HONG KONG UNIVERSITY OF SCIENCE & TECHNOLOGY
COMP3311: Database Management Systems

Spring Semester, 2017

Final Examination

May 18, 2017 12:30 to 15:30

Name: ___________________ **Student Number:** ______________________

Email: ___________________ **Lecture session:** ____________________

Instructions:

1. This is an open-note examination. You are allowed to bring and use the printed course slides from lectures, tutorials and labs in the examination. No electronic device except a simple scientific calculator is allowed throughout the exam.
2. This examination paper consists of **27** pages and **6** questions.
3. Please write your name, student ID and Email on this page.
4. For each subsequent page, please write your student ID at the top of the page in the space provided.
5. Please answer **all** the questions within the space provided on the examination paper. You may use the back of the pages for your rough work and afterwards drawn a diagonal line through it to show that it is not part of your answer.
6. Please read each question very carefully and answer the question clearly and to the point. Make sure that your answers are neatly written, readable and legible.
7. Leave all pages stapled together.
8. The examination period will last for **3 hours**.
9. Stop writing immediately when the time is up.

Questions	Marks	Scores
1	20	
2	16	
3	16	
4	16	
5	16	
6	16	
Total	100	

Student Number: ___________________

Q1 Multiple-choice Questions (20 marks) Choose the correct answer for each question and put in the boxes.

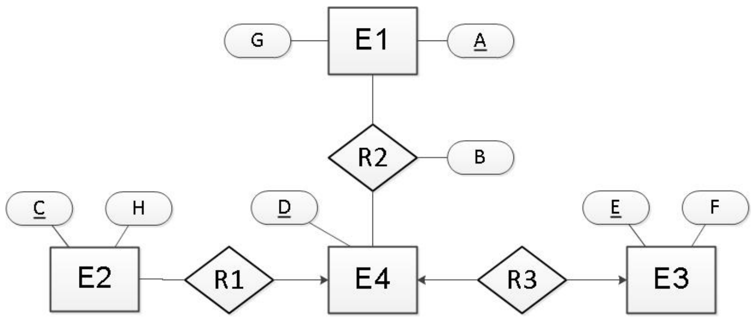
1	D
2	B
3	A
4	A
5	D

6	A
7	D
8	B
9	B
10	C

11	C
12	C
13	D
14	A
15	A

16	C
17	C
18	B
19	B
20	D

1. Referring to the following ERD, which of the following FDs are valid in a possible set of translated relational schemas?



I. E1: A → G

II. E2: C → DH

III. E4: E → D

IV. R2: AD → B

A. I and II only.
B. I, II and IV only.
C. I, II and III only.
D. All of them.

2. Consider the following tables in a university database where the primary keys are underlined and foreign keys are in italic. Student has 1000 records, Course has 100 records and Department has 10 records. The attributes are in usual meaning, such as sid is student ID and sname is student name.

- Student(<u>sid</u>, sname, age, *cid*)
- Course(<u>cid</u>, cname, credits, *did*)
- Department(<u>did</u>, dname)

Which of the following statements about the join result is wrong?

A. $Student \bowtie Course$ has at most 1000 records.
B. $Student \bowtie Department$ has at most 1000 records.
C. $Course \bowtie Department$ has at most 100 records.
D. $Department \bowtie (Student \bowtie Course) = Course \bowtie (Student \bowtie Department)$

3. Consider the university database in previous question again. Suppose *V*(credits, Course)= 5 and *V*(age, Student)=10. Assuming that the records are uniformly distributed and the attributes are independent with each other.

