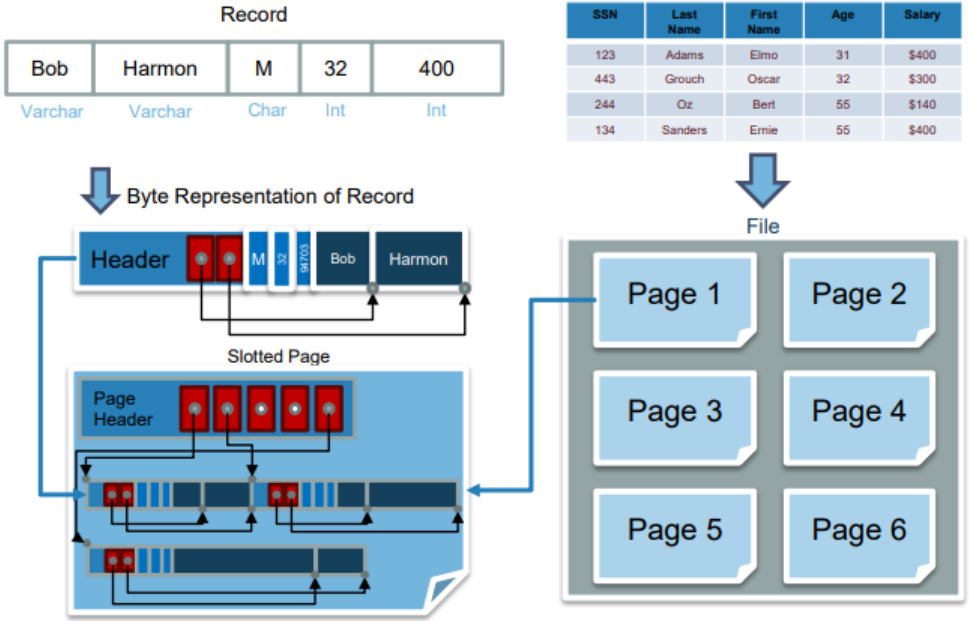


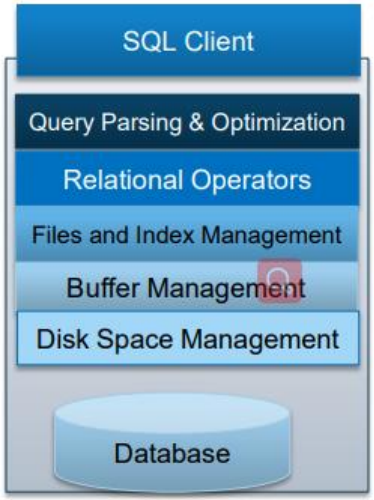
Relational Operators



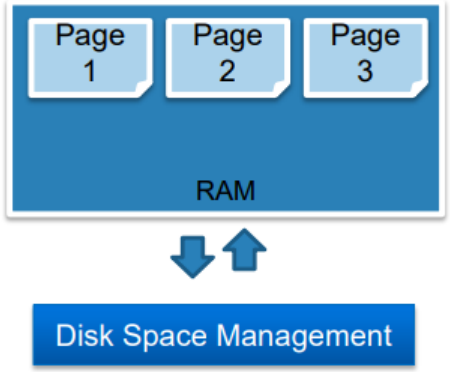


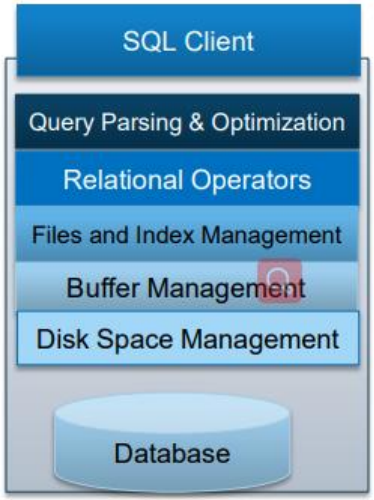
Files and Index Management





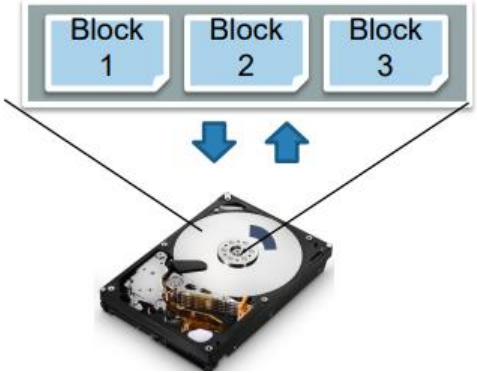
Buffer Management

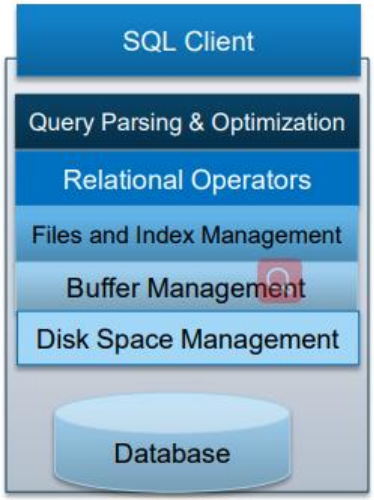




- Draw data from disk and store it in the RAM
- Send back long-term unused data to the disk from RAM

Disk Space Management





Page/Record Formats

Overview: Files of Pages of Records

- Tables stored as logical files consisting of pages each containing a collection of records

- File (corresponds to a table)
  - Page (many per file)
    - Record (many per page)

Overview: Pages and I/Os

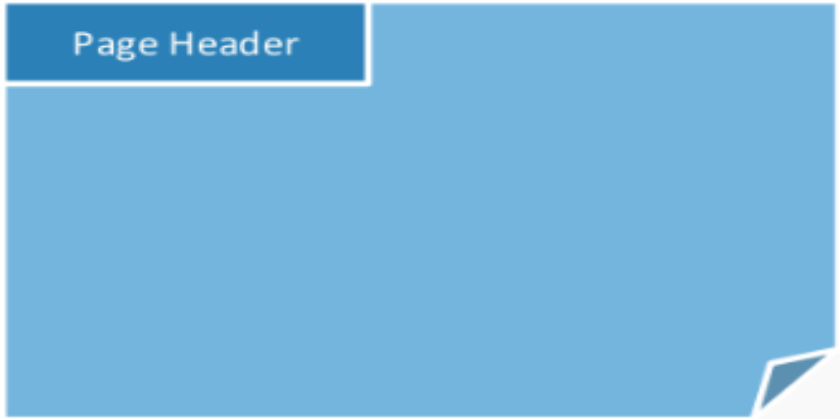
- Pages are managed
  - in memory by the buffer manager: higher levels of database only operate in memory
  - on disk by the disk space manager: reads and writes pages to physical disk/files
- Unit of accesses to physical disk is the page
  - **Cannot** fetch fractions of a page
  - I/O: unit of transferring a page of data between memory and disk (read OR write 1 page = 1 I/O)

Page basics: the header

The **page header** is a portion of each page reserved to keep track of the records in the page.

The page header may contain fields such as:

- Number of records in the page
- Pointer to segment of free space in the page
- Bitmap indicating which parts of the page are in use



Fixed Length Records

Records are made up of multiple **fields** (think: values for columns in a table).

We have **fixed length records** when both the following are true:

- Record lengths are fixed: every record is always the same number of bytes
- Field lengths are consistent: the first field always has N bytes, the second field always has M bytes, etc.

