CSE 444 – Homework 1
Relational Algebra, Heap Files, and Buffer Manager

Name: Yuanfeng Li

Question	Points	Score
1	10	
2	15	
3	25	
Total:	50	

1 Simple SQL and Relational Algebra Review

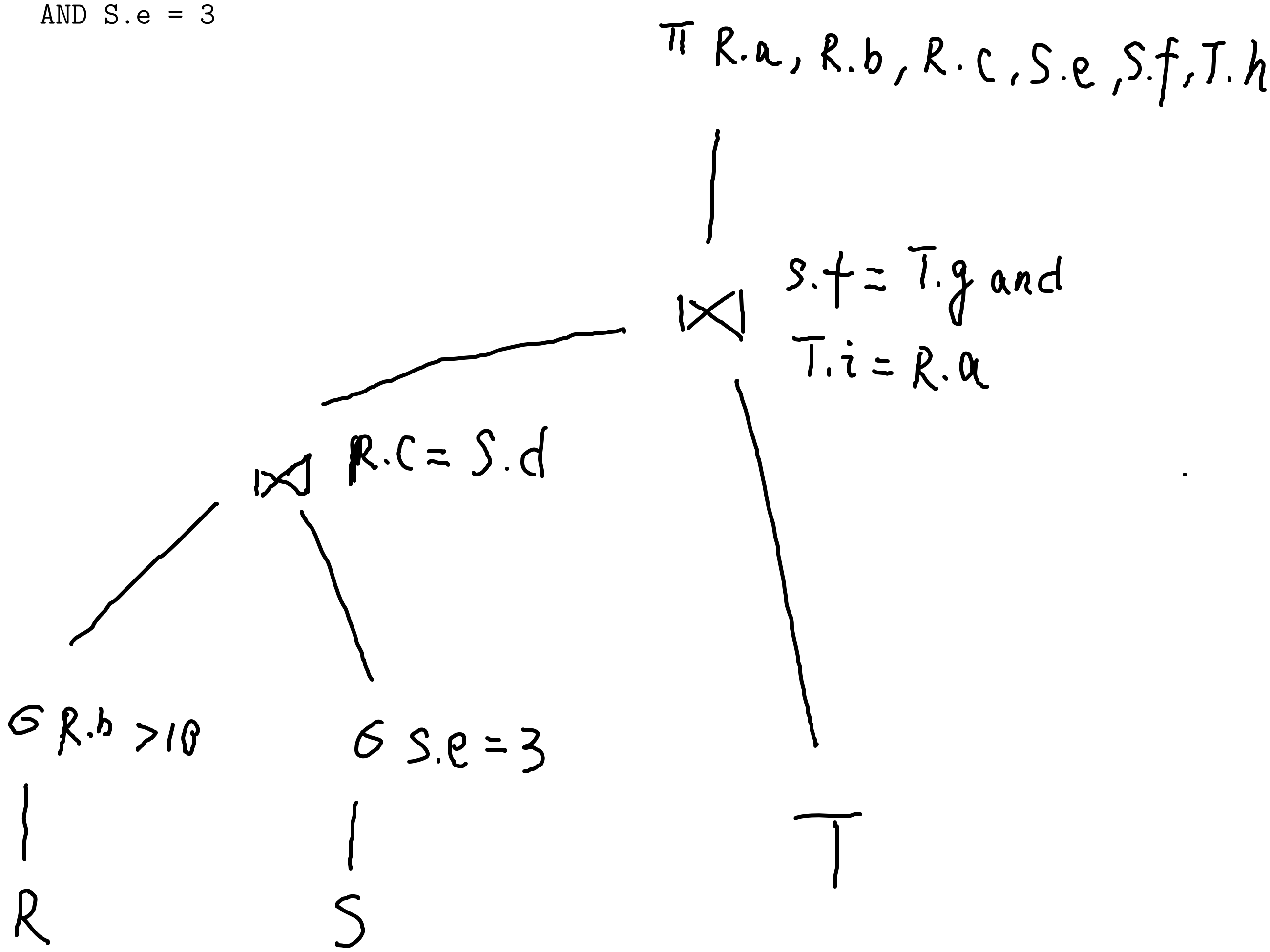
1. (10 points)

   When a user (or application) submits a SQL query to a relational DBMS, the SQL query takes the form of a *string*. Through a series of steps, the DBMS translates this string into a logical query plan. In this exercise, we practice manually translating two simple SQL queries into logical query plans.

   Consider relations `R(a,b,c)`, `S(d,e,f)`, and `T(g,h,i)`.

