CSC3170 Assignment2

Correspondence: Yuyang Xia

October 2024

1 Introduction

- This is the second homework for CSC3170. It weighs 10% of your final grade.
- The deadline for this homework is October 27th 11:59 pm.
- This assignment includes two parts, a writing part, and a coding part, each part weighs 50%.
- Directly using AI-generated answers or other students' answers will be considered as cheating.

2 Writing part (50%)

2.1 Number of tuples in one page (10%)

1. What is the smallest size, in bytes, of a record from the following schema in a slotted page? Assume that the record header is 4B, the boolean is 1B, and the date is 8B, disregarding word-alignment.(2%)

```
name VARCHAR
student BOOLEAN
birthday DATE
state VARCHAR
```

Figure 1: Provided schema.

4B+0B+1B+8B+0B=13B. Based on the storing order: record header, name, student, birthday, and state.

2. What is the maximum number of records that can be stored in a 2KB page given the schema above? Assume the slot count, free space pointer, record pointer, and record length costs 8B each. Note that you are using the N-ary storage model (NSM) and need to use padding to conduct word-alignment where each word is 8B long. (4%)

   After padding, each record occupies 16 bytes: one word for the record header and the student attribute (as the student attribute is padded alongside the name attribute and the record header), and one word for the birthday attribute.

   Each page only has one slot count and one free space pointer, but each record will have its record pointer and record length.
   So the maximum number is (2048-8-8)/(16+8+8) = 63.

3. If we are required to use reordering to conduct word-alignment, what is the maximum number of records that can be stored with the same setting in part 2? (4%)

   Similar to problem 2.1.2, the reordering here does not help reduce the space cost. The answer is still 63.

2.2 Number of I/Os for query (20%)

Consider a database with a single table `Class`(id, name, instructor, size, credits), where id is the *primary key*, and all attributes are the same fixed width. Suppose `Class` has 10000 tuples that fit into 500 pages, ignore any additional storage overhead for the table (e.g., page headers, tuple headers).

Additionally, you should make the following assumptions:

- The DBMS does not have any additional meta-data (e.g., sort order, zone maps).
- `Class` does not have any indexes (including for primary key id).
- None of `Class`'s pages are already in the buffer pool.
- Content-wise, the tuples of `Class` will make each query run the longest possible (this assumption is critical for solving part 1).

- The tuples of `Class` can be in any order (this assumption is critical for solving part 2 when you compute the minimum versus the maximum number of pages that the DBMS will potentially have to read)

1. Consider the following query:

```
SELECT MAX(credits) from Class
    WHERE size > 50
```

   (a) Suppose the DBMS uses the decomposition storage model (DSM) with implicit offsets. How many pages will the DBMS potentially have to read from disk to answer this query? (5%, keep in mind our assumption about the contents of `Class`!)

   200 pages. The attributes *credit* and *size* may both need 100 pages.

   (b) Suppose the DBMS uses the N-ary storage model (NSM). How many pages will the DBMS potentially have to read from disk to answer this query? (5%, keep in mind our assumption about the contents of `Class`!)

   500 pages. You may need to read all the pages to find the attribute.

2. Consider another query:

```
SELECT name, instructor, size from Class
    WHERE id = 114514 or id = 23333
```

   (a) Suppose the DBMS uses the decomposition storage model (DSM) with implicit offsets. What is the minimum and maximum number of pages that the DBMS will potentially have to read from disk to answer this query? (5%)

   4 pages and 106 pages. 4 pages. Suppose both primary keys appear on the first page. Since all attributes are of the same fixed width, each attribute of course id=15445 and course id=15645 will also appear on the same page. We'll thus need to read 1 page to find the two primary keys and read 3 pages to access course name, instructor, and class size at their corresponding offsets.

106 pages. There are 100 pages per attribute. In the worst case, we scan through all 100 pages to find the two primary keys. In the worst case, the two primary keys will be located on different pages. Since all attributes are of the same fixed width, each attribute of course id=15445 and course id=15645 will also appear on different pages. Hence we must read 2 pages to access each attribute of course id=15445 and course id=15645 at their corresponding offsets. Thus, we read 6 pages in total to access the course name, instructor, and class size.

(b) Suppose the DBMS uses the N-ary storage model (NSM). What is the minimum and maximum number of pages that the DBMS will potentially have to read from disk to answer this query? (5%)

1 page and 500 pages.

2.3 Cuckoo hashing (20%)

Consider the following cuckoo hashing schema:

1. Both tables have a size of 4.
2. The hashing function of the first table returns the fourth and third least significant bits: $h_1(x) = (x >> 2)$ & $0b11$ ($0b11$ is a binary representation and equals to 3 in decimal).
3. The hashing function of the second table returns the least significant two bits: $h_2(x) = x$ & $0b11$.
4. When inserting, try Table 1 first.
5. When replacement is necessary, first select an element in the second table.
6. The original entries in the table are shown in the figure below.

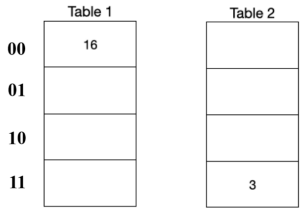

Figure 2: Initial status.

As shown on the left, the entries in each table are arranged from up to down.

1. Which is first inserted to the tables, 16 or 3? (5%) 16. Based on the 22nd page of 7th lecture slides. We will try to insert the element into table 1 and then table 2. Only when we cannot directly insert it, will we replace the elements in the table.

2. Now we insert 8 and delete 16. Please draw the hash tables after the operations. (5%)

   8 in Table 1's 10.

   3 in Table 2's 11.
   Other places are left blank.

3. After the operations of part 2, we further insert 19 and then insert 4. Please draw the hash tables after the operations. (5%)

   19 in Table 1's 00.

   4 in Table 1's 01.

   8 in Table 1's 10.

   3 in Table 2's 11.
   Other places are left blank.

4. After the operations of parts 2 and 3, what is the minimum number of insertions to cause an infinite loop? (5%)

   One insertion. Consider inserting 35.

3 Coding part (50%)

You are required to implement a classical page replacement algorithm, the *Optimal Page Replacement Algorithm*.

Note that this coding assignment is different from assignment 1. You are expected to use Java/Python/C++ for this problem instead of SQL.

3.1 Problem description

This is a simplified version of the page replacement question. In this problem, we assume that the user only wants to read the files in the disk **(i.e., a read-only task)** and the buffer pool is currently empty. Each time, the user will

only read one page of the on-disk files. Given the page reading sequence of the user, you want to figure out the minimal number of pages that the DBMS must read from the disk. Formally speaking, given a page call sequence with length n and a buffer capable of storing m pages, you need to design the optimal way of storing pages in the buffer to minimize the number of reading pages from the disk.

3.2 Input format

The first line contains two integers n and m ($n, m \leq 100000$). The second line contains an array A with n integers, meaning that the ith called page is the page with id A_i ($1 \leq A_i \leq 10^5$).

3.3 Output format

One line contains one integer, representing the minimal number of reading pages from the disk.

3.4 One example

Input:

```
5 2
1 2 3 2 4
```

Output:

```
4
```

Explanation: except for the A_4, other pages must be read from the disk.

3.5 Limitations

Time limit: 1s
Memory limit: 256MB
1-6 test points: $n, m \leq 1000$
7-10 test points: $n, m \leq 100000$