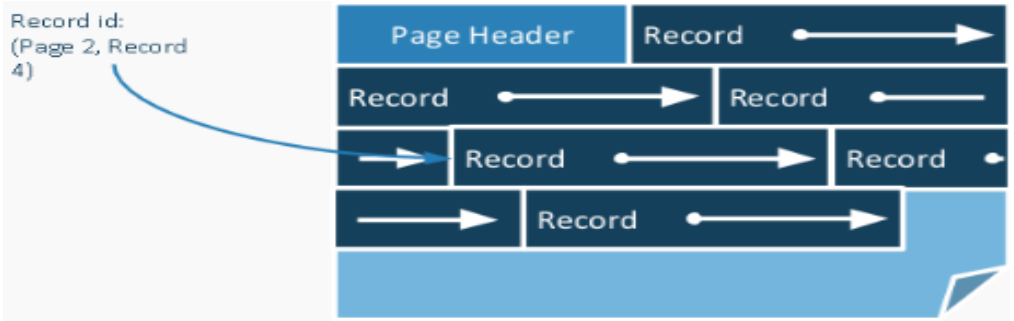
For example:

```
                |  length 5     |  2   |     3     |     4     |
Record1 =       |     Field1    |Field2| Field3    |  Field4   |
Record2 =       |     Field1    |Field2| Field3    |  Field4   |
Record3 =       |     Field1    |Field2| Field3    |  Field4   |
```

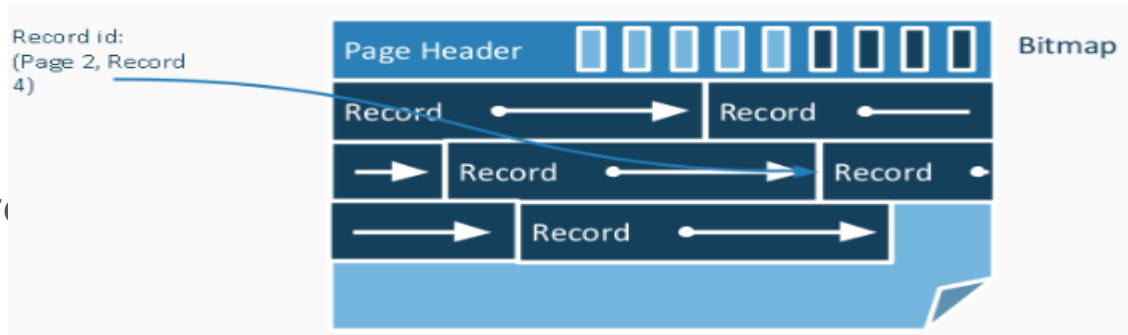
Fixed Length Records

We can store fixed length records in two ways:

- **packed**: no gaps between records, record ID is location in page



- **unpacked**: allow gaps between records, use a bitmap to keep track of where the gaps are



Variable Length Records

We have **variable length records** when we don’t satisfy the conditions for fixed length records, that is, either:

- Record lengths are not fixed: records can take different number of bytes
- Field lengths are not consistent: the third field may take 0 to 4 bytes

Variable Length Records

Two ways to store variable length records:

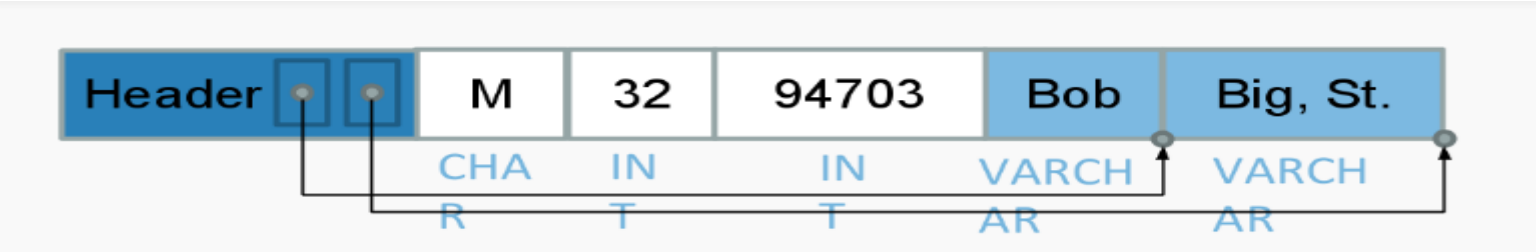
- Delimit fields with a special character ($ in the diagram below)

  - What if F2 contains ‘$’?

