COL362 / COL632 Minor 2

Divyanshu Agarwal

TOTAL POINTS

19 / 30

QUESTION 1

1 1.a **2 / 2**

✓ **+ 1 pts** *Created correct joins: 1 mark*

✓ **+ 0.5 pts** *Created correct projection*

✓ **+ 0.5 pts** *Identified the joins*

QUESTION 2

2 1.b **0 / 4**

✓ **+ 0 pts** *incorrect/not attempted*

QUESTION 3

3 2.a **2 / 2**

✓ **+ 1 pts** *Approach*

✓ **+ 1 pts** *Issues + Solutions*

QUESTION 4

4 2.b **1 / 2**

✓ **+ 1 pts** *correct answer with reason*

QUESTION 5

5 2.c **3 / 3**

✓ **+ 1 pts** *Challenge 1 correct*

✓ **+ 1 pts** *Challenge 2 correct*

✓ **+ 1 pts** *Challenge 3 correct*

QUESTION 6

6 3.a **0 / 3**

✓ **+ 0 pts** *Incorrect*

QUESTION 7

7 3.b **0.5 / 3**

✓ **+ 0 pts** *INCORRECT*

+ 0.5 *Point adjustment*

QUESTION 8

8 4 **3 / 3**

✓ **+ 3 pts** *Correct Structure*

QUESTION 9

9 5.a **3 / 3**

✓ **+ 1.5 pts** *partial Correct part a*

✓ **+ 1.5 pts** *full partial correct a*

QUESTION 10

10 5.b **3 / 3**

✓ **+ 1.5 pts** *partial Correct part b*

✓ **+ 1.5 pts** *full partial correct b*

QUESTION 11

11 6 **1.5 / 2**

✓ **+ 1.5 pts** *Partially Correct pseudo code*

Pseudocode for handling the case when output buffer is full is missing

COL 362 / COL 632 - Minor2 Examination

25th March 2023

Exam time: 08:00 IST – 09:00 IST

Duration: 1 hour **Total Points: 30**

Answer all questions.

This is **NOT** an open-book exam and **NOT** a collaborative exam. You are expected not to seek help from any other person(s) directly or indirectly (through web, forums, emails, messages etc.) for answering these questions.

Honor Code

I acknowledge the IITD Honor Code and confirm that the answers I have written for this exam are entirely my own.

1. I have **not** used the help of any person, organization or discussions during the examination in answering.
2. I have **not** given or received assistance in either answering, or providing specimen answer/hint/diagram/model/code during the exam.

I am fully aware that if found to have used unfair and disallowed practices in the examination, I am liable for strict action, which could include receiving a failing grade in this course, as well as other actions as deemed fit by the institute disciplinary committee.

Name: DIVYANSHU AGARWAL

Entrynumber: 2020CS10343

Signature: Divy

Date: 25/03/~~2003~~ 2023

Instructions

- **Please make sure you have read the honor code in the front page, and accepted it by filling all the details.**
- **Please make sure you have no reference material, mobile phone, laptop, or any other device that could be construed to help you to answer this exam**
- Questions are made as unambiguous as possible. In case you need, make appropriate and meaningful assumptions and state them clearly (only if required).
- **Answer only within the space given below each question.** If you need correct an answer, then you must neatly cross-out the incorrect answer you have filled, and write it separately at the end (write the question number correctly).

Questions

Question 1 ..... *6 points*

Consider the relation definition in SQL and the query below (also note the additional information given in comments):