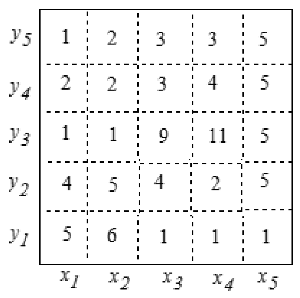
Given a query as follows:

```
SELECT name
FROM Student, Course
WHERE credits = 3 AND age=21
```

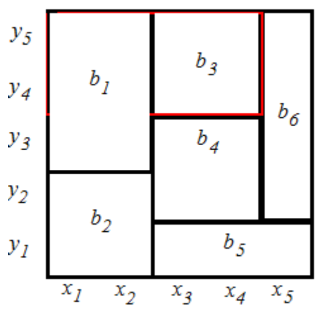
What is the most reasonable expected number of records in the query result?

A. 10
B. 100
C. 2000
D. 100000

4. Consider the multi-dimensional histogram example in the following figure. The left diagram represents the attribute dependence between attributes X = $\{x_1,x_2,x_3,x_4,x_5\}$ and Y = $\{y_1,y_2,y_3,y_4,y_5\}$. The right diagram represents the collection of the records in a set of buckets B = $\{b_1,b_2,b_3,b_4,b_5,b_6\}$ in the database system. The number of records in each bucket may be different, which is the information available for query size estimation. For example, b_1 has 9 records and b_3 has 13 records. Which of the following statements about the estimation of the number of records in the query window "where $y_1 \le y \le y_2$ and $x_3 \le x \le x_5$" is the best one?



y_5	1	2	3	3	5
y_4	2	2	3	4	5
y_3	1	1	9	11	5
y_2	4	5	4	2	5
y_1	5	6	1	1	1
	x_1	x_2	x_3	x_4	x_5



A. 21

B. 26

C. 49

D. None of the above

5. Let r and s be two relations having size b_r and b_s pages. Suppose the hash function generate n buckets. Which of the following statements about applying hash join on r and s is **false**?

A. We always partition r and s in the first phase.
B. We can partition r and s with cost $2b_r + 2b_s$ pages if the memory has at least n +1 pages.
C. We can use in-memory hash index on a bucket of either r or s in the second phase.
D. We are able to implement the hybrid hash join if the memory size is larger than the size of a bucket of r.

6. Which of the following statements about indexing a file (a table) are true?

I. We are able to locate a record in the file generally faster by using a dense index than a sparse index.
II. When a new record is inserted, some entries on the dense index file must be updated.
III. We should have more than 1 level when the index is B+ tree.

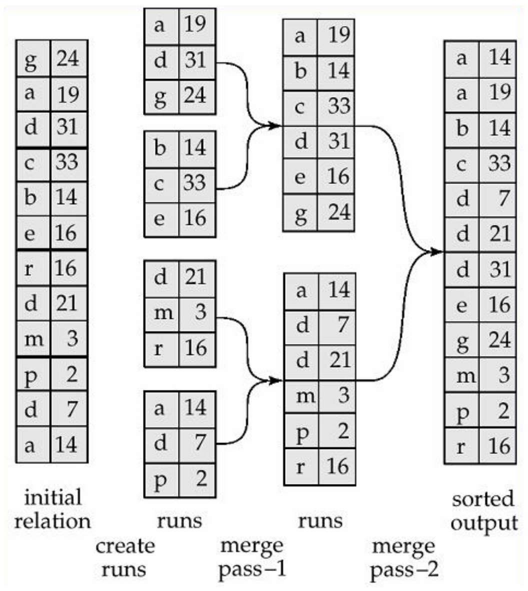
A. I only.
B. I and II only.
C. II and III only.
D. All of them.

7. Which of the following statement(s) about processing the following query optimization is/are **false**: $\pi_{sailor\_id}\, \sigma_{rating=8}(SAILORS)$?

I. The processing cost the query is always reduced when we use an index on rating.
II. Dynamic programming method is not applicable in the query.
III. If we swap the order of projection and selection in the query, we are able have the same answer.