Variable Length Records

Two ways to store variable length records:

- Array of field offsets

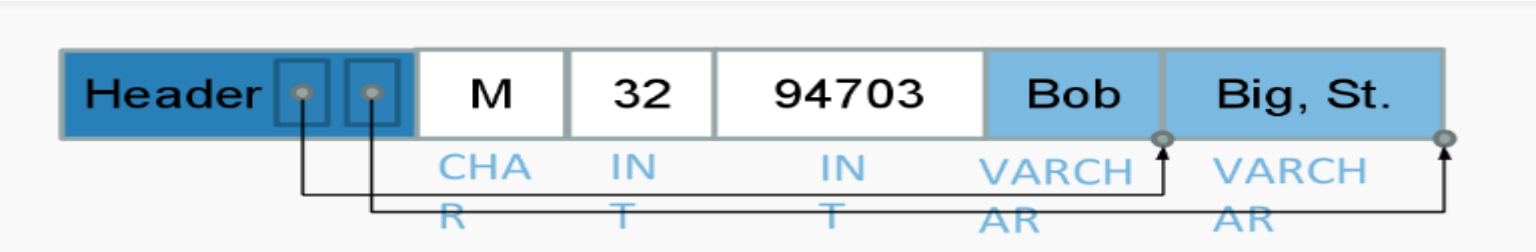


  - Each record contains a **record header**
  - Variable length fields are placed *after* fixed length fields
  - Record header stores **field offset** indicating where each variable length field ends

Variable Length Records

Two ways to store variable length records:

- Array of field offsets



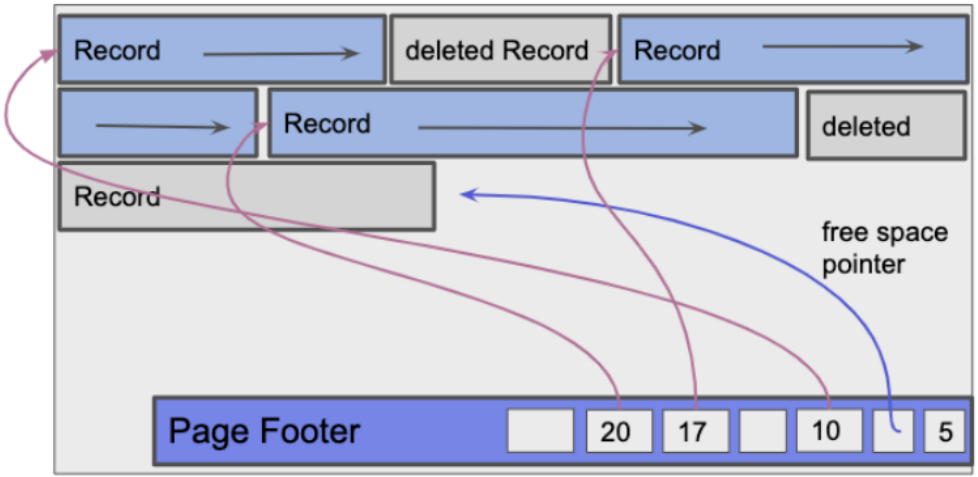
  - An aside: this is not actually sufficient for storing NULLs
    - Cannot distinguish between empty string (“”) and NULL
    - Need some extra metadata (e.g. bitmap in record header or special char in field), which varies widely between different DBMS

Variable Length Records

- How do we know where each record begins?
- What happens when we add and delete records?