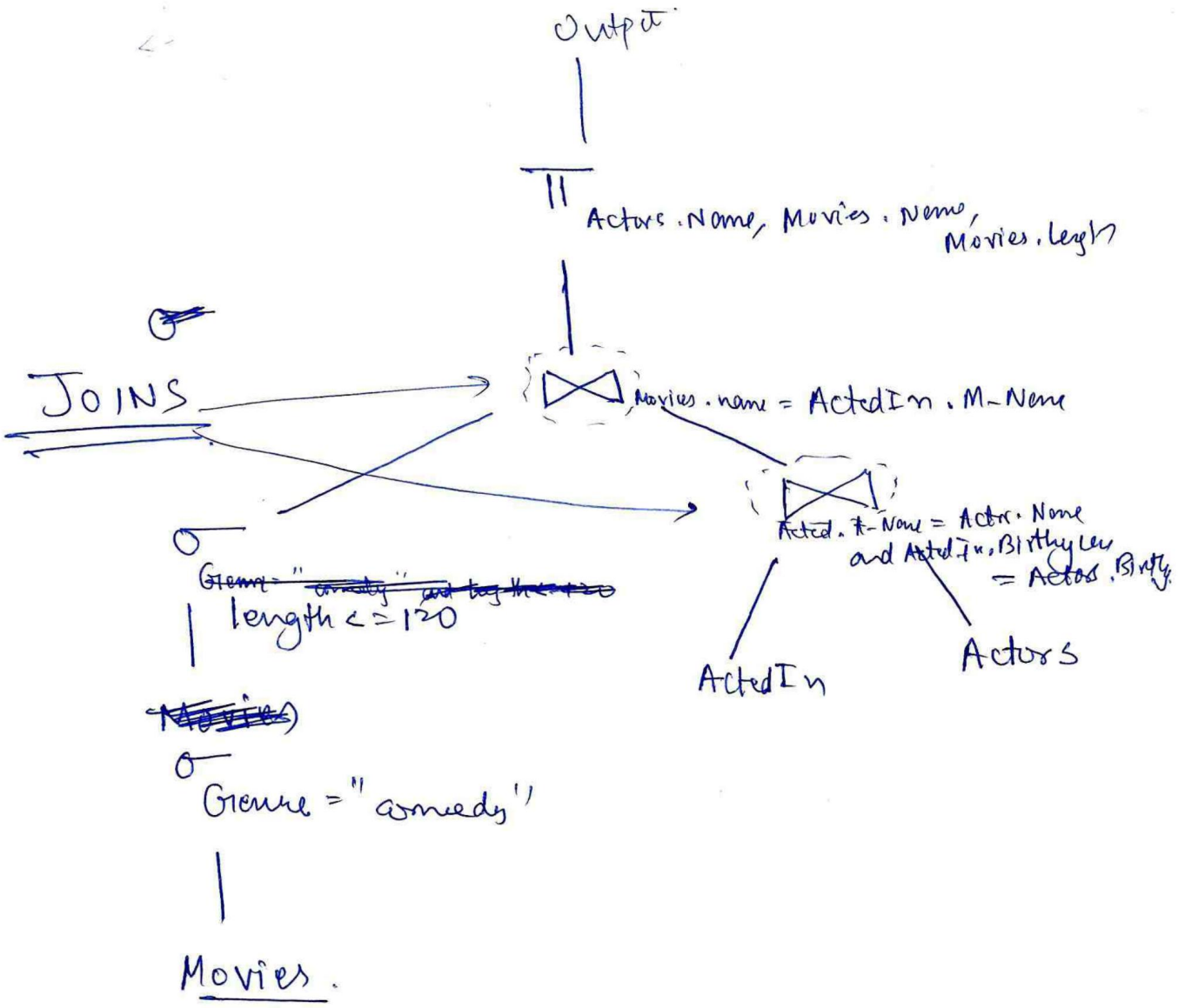
```
CREATE TABLE Movies (
     Name varchar(30),
     Genre varchar (5) NOT NULL,
        /*Genre are from {'Comedy', 'Action', 'Drama', 'Romance'}*/
     Length numeric (3,0),
        /* Length ranges from 50 -- 200 */
     PRIMARY KEY (Name)
);
CREATE TABLE Actors (
     Name varchar(256),
     Birthyear numeric (4,0), /* Birthyear > 1940 */
     Location varchar(256),
     PRIMARY KEY (Name, Birthyear)
);
CREATE TABLE ActedIn (
     A_Name varchar(256),
     A_Birthyear numeric(4,0),
     M_Name varchar(30),
     CONSTRAINT FK_Actor FOREIGN KEY (A_Name, A_Birthyear)
        REFERENCES Actor(Name, Birthyear),
     CONSTRAINT FK_Movies FOREIGN KEY (M_Name)
        REFERENCES Movies(Name)
);


-- Data loading scripts not shown
-- Actors table contains 1,000 tuples
-- Movies table contains 10,000 tuples
-- ActedIn table contains 35,000 tuples


SELECT Actors.Name, Movies.Name, Movies.Length
  FROM Actors, Movies, ActedIn
  WHERE
     Genre='Comedy' AND Length <= 120 AND
     ActedIn.M_Name = Movies.Name AND
     ActedIn.A_Name = Actors.Name AND
     ActedIn.A_Birthyear = Actors.Birthyear;
```

2020CS10343

2 1(a) Give the logical plan diagram of the SQL query given above. Identify the join(s) in the query.