A. I only.
B. I and II only.
C. II and III only.
D. All of them.

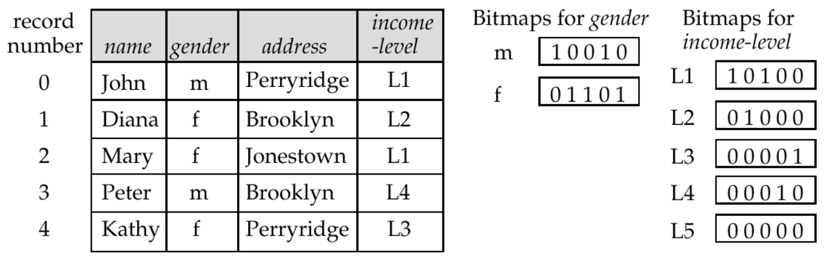
8. Consider an external sort-merge operation in the following example, where we assume 1 record per page and 3 buffer pages in memory. The total cost of running the operation is 60 pages according to the number of runs and merges shown as follows.



Suppose now we have 5 buffer pages. The total cost of running the above operation becomes:

A. Still equal to 60 pages.
B. Smaller than 60 pages.
C. Larger than 60 pages.
D. Cannot process the sort-merge operation by 5 buffer pages.

9. Consider the bitmap example shown in the following figure. Which one is the bitmap operation for finding the records satisfying the conditions “gender = female” and “income level lower than or equal to L2”. Levels L1 is regarded lower than L2.



record number	*name*	*gender*	*address*	*income -level*
0	John	m	Perryridge	L1
1	Diana	f	Brooklyn	L2
2	Mary	f	Jonestown	L1
3	Peter	m	Brooklyn	L4
4	Kathy	f	Perryridge	L3

Bitmaps for *gender*	
m	10010
f	01101

Bitmaps for *income-level*	
L1	10100
L2	01000
L3	00001
L4	00010
L5	00000

A. 01101 AND (NOT 01000)
B. 01101 AND (10100 OR 01000)
C. 01101 AND 10100 AND 01000
D. 10010 OR (10100 AND 01000)

10. We usually do not index every attribute in a database. Which of the following claims are the reasons?

I. Every index requires additional CPU time and disk I/O overhead during inserts and deletions.
II. We are not allowed to define index on some attributes.
III. For queries which involve conditions on several search keys, efficiency might not be bad even if only some of the keys have indices on them. Therefore, database performance is not improved much if many, even not all, indices already exist.

A. I and II only
B. II and III only
C. I and III only
D. All of them

11. Assume an actor database with the following sizes of each attribute "Actor (ID: 4 bytes, Name: 20 bytes, Company: 10 bytes)".

Assume also that the relation is sorted on the name and we want to create a single-level ordered index on ID (the primary key of Actor). There are a total of 160 actors. Each page is 100 bytes and each pointer is 6 bytes.

What is the cost of processing the following query: find the actor with ID=10.

A. 3
B. 4
C. 5
D. None of above

12. Consider a linear hash index with the following characteristics:

- A bucket can hold 50 pairs of <index value, pointer> entries.
- The file being indexed has 1000 records.
- The indexed field does *not* have any duplicates.
- The average occupancy (utilization) is 50%.

What is the cost of the worst-case of searching a record in the above database setting?

A. 1
B. 11
C. 21
D. None of the above

13. Using the database setting in the above question (Question 12) again, we now allow duplicates in the indexed field.

Then the cost of the worst-case of searching a record

A. may be larger than or smaller than the answer of Question 12.
B. should be smaller than the answer of Question 12.
C. should be larger than the answer of Question 12.
D. should be the same as the answer of Question 12.

14. Consider the following schedule:

S = <W2(B), R1(B), W1(A), W2(B), C2, R1(B), C1>.

Which of the following statements is true?

A. S is recoverable but not cascadeless.
B. S is recoverable and cascadeless.
C. S is cascadeless but not recoverable.
D. S is neither recoverable nor cascadeless.