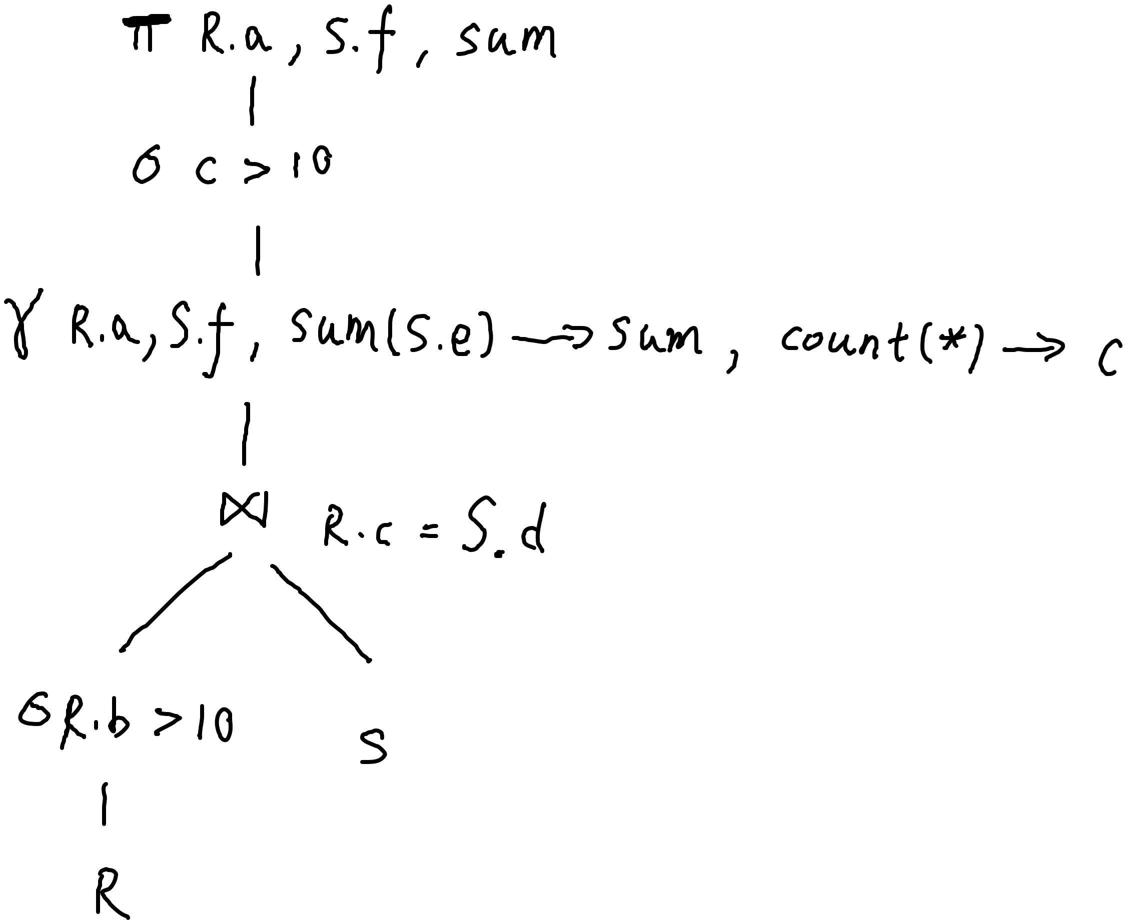
   (a) (5 points) Write a Relational Algebra expression in the form of a *logical query plan* that is equivalent to the SQL query below.

   ```
   SELECT R.a, R.b, R.c, S.e, S.f, T.h
   FROM R, S, T
   WHERE R.c = S.d
   AND S.f=T.g
   AND T.i=R.a
   AND R.b > 10
   AND S.e = 3
   ```



(b) (5 points) Write a Relational Algebra expression in the form of a *logical query plan* that is equivalent to the SQL query below.

```
SELECT R.a, S.f, sum(S.e) as sum
FROM R, S
WHERE R.c = S.d
AND R.b > 10
GROUP BY R.a, S.f
HAVING count(*) > 10
```



2 Data Storage: Heap Files

2. (15 points)

   Consider a relation S with the following schema:

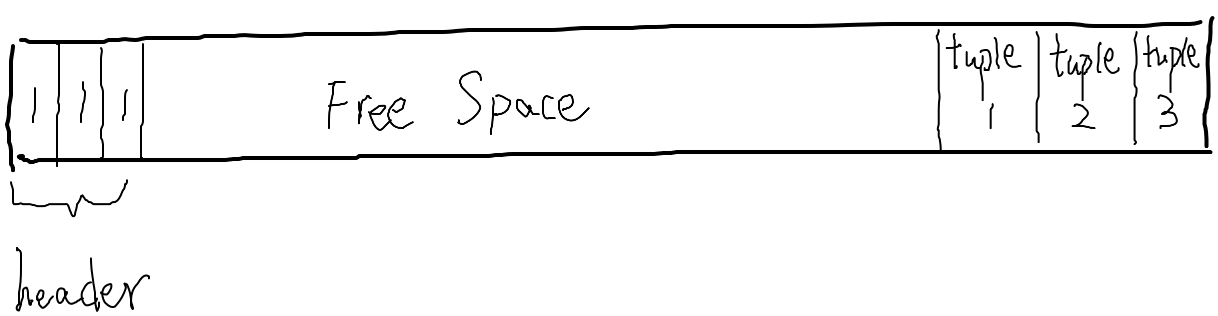
   ```
   S(a int, b char(10), c char(20))
   ```

   Consider the following small instance of S:

a	b	c
1	Orange	First
2	Red	Second
3	Blue	Third

   Assume that S is stored in a Heap file on disk. Assume also a page size of 8KB (8192 bytes) and 4-byte integers.

   (a) (5 points) Assume the same on-disk representation as used in SimpleDB (please refer to the lab1 instructions), draw a *schematic* representation of the Heap file **page** on disk storing the S instance. For examples of what to draw, see lecture 4, slides on "Page Formats", but check the SimpleDB documentation to figure out the format of the page that you should draw! Fill in the fields to show the three tuples on the page. Show where the empty space is on the page (if any). Show padding/unused space (if any). Do not worry about endianness nor about getting all proportions right. If you want, you may specify byte offsets for the records but you do NOT have to show that information.



(b) (5 points) The size of each S tuple in bytes is 4+10+20 or 34 bytes (this is different form SimpleDB where all strings are 128+4 bytes in length). How many S tuples fit on one page? What is the size of the page *header* in bytes? How many pages are necessary to hold an instance of S with 1000 tuples (remember the space necessary for the page header)?

There can be 234 S tuples fit on one page.

According to Simple DB documentation

tuples Per Page = (Page SIZE X 8 bits) / (tuple size X 8 bits + 1)

= 8000 bytes X 8 bits / 34 bytes X 8 bits + 1

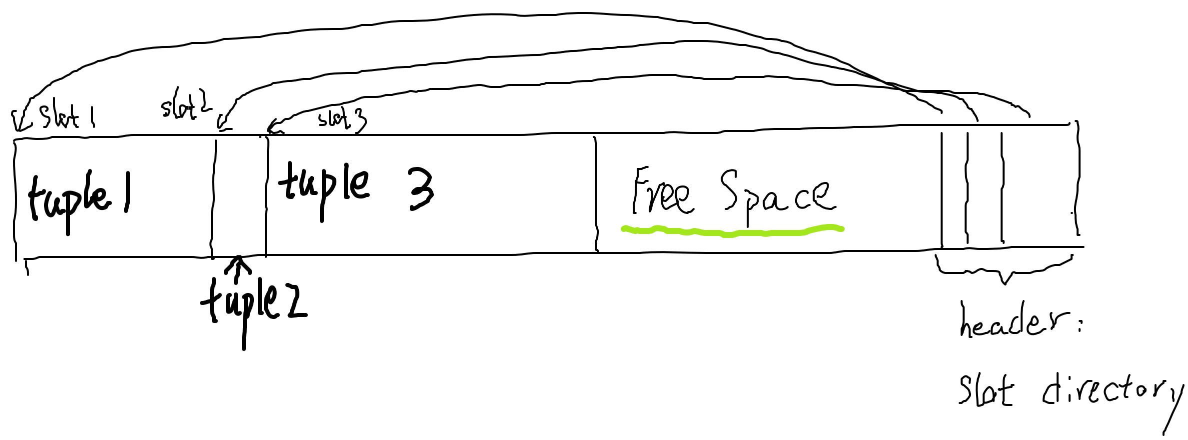
= 234.4

≈ 234 tuples

header Size = 234 X 1 bit = 234 bits ⇒ $\frac{234 \text{ bits}}{8 \text{ bits}}$ = 29.25 bytes ≈ 30 bytes

pages = $\frac{1000 \text{ tuples}}{234}$ = 4.27 ≈ 5 pages

(c) (5 points) Now imagine that we wanted to extend SimpleDB to support *variable-length* tuples, draw a *schematic* represenation of the modified Heap file page on disk storing the S instance (the one with the three tuples). Assume the strings are now variable lengths (VARCHAR). Do not worry about the detailed representation of the records themselves, though.