Variable Length Records: Slotted Pages

- Move page header to *end* of page (footer) - to allow for header to grow
- Store length and pointer to start of each record in footer
- Store number of slots and pointer to free space



Slotted Page (Unpacked Layout)

Variable Length Records: Fragmentation

- Deleting records causes fragmentation if we use an **unpacked** layout:
  - [3 byte record] [2 byte record] [3 byte record] [2 bytes free space]
  - If we delete the 2 byte record, we have 4 bytes free on the page but cannot insert a 4 byte record

File Organization

Heap Files and Sorted Files

A **heap file** is just a file with no order enforced.

- Within a heap file, we keep track of pages
  - Within a page, keep track of records (and free space)
    - Records placed arbitrarily across pages

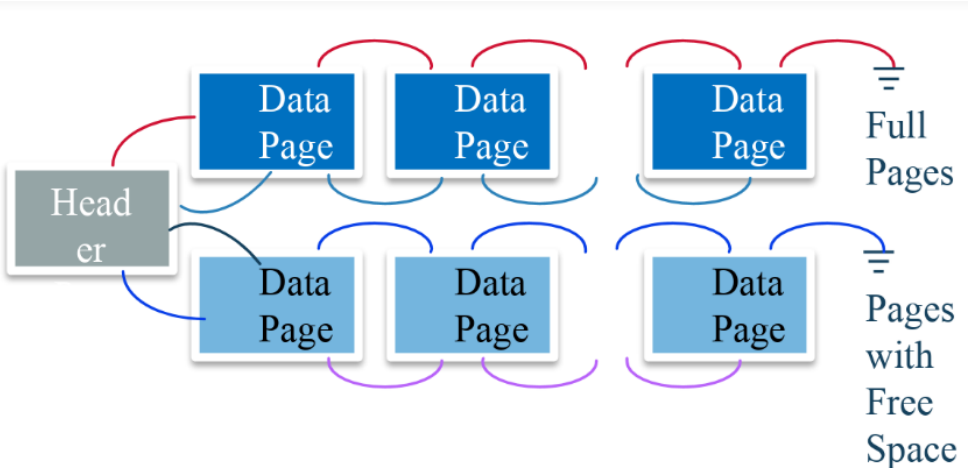
Record ID (RID) = <page id, slot #>

A **sorted file** is similar to a heap file, except we require it be sorted on a key (a subset of the fields).

Implementing Heap Files

One approach to implementing a heap file is as a **list**.

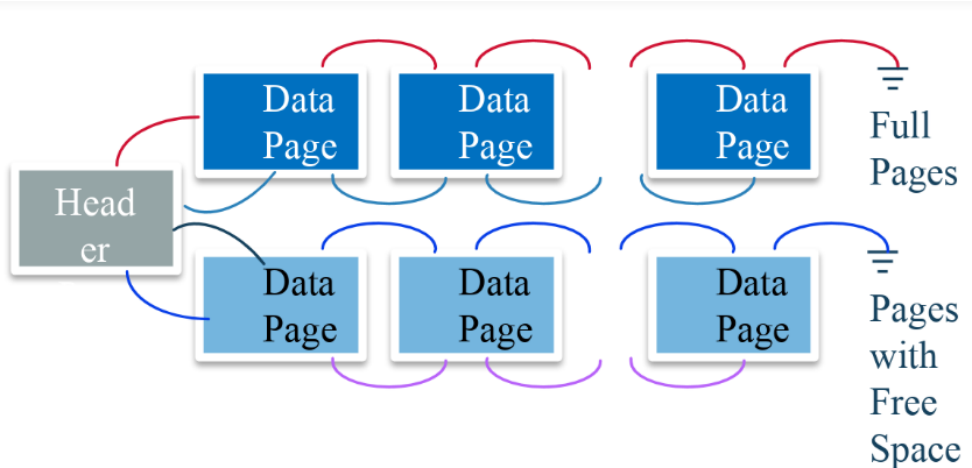
- Each page has two pointers, free space, and data.
- We have two linked lists of pages, both connected to a **header page**
  - List of full pages
  - List of pages with some empty space



Implementing Heap Files

One approach to implementing a heap file is as a **list**.