15. Consider the following schedule S = <W1(A), W3(B), R2(B), W2(A), R3(B)>.

Which of the following schedule has the same precedence graph with the schedule S?

A. <W1(A), R2(A), R3(B), W2(B)>
B. <W1(A), R3(A), W2(B), R3(B)>
C. <R3(B), R2(B), R1(A), W2(A)>
D. <R1(A), W3(A), R2(A), W2(B)>

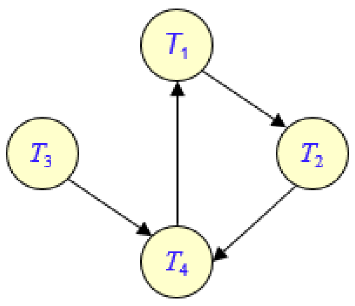
16. Consider the following schedule

S = <W2(A), R1(A), W3(B), W4(C), W4(B), R2(C)>.

Which of the following sequences of commits after S ensures that the schedule is recoverable?

A. C3, C4, C1, C2
B. C2, C1, C3, C4
C. C3, C4, C2, C1
D. C2, C1, C4, C3

17. Which of the following statements is true about the following wait-for graph?



A. The system is not in a deadlock state but would be after adding an edge pointing from T3 to T1.
B. The system is not in a deadlock state but would be after T4 is no longer holding a data item needed by T3.
C. The system is in a deadlock state but would not be after removing edge between T4 and T1.
D. The system is in a deadlock state but would not be after removing edge between T3 and T4.

18. Consider the following schedule:

T1	T2

x-lock(X)
read(X)	

	s-lock(Y)
	{A}
	read(Y)
	write(Y)

write(X)
s-lock(Y)
read(Y)
{B}	

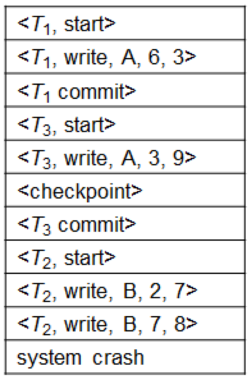
commit T1
unlock(X)
{C}	

	read(X)
	commit T2
	{D}

Find all the positions A to D in the schedule that we can insert ‘unlock(Y)’ according to Rigorous 2PL.

A. {A} {C}
B. {C} {D}
C. {B} {C} {D}
D. {A} {B} {D}

19. Consider the log corresponding to a schedule of three concurrently executing transactions T_1, T_2, and T_3 where a system crash occurs as indicated. Assume that we use the immediate update protocol with checkpointing.

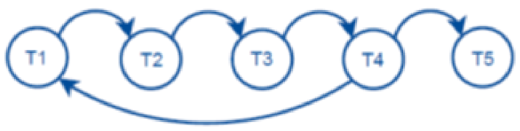


<T_1, start>
<T_1, write, A, 6, 3>
<T_1 commit>
<T_3, start>
<T_3, write, A, 3, 9>
<checkpoint>
<T_3 commit>
<T_2, start>
<T_2, write, B, 2, 7>
<T_2, write, B, 7, 8>
system crash

Which of the following recovery actions on transactions are sufficient when we scan the log backward?

A. T_1 and T_2 to be undone; T_3 to be redone.
B. T_2 to be undone; T_3 to be redone.
C. T_1 and T_2 to be redone; T_3 to be undone.
D. T_2 to be redone; T_3 to be undone.

20. Which of the following schedule corresponds to the given precedence graph?



A. <R2(Y), W3(Y), W1(X), R2(X), W4(Y), R5(X), R3(X)>.
B. <W4(X), R1(X), W1(X), R2(X), R3(X), W5(Y)>.
C. <R1(X), W4(X), W2(X), W5(X), W3(X)>.
D. <W1(X), W2(X), W3(X), W4(X), W5(X), W4(Y), W1(Y)>.