3 Buffer Manager and Simple Query Execution

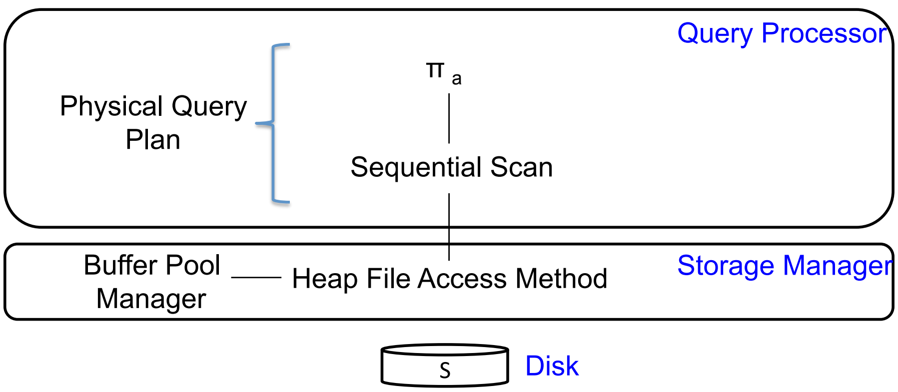
3. (25 points)

Consider a buffer pool large enough to hold 10 pages. The buffer pool is initially empty. Consider also a buffer pool manager that uses an LRU page replacement policy.

Consider the same S relation as in the earlier question and consider the following SQL query:

```
SELECT S.a
FROM S
```

A simple *physical query plan* for this query is the following. We show the query plan in the context of the relevant DBMS architecture components:



To execute the query, the system will call `open()` and then `next()` on the project operator. We ignore `hasNext()` in this exercise.

Consider that relation S is stored in a heap file on disk.

(a) (5 points) Explain how the execution of this query will proceed as the system calls `open()` and then `next()` on the topmost, project operator. You only need to describe what happens on the call to`open()` and then on the first call to `next()`. You do not need to describe subsequent calls to `next()`.

Your explanation should describe the control flow (who calls whom and when) between (1) the project operator, (2) the sequential scan operator, (3) the heap file access method, and (4) the buffer pool manager. Similar to the SimpleDB design, consider that the buffer pool manager will call the heap file to actually read a page from disk.

π_a open( ), next( ) 1 step

Seq. Scan open( ), next( ) 2 step

4 step bp.getPage()

Buffer Pool Manager

Heap File Access Method open( ), next( ) 6 step

heap. ReadPage( ) 5 step

2) In general the execution of this query will be proceed as following:

1. System calls open( ) & next ( ) on project operator $\pi_{S.a}$.
2. projector ask Sequential Scan to call open( ) & next( ).
3. Sequential Scan calls the open( ) in Heap File Access Method

4& 5. 4 HeapFile must calls getPage( ) from Buffer Pool Manager to ensure that these pages get loaded in memory) 5 buffer pool calls readPage( ) from Heap File Access.

6. Onece HeapFile get the page from main memory, it will open( ) it and execute next( )

(b) (5 points) What will be the content of the buffer pool after the first `next()` call on the project operator returns a tuple.

It will be the first page loaded in main memory that returned the first tuple to the next() operator.

There will be only one page in the buffer Pool now because it only called next() once.

(c) (5 points) What will be the content of the buffer pool after the second `next()` call on the project operator returns a tuple.

- This answer is the same answer for (b).

  There will be only one page in the buffer pool, because each page can contain more than two tuples (234 in simpledb), so it won't need to read another page into the buffer Pool.

- However, if the pages in this query is different from the page from simpledb. Then there might be maximum 2 pages as content in buffer pool. In which situation, that the first page only has one tuple in it.

(d) (5 points) Assuming S contains 10,000 tuples, what will be the content of the buffer pool after 1000 next() calls on the project operator.

If each page is 8 Kb large and use the same schematic from SimpleDB.

Then the content will be 5 pages that recently loaded into the main memory. Because each page can store 234 tuples in it, so 1000 next calls 1000 tuples, $\frac{1000}{234} = 4.27 \approx 5$ pages.

The buffer pool is large enough to hold 10 pages, so all 5 pages will be hold as content in buffer pool.

(e) (5 points) Assuming S contains 10,000 tuples, what will be the content of the buffer pool after the query completes?

It will be the last 10 pages that loaded to main memory, because the buffer Pool can only hold 10 pages.

And 10,000 tuples needs $\frac{10,000}{234} \approx 42.7 \approx 43$ pages to store. But buffer pool is only large enough for 10 pages, and using LRU method, all the other 33 pages will be discarded