National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	Advance Database Concepts	Course Code:	CS4064
Degree Program:	BS (Computer Science)	Semester:	Spring 2023
Exam Duration:	60 Minutes	Total Marks:	25
Paper Date:	Mon 10-Apr-2023	Weight	15%
Section:	All	Page(s):	1
Exam Type:	Midterm-II	Total Questions:	4

Instruction/Notes: **Solve the questions in the given order.**

You will not get any credit if you do not show proper working, reasoning and steps as asked in question statements.

Q1. *(6 points)* Assume a relation R (A, B, C) is given; R is stored as an ordered file (un-spanned) on non-key field C and contains 100,000 records. Attributes A, B and C need 10 bytes of storage each (i.e., record size is 30 bytes), and blocks have a size of 1024 bytes. Each A value occurs at an average 5 times in the database, each B value occurs 50 times in the database, and each C value occurs 5000 times in the database. Assume there is no index structure exists. Estimate the number of block fetches needed to compute the following queries (where C_a, C_b, C_{c1} and C_{c2} are integer constants):

a. SELECT COUNT(*) FROM R WHERE B = C_b;

b. SELECT A, B FROM R WHERE C = C_{c1};

c. SELECT A, B FROM R WHERE C = C_{c1} OR C = C_{c2};

Ans: bfr=1024/30=34; b=100,000/34= 2942

a) O(b) = 2942

b) O(log(b) + s/bfr - 1) = O(12 + 5000/34 – 1) = O(12 + 148 - 1) = 159

c) 2 * 159 (i.e., same cost as of part-c) = 318

Q2. *(9 points)* Assume that you have just built a dense B^+-tree index on a heap (unordered) file containing 1,000,000 records. The key field for this B^+-tree index is a 30-byte string, and it is a candidate key. Pointers (Record/block) are 10-byte values. The size of one disk page is 1024 bytes. The index was built in using the bulk-loading algorithm, and the nodes at each level were filled up as much as possible.

a. How many levels does the resulting tree have? Show your working.

b. For each level of the tree, how many nodes are at that level?

c. How many levels would the resulting tree have with all nodes 75% fill factor?

Ans: see note book.

order p = 26; (p * 10) + ((p - 1) * 30) <= 1024

order p_{leaf} = 25; (p_{leaf} * (30+10)) + 10 <= 1024

b1= ceiling(1,000,000/25) = 40,000

fo = 26

b2= ceiling(40000/26)= 1539

b3= ceiling(1539/26)= 60

b4= ceiling(60/26)= 3

b5= 1

a. x=5; OR x = ceiling($\log_{fo}$ (b1)) + 1 = ceiling($\log_{25}$ 40000) + 1 = 4 + 1 = 5 levels

b. Lev1(b1)=40000, Lev2(b2)=1539, Lev3(b3)=60, Lev4(b4)=3, Lev5(b5)=1; bi=41603

c. Avg no of keys in leaf nodes= ceiling(0.75*p_{leaf}) =.75* 25 = 19

Avg fo of internal node= fo = ceiling(.75*p) = .75*26 = 20

b1= ceiling(1,000,000/19) = 52,632

b2= ceiling(52632/20)= 2632

b3= ceiling(2632/20)= 132

b4= ceiling(132/20)= 7

b5= 1

x=5 OR x = ceiling($\log_{fo}$ (b1)) + 1 = ceiling($\log_{20}$ 52632) + 1 = 4 + 1 = 5 levels

Q3. *(4 points)* Suppose that the following are the most often used queries on the database. You are required to improve the performance of these two queries. Which column(s) are more appropriate to create indexes on these two large tables? Write down the column name and type of index (i.e., B^+-tree, Hash, or Bitmap). Explain in one sentence why you choose the column(s).

Query1: *SELECT * FROM customer C JOIN order O ON C.CID = O.CID WHERE C.gender ='M';*

Query2: *SELECT * FROM customer C JOIN order O ON C.CID = O.CID WHERE O.date > '01-Jan-2023';*

Customer

CID	Name	Age	Gender
100	Tahreem	19	F
200	Izaan	21	M
300	Isbah	24	F
400	Ismail	50	M

Order

OID	CID	Date
1	400	12-Dec-2022
2	200	25-Dec-2022
3	200	10-Jan-2023
4	300	14-Feb-2023

Ans: Filter columns: O.date (B+-tree), C.gender (Bitmap), Joining Columns: C.CID (Hash/B+-tree), O.CID (Hash/B+-tree).

Q4. *(6 points)* Suppose that we are using extendable hashing on a file that contains records with the following search-key values: 19, 14, 21, 8, 23, 15, 31, 25. Show the extendable hash structures for this file if the hash function is h(k) = k mod 4 and buckets can hold **three** records. Use higher bits (i.e., left to right) of hash value to determine a directory entry.

Ans:

Keys:	**19**	**14**	**21**	**8**	**23**	**15**	**31**	**25**
h(k):	**3**	**2**	**1**	**0**	**3**	**3**	**3**	**1**
binary:	**11**	**10**	**01**	**00**	**11**	**11**	**11**	**01**

d=0

19, 14, 21, ~~8~~

d=1

0	**21, 8**	**0**
1	**19, 14, 23,~~15~~**	**1**

d=2

00, 01	**21, 8**	**0**
10	**14**	**10**
11	**19, 23, 15, ~~31~~**	**11**

Use new hash function i.e., k mod 8 to re-allocate keys:

Keys:	**19**	**14**	**21**	**8**	**23**	**15**	**31**	**25**
h(k):	**3**	**6**	**5**	**0**	**7**	**7**	**7**	**1**
binary:	**011**	**110**	**101**	**000**	**111**	**111**	**111**	**001**

d=0

19, 14, 21, ~~8~~

d=1

0	**19, 8**	**0**
1	**14, 21, 23, ~~15~~**	**1**

d=2

00, 01	**19, 8**	**0**
10	**21**	**10**
11	**14, 23, 15, ~~31~~**	**11**

d=3

000, 001, 010, 011	**19, 8, 25**	**0**
100, 101	**21**	**10**
110	**14**	**110**
111	**23, 15, 31**	**111**