CS W186 - Spring 2024
Exam Prep Section 2
Disks and Files

Question 1: Files, Pages, Records

Consider the following relation:

```
CREATE TABLE Cats (
      collar_id INTEGER PRIMARY KEY, -- cannot be NULL!
      age INTEGER NOT NULL,
      name VARCHAR(20) NOT NULL,
      color VARCHAR(10) NOT NULL
);
```

You may assume that:

- INTEGERs are 4 bytes long;
- VARCHAR(n) can be up to *n* bytes long.

(a) As the records are variable length, we will need a *record header* in the record. How big is the record header? You may assume pointers are 4 bytes long, and that the record header only contains pointers.

Answer: 8 bytes.
In the record header, we need *one pointer for each variable length value*. In this schema, those are just the two VARCHARs, so we need 2 pointers, each 4 bytes.

(b) Including the record header, what is the smallest possible record size (in bytes) in this schema? (Note: NULL is treated as a special value by SQL, and an empty string VARCHAR is different from NULL, just like how a 0 INTEGER value is also different from NULL. We will provide the necessary clarification should similar questions appear on an exam.)

Answer: 16 bytes (= 8 + 4 + 4 + 0 + 0)
8 for the record header, 4 for each of integers, and 0 for each of the VARCHARs.

(c) Including the record header, what is the largest possible record size (in bytes) in this schema? Answer: 46 bytes (= 8 + 4 + 4 + 20 + 10)

(d) Now let's look at pages. Suppose we are storing these records using a slotted page layout with

variable length records. The page footer contains an integer storing the record count and a pointer to free space, as well as a slot directory storing, for each record, a pointer and length. What is the maximum number of records that we can fit on a 8KB page? (Recall that one KB is 1024 bytes.)

Answer: 341 records (= (8192 - 4 - 4) / (16 + 4 + 4))
We start out with 8192 bytes of space on the page.
We subtract 4 bytes that are used for the record count, and another 4 for the pointer to free space. This leaves us with 8192 - 4 - 4 bytes that we can use to store records and their slots. A record takes up 16 bytes of space at minimum (from the previous questions), and for each record we also need to store a slot with a pointer (4 bytes) and a length (4 bytes). Thus, we need 16 + 4 + 4 bytes of space for each record and its slot.

(e) Suppose we stored the maximum number of records on a page, and then deleted one record. Now we want to insert another record. Are we guaranteed to be able to do this? Explain why or why not.

Answer: No, we deleted 16 bytes but the record we want to insert may be up to 46 bytes.

(f) Now suppose we deleted 3 records. Without reorganizing any of the records on the page, we would like to insert another record. Are we guaranteed to be able to do this? Explain why or why not.

Answer: No; there are 48 free bytes but they may be fragmented - there might not be 46 contiguous bytes.

Question 2: Files, Pages, Records

Consider the following relation:

```
CREATE TABLE Student (
      student_id INTEGER PRIMARY KEY,
      age INTEGER NOT NULL,
      units_passed INTEGER NOT NULL,
);
```

(a) Are Student records represented as fixed or variable length?

Answer: Fixed. There are no variable length fields (e.g. VARCHARs).

Note: Some students asked about whether a NULLable field would make it variable length. Records with NULLable fields can be represented either as fixed or variable length (see lecture DBF 1 → "Fixed and Variable Length Records").

In general, "fixed" vs. "variable" length is a property of the chosen representation of the record, and not necessarily the record itself. VARCHAR(20) could be represented by padding to exactly 20 bytes if we really wanted to (see the above lecture video for this as well).
However, records with only fixed-length fields admit only fixed-length representations, so the for this question, the answer is clear.

(b) To store these records, we will use an unpacked representation with a page header. This page header will contain nothing but a bitmap, rounded up to the nearest byte. How many records can we fit on a 4KB page? (Recall that one KB is 1024 bytes.)

Answer: 337 records

We start with $4 * 1024 = 4096$ bytes on the page. Next, our fixed-length record is made up of 3 integers for a total of $4 + 4 + 4 = 12$ bytes. Lastly, we need one bit (⅛ of a byte) in the bitmap for each record. When we divide 4096/12.125, we get a bit over 337 records. It's important to note that the bitmap size must be rounded to the nearest byte, so with 337 records, the bitmap must be 344 bits (closest multiple of 8 bits), or 43 bytes. This makes the total number of bytes used $337 * 12 + 43 = 4087$, which fits within our page.

(c) Suppose there are 7 pages worth of records. We would like to execute

```
SELECT * FROM Student WHERE student_id = 3034213355; -- just some number
```

Suppose these pages are stored in a heap file implemented as a linked list. What is the minimum and maximum number of I/Os required to answer the query?

Answer: Minimum: 2. Maximum: 8.

First we need to read the header page. In the best case, we find the record on the first page we search, for a total of 2 I/O's (header page + 1 data page). In the worst case, we find the record on the last page we search, for a total of 8 I/O's (header page + 7 data pages)

(d) Now suppose these pages are stored in a sorted file, sorted on student id. What is the minimum and maximum number of I/Os required to answer the query? You can assume sorted files do not have header pages.

Answer: Minimum: 1. Maximum: 3.
As seen in Lecture 5, we do binary search to find records in a sorted file.

Question 3: Files, Pages, Records

(a) Suppose we are storing variable length records in a linked list heap file. In the ”pages with space” list, suppose there happens to be 5 pages. What is the maximum number of IOs required in order to insert a record?

You may assume that *at least one of these pages contains enough space*, and additionally that *it will not become full after insertion*.

Answer: 7 I/Os.
(1) Read header page.
(5) Read up to all 5 data pages (since the records are variable length, not all pages with space might have enough space to store a large record).
(1) Write updated data page (with the newly inserted record).

(b) Continuing from part (a), suppose that the page does become full after insertion. Now, we need to move that page to the ”full pages” list.

Assume we have already done all necessary page reads for part (a)’s worst case (and that those pages are still in memory), but have not yet done any page writes.

How many additional I/Os do we need to move the page to the ”full pages” list?

Answer: 5 additional page I/Os.
The idea behind the answer is this: We have two doubly linked lists; one for the full pages, another for the non-full pages; they are both connected to the header page.
We have a page at the end of the ”non-full pages” list (from part (a)) that we would like to move to the head of the ”full pages” list (the head, because that’s the cheapest place to insert, which we are allowed to do because the pages are not linked in any particular order).
So - we are doing a doubly-linked list node movement from the tail of one list to the head of the other, which is done through pointer updates. We need to update pointers on the page we’re moving, its old neighbour, and its two new neighbours (the header page, and the old first page in the full pages list). Counting this out, this becomes:
(1) Write updated data page.
(1) Write previous non-full page (the old backwards neighbour).
(1) Read old first full page (the new forwards neighbour).
(1) Write old first full page.
(1) Write header page (the new backwards neighbour).

(c) Now suppose records are fixed length; what is the maximum number of I/Os to insert a record? Assume that the page we insert into does not fill up after the insertion.

Answer: 3 page I/Os.
(1) Read page header.
(1) Read a non-full page.
(1) Write updated non-full page.
Because the record is fixed length, the first page in the “pages with space” list is guaranteed to have enough space to insert at least one record.

(d) Now suppose we are using a page directory, with one directory page. What is the maximum number of I/Os required to insert a record?

Answer: 4 page I/Os.
(1) Read page directory.
(1) Read data page with space.
(1) Write to data page with space.
(1) Write updated page directory (we need to update the amount of free space).