Q2 (16 marks) Join Operations

Consider the following two tables:

- R(A, B)
- S(*A*, C, D, E)

Where

- Primary keys are underlined and foreign keys are in italic.
- Each attribute in R and S is 20 bytes.
- Each page is 800 bytes.
- There are 5000 records (tuples) of R, 10000 records of S.

(a) Fill in the blanks: **(6 marks)**

Relation R contains ________250_____________ pages.

Relation S contains ___________1000_________ pages.

(i) Suppose we want to join (natural join) the two tables and use S as the *outer* relation.

The minimal memory buffer to run the join is ________3___________ pages.

If we have 102 page memory buffer, the cost for using block-nested join is______3500_____ pages.

(ii) Suppose we want to optimize the block-nested join, the minimal memory buffer we should have to achieve the most efficient the block-nested join is ____1002____ pages. (Or 252 if using r as outer relation)

(iii) Suppose the relations R and S have been sorted on A, the cost of running the merge join is _________1250____________ pages.

Consider the following query for Parts (b) and (c) and assume the memory buffer is 59 pages:

```
Select R.B
From R, S
Where R.A = S.A
```

(b) We use hash join to process the above query, in which the hash function generates10 buckets for a given relation. The process is detailed in the following steps: **(8 marks)**

1) We first partition R and write back to the disk. Before writing the R back to the disk, we first remove *unnecessary* attributes of R as many as possible.
2) Then we partition S. Before writing the S back to the disk, we first remove *unnecessary* attributes of S as many as possible.
3) Finally we read each bucket of S (as the built input) and match it against the corresponding bucket of R (as the probe input).

The cost of Step 1 is ______500__________ pages, where the unnecessary attributes are ________________. (Leave it blank if you think all attributes are necessary.)

The cost of Step 2 is _______1250_____________ pages, where the unnecessary attributes are ______C, D, E__________. (Leave it blank if you think all attributes are necessary.)

The cost of Step 3 is _________500____________ pages.

The overall cost of this process is _________2250____________ pages.

If we do not remove unnecessary attributes in Steps 1 and 2, then the overall cost of this process is ________3750_____________ pages.

If we change the select clause in the query from "Select R.B" to "Select R.A", then the overall cost of this process is ________2000___________ pages.

(c) We now use hybrid hash join to further optimize the processing of the query: **(2 marks)**

The process is modified as the following steps:

1) We first partition R and remove *unnecessary* attributes of R as many as possible. We then keep as many as possible the first *x* buckets in the memory and write other buckets back to the disk. (hint: x is some integer from 1 to 10)
2) Then we partition S. Each record that belongs to the first x buckets is matched directly with the corresponding bucket of S (which is still in memory). Before writing other S buckets back to the disk, we first remove *unnecessary* attributes of S as many as possible.
3) Finally we read the remaining bucket of S (as the built input) and match it against the corresponding bucket of R (as the probe input).

The overall cost of this improved implementation is _________2050____________ pages.

The gain (saving) in this optimization is ________200_____________ pages.

Q3 (16 marks) Indexing

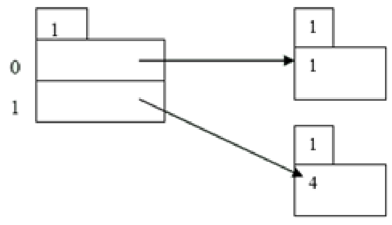
Suppose that we are using extendable hashing on a file that contains records with the following search-key values *K*: 1, 4, 5, 15, 16. Assuming the search-key values arrive in the given order (i.e. 1 being the first coming key and 16 being the last one).