4 1(b) Consider the joins and selections in the above logical plan (separately). Present **one** *physical* query plan for each operator type (i.e., any one join, any one selection, etc.), and derive its cost in terms of number of disk block transfers and disk seeks. Note that the count of tuples in each table and range of values for some attributes is given in the above given SQL code above (in comments). In your calculations, assume that each disk-block size is 512 bytes. Make appropriate additional assumptions if required. Write your assumptions clearly and explicitly.

We can have our ~~Data~~ Movies sorted according to movie.length & grouped on Genre

For the σ Genre = "Comedy" we can have index based ~~Selection~~ Selection on Genre. (dense, clustered)

→ now the table we would itself be sorted by length then σ length < 120 ,

Can be done ~~again~~ using ~~a dense clustered~~ ~~to find first~~ going to length = 50 by a dense clustered index then traverse B+ tree pointers using leafnodes till length exceeds 120.

Both the Joins can be done using nested loop join.

2020CS10343

(...contd.)

Question 2 ..... *7 points*

2(a) In most of our discussion of B^+-Trees, we assumed that the tree is being built on a single key, but we mentioned that it can be adapted for multiple keys to be indexed together. How do you design a B^+-Tree index on multiple fields? What complications do you foresee, and how can they be handled efficiently?

Suppose we have to build a hash tree on attributes $a_1, a_2, \ldots a_k$.

Instead of storing keys with just a single value we can store keys as a tuple of k values. and we can define an order on these tuples, so that we can compare tuples $(b_1, b_2, \ldots b_k)$ and $(a_1, a_2, \ldots a_k)$

The complications I foresee is that the block size will be able to accomodate much fewer of these tuples and thus n would be lower, h would be higher. Comparisons would also take higher. They can be handled by having smaller no. of fields or allowing each level to only 1 attribute like in a k-d tree.

2020CS10343

[2] 2(b) Is it possible to construct a dense index using B+-Tree? If yes, explain how. If no, explain why not?

Yes, it is possible to construct a dense index using B+Tree.

We can have pointers from the leaf nodes to relevant records in the table.

We can make sure we begin with having all the ~~key, value) relation for a~~ (pointer, key) pairs for our relation in the leaf nodes of the B+ tree.

3 2(c) List and explain the challenge(s) one faces when trying to build a B^+-Tree on variable length keys (for e.g., a person name).
Note that due to variable length nature of the attribute, it is possible that in the worst-case attribute definition may be such that its length is nearly $\frac{1}{3}$ or even $\frac{1}{2}$ of the B^+-Tree page.

A B+ tree works efficiently because we restrict each node to have atleast ~~$\lceil \frac{n}{2} \rceil$ etcit~~ $\lceil \frac{n+1}{2} \rceil$ child pointers.

So that we can guarantee the overall height of the B+tree tobe $\leq \lceil \log_{(\frac{n}{2})} \text{tr} \rceil$
And thus we have a guarantee of # of block seeks & transfers.

n remains fixed when the key length is fixed since we can fit a fixed amount of pointer, key pairs in a block.

But if the key is variable length we might only be able to fit ~~1 or 2~~ much lesser of these
and the tree height would grow much larger.
we might need to do upto br block seeks & accesses in the worst case.
Moreover, the insertion and deletions are designed to be efficient ~~since~~ based on the fact that the leaf nodes are atleast halffull.
If that is not the case we would have to redesign the insertion and deletion, they would be much more costly.