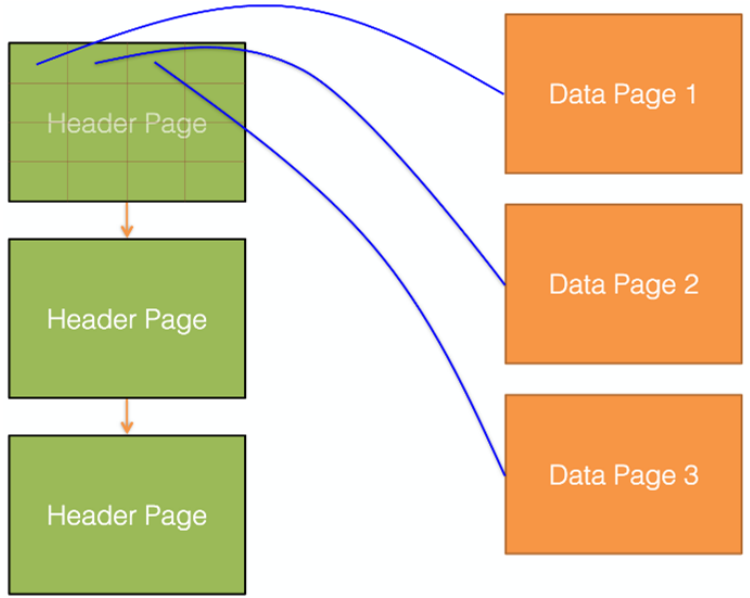
- How do we find a page to insert a 20-byte record in?



Implementing Heap Files

A different approach to implementing a heap file is with a **page directory**.

- We have a linked list of **header pages**, which are each responsible for a set of data pages
  - Stores the amount of free space per data page



Worksheet

True and False - A

When querying for an 16 byte record, exactly 16 bytes of data is read from disk.

True and False - A

When querying for an 16 byte record, exactly 16 bytes of data is read from disk.

False, an entire page of data is read from disk.

True and False - C

In a heap file, all pages must be filled to capacity except the last page.

True and False - C

In a heap file, all pages must be filled to capacity except the last page.

False, there is no such requirement.

True and False - D

Assuming integers take 4 bytes and pointers take 4 bytes, a slot directory that is 512 bytes can address 64 records in a page.

True and False - D

Assuming integers take 4 bytes and pointers take 4 bytes, a slot directory that is 512 bytes can address 64 records in a page.

False, we have the free space pointer, which doesn't fit after 64 * (4 + 4) = 512 bytes of per-record data in the slot directory.

True and False - E

In a page containing fixed-length records with no nullable fields, the size of the bitmap never changes.

True and False - E

In a page containing fixed-length records with no nullable fields, the size of the bitmap never changes.

True, the size of the records is fixed, so the number we can fit on a page is fixed.

True and False - Variable Length Records - A

Variable length records do not need a delimiter character to separate fields in the records

True and False - Variable Length Records - A

Variable length records do not need a delimiter character to separate fields in the records

True, using a record header eliminates this requirement.

True and False - Variable Length Records - B

Variable length records always match or beat space cost when compared to fixed-length record format

False, extra space is required for the record header.

True and False - Variable Length Records - C

Variable length records allow access to any field without scanning the entire record

True and False - Variable Length Records - C

Variable length records allow access to any field without scanning the entire record

True, can calculate position of any field using arithmetic.

Fragmentation and Record Formats #1

Is fragmentation an issue with packed fixed length record page format?

Fragmentation and Record Formats #1

Is fragmentation an issue with packed fixed length record page format?

No, records are compacted upon deletion.

Fragmentation and Record Formats #2

Is fragmentation an issue with variable length records on a slotted page?

Fragmentation and Record Formats #2

Is fragmentation an issue with variable length records on a slotted page?

Yes.