(a) Show the extendable hash structure for each insertion of the above key values file if the hash function is *h*(*K*) = *K* mod 8 and buckets can hold two records. Please follow the convention of extending binary hash indices starting from the most significant bit (For example, the hash value of the key "4" is 100 and the most significant bit is 1). **(8 marks)**

(i) Fill in the following table the hash value of all the keys:

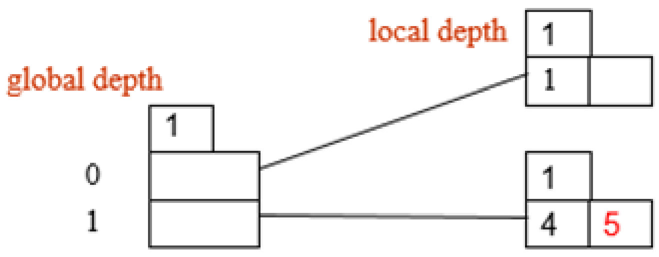
1	4	5	15	16
001	100	101	111	000

(ii) After inserting 1, 4 we have the following configuration:

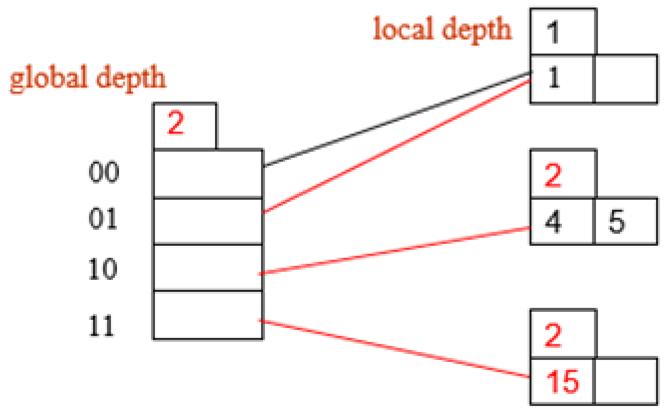


Continue the above configuration by doing the remaining key insertions (i.e. 5, 15,16):

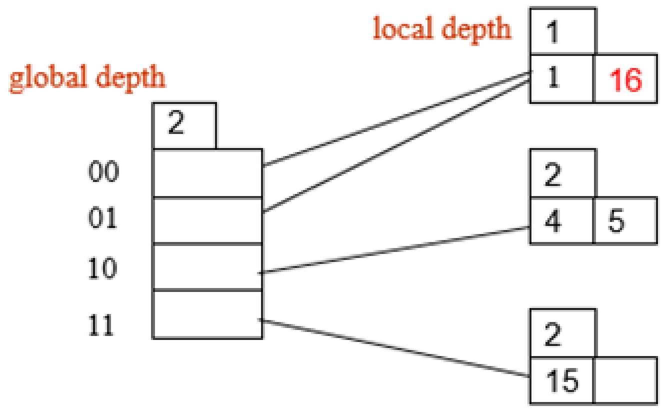
After inserting 5:



After inserting 15:



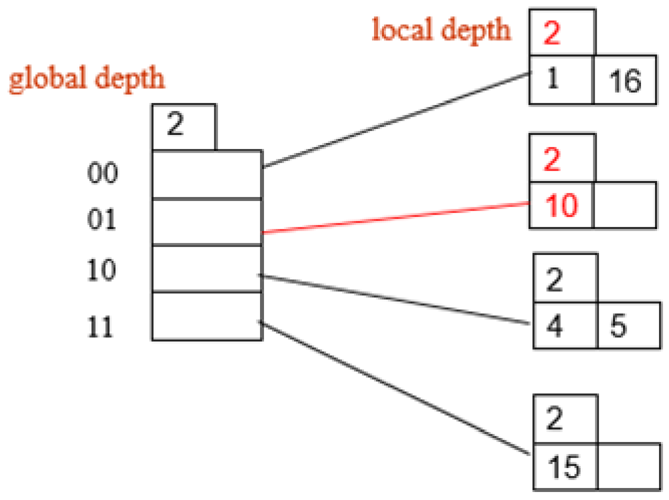
After inserting 16:



(iii) Draw the configuration in Part a(ii) if we continue with the following sequence of operations: (Note that you may need to merge buckets and update the directory after deletion.)