2020CS10347

Question 3 ..... *6 points*

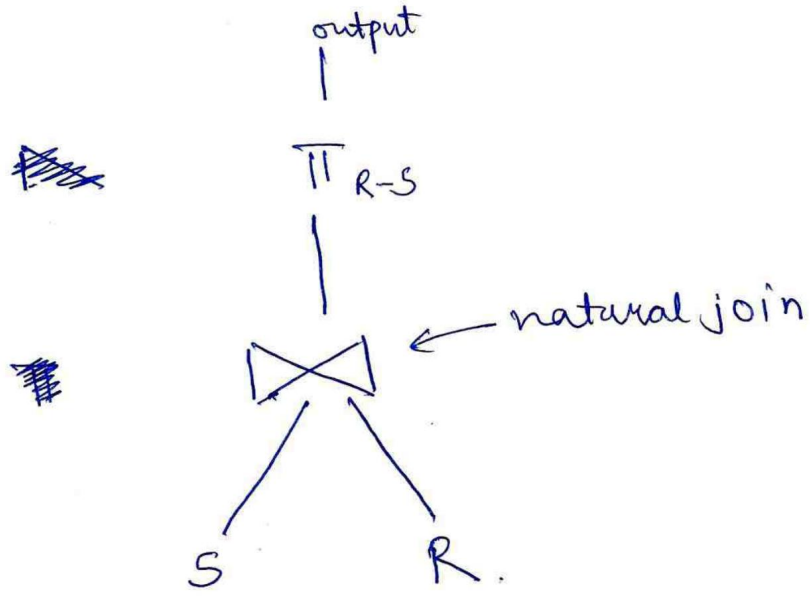
The division operator of relational algebra, "÷", is defined as follows. Let r(R) and s(S) be relations, and let $S \subseteq R$; that is, **every attribute** of schema S is also in schema R. Given a tuple t, let t[S] denote the projection of tuple t on the attributes in S.
Then r÷s is a relation on schema R−S (that is, on the schema containing all attributes of schema R that are not in schema S).

A tuple t is in r÷s if and only if **both of two conditions hold** d:

- $t \in \Pi_{R-S}(r)$
- For *every tuple* $t_s \in s$, there is a tuple $t_r \in r$ satisfying both of the following:
  1. $t_r[S] = t_s[S]$ (i.e. projection of attributes of tuple t_r and t_s on schema S is equal)
  2. $t_r[R-S] = t$ (i.e., the output tuple is only the projection of attributes in R but not in S)

Based on this, answer the following:

3 3(a) Write the above operator using only selection, projection and join relational operators.



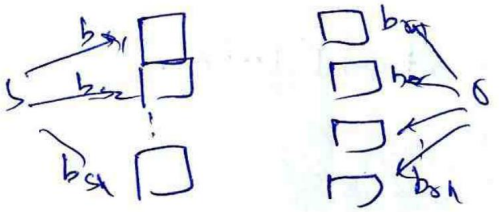
We first natural join S & R, since ~~we~~ every tuple is S is also in R, we essentially get all tuples such that $t_r[S] = t_s[S]$

we then ~~s~~ project to get only R-S attributes

3 3(b) Design a hash-based algorithm for evaluating the division operator (defined above). Explain its working through a pseudo-code, highlighting the disk I/O steps.

~~for i in 1 to b_s~~
~~load~~ b_s to M
~~for tuples i=1 to t_{b_s} in b_i~~
Has

Hash the attributes on relation R and S
load all b_r, b_s separately.
~~and~~ and get k buckets of records from R, S

