CS W186 Databases - Fall 2019
Guerilla Section 1: Disks, Files, B-Tree Indices

Sunday, September 15, 2019

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:

- `INTEGER`s 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.

(b) Including the record header, what is the smallest possible record size (in bytes) in this schema?

(c) Including the record header, what is the largest possible record size (in bytes) in this schema?

(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 a slot directory with a pointer per record, as well as an integer storing the record count and a pointer to free space. What is the **maximum** number of records that we can fit on a 8KB page? (Recall that one KB is 1024 bytes.)

(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.

(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.

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 fixed or variable length?

(b) To store these records, we will use an unpacked representation with a page header. This page header will contain nothing but a bitmap, which is as small as possible, rounded up to the nearest byte. How many records can we fit on a 4KB 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 list. What is the minimum and maximum number of pages in this heap file we would need to touch to answer the query?

(d) Now suppose these pages are stored in a sorted file. What is the minimum and maximum number of pages you would need to touch?

Question 3: Files, Pages, Records

(a) Suppose we are using a heap file, with a linked list: one header page, and are 5 pages in the "pages with space" list. Suppose records are variable length. What is the maximum number of pages we might have to access in order to insert a record?

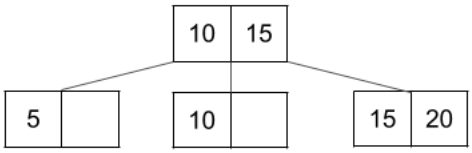
(b) If our insertion caused a page to fill up in the question above, we might need to move that page to the "full pages" list. Including the page we just filled up, how many pages might we have to touch to make that happen?

(c) Now suppose records are fixed length; what is the maximum number of pages we might have to access to insert a record?

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

Question 4: B-Trees

Given the following B+ tree:



(a) Draw what the tree looks like after adding 2, 6, and 12 in that order.

For the following questions, consider the tree directly after 2, 6, and 12 have been added.

(b) What is the maximum number of inserts we can do without changing the height of the tree?

(c) What is the minimum number of inserts we can do that will change the height of the tree?

Question 5: B-Trees

Consider the following schema:

```
CREATE TABLE FineWines (
    name CHAR(20) NOT NULL,
    years_aged INTEGER NOT NULL,
    price INTEGER NOT NULL
);
```

Suppose that we have built an index over <years_aged, price>, and suppose this index is two levels deep (in addition to the root node), and data is stored by *list of references* in separate data pages, each of which can hold hundreds of records.

(a) We would like to execute the query

```
SELECT * FROM FineWines
WHERE years_aged = 5
  AND price = 100
```

What is the worst case number of page accesses to execute this query on an unclustered index if there is 1 matching record? 2 matching records? 3 matching records?

(b) What is the worst case number of page accesses to execute this query on a clustered index if there is 1 matching record? 2 matching records? 3 matching records?

(c)
```
SELECT * FROM FineWines
WHERE price = 100
```

What is the worst case number of page accesses to execute this query if there are 150 data pages?

(d)
```
DELETE FROM FineWines
WHERE years_aged = 1
  AND price = 5
```

Suppose this query deletes exactly one record. How many page reads and writes are needed to execute this query? (Assume no tree rebalancing is required.)