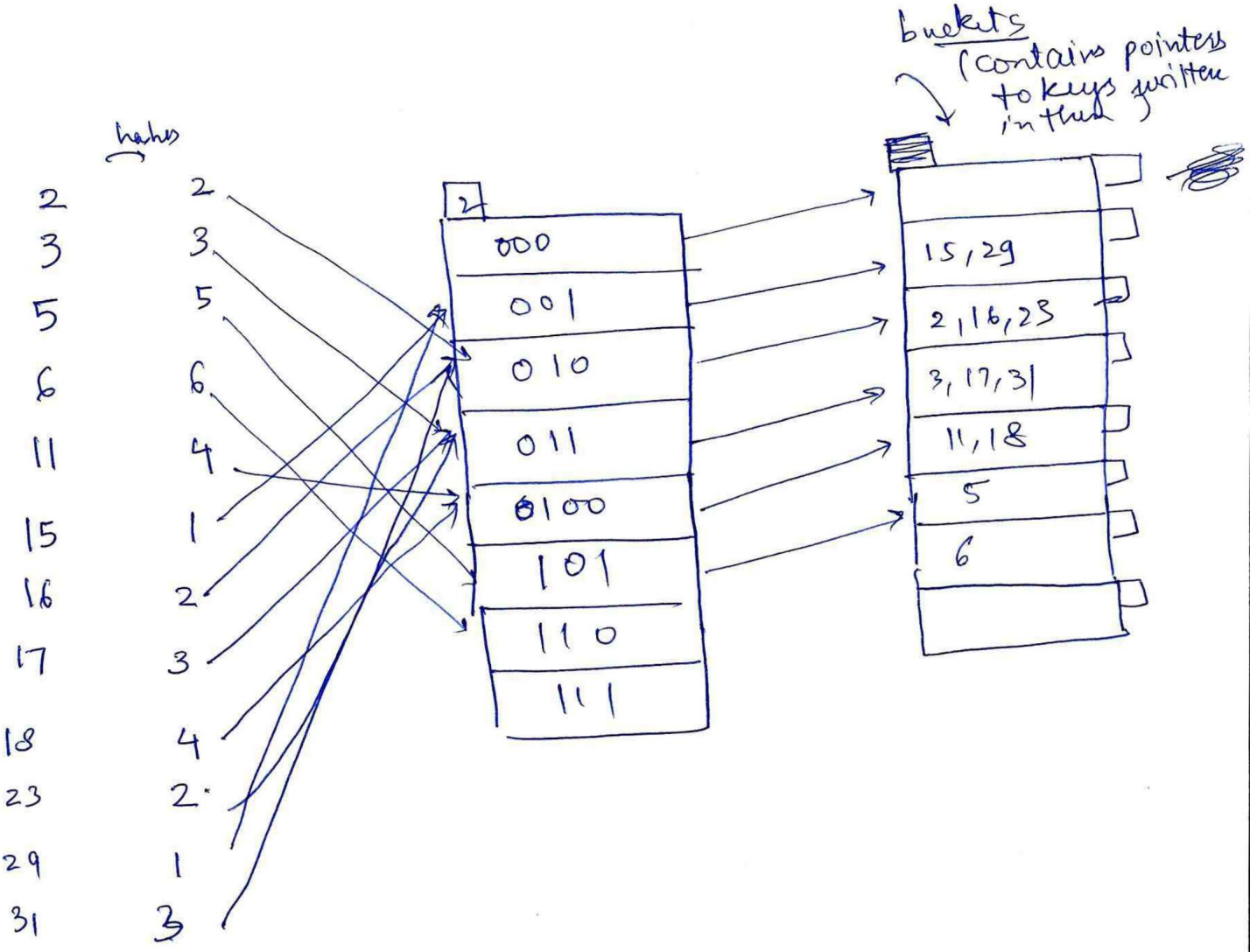


b_{s1} b_{s2} - - b_{sk} (~~dis~~ write to disk.
b_{r1} b_{r2} - - b_{rk}

now doa join of bucket
b_{si} with b_{ri} ← load bucket from disk
such that ~~r=~~ $t_r = t_s$
~~and~~ then ~~w~~ output
t_{R-S}

Question 4 ..... *3 points*

Suppose you are given with the hash function h(x) = x%7 (that is, the remainder when x is divided by 7). Show the hash index on inserting the following sequence of values: 2,3,5,6,11,15,16,17,18,23,29,31. Assume that the method of constructing the hash index is *extendible hashing* and that each bucket can store 3 values. (*Note:* Only the final structure needs to be shown. There will be no partial credit for this question.)



Question 5 ..... *6 points*

Give an example of a relational-algebra expression and a query-processing strategy in each of the following situations, and justify your answer briefly:

3 5(a) MRU is preferable to LRU.

when we are computing join of two tables using nested-loop join, then MRU performs better than LRU because once a block has been used it is again used after 1 whole iteration. so the most recently used will be used the most later.

3 5(b) LRU is preferable to MRU.

when we are computing join (⋈) of tables but using an indexed based join.., and the indexing is done using B+ trees.

We might require repeated access to the same blocks for the inner table as we would search through a B+ tree and the upper few levels like the root & its few children would be repeatedly required. So LRU would perform better as used previously will be required again and again.

Question 6 ..... *2 points*

Write the pseudo-code for the indexed nested-loop join for data residing on disk. The pseudocode must describe the reading and writing of blocks from disk. (Note: No partial marks will be awarded for this question. Unless your pseudo-code is correct, only 0 marks will be awarded).

I am considering tables r and s with an index on table s. The index is assumed to be dense clustered and t_r tuples r, t_s tuples in s.

~~for i in 1 to t_r~~ read

& I am assuming a block ~~of~~ stores k tuples (for relation r)

```
for i in 1 to br
    load bi into M
    for tuples 1 to K.                    # searching tuples in bj
        search(ti.A)                      ## search for ti.A
        if (found)
            load bj into M
            for t = # first matching tuple in bj
            while(t.A == ti.A)            # match
                write((ti, t))
                t++
```

End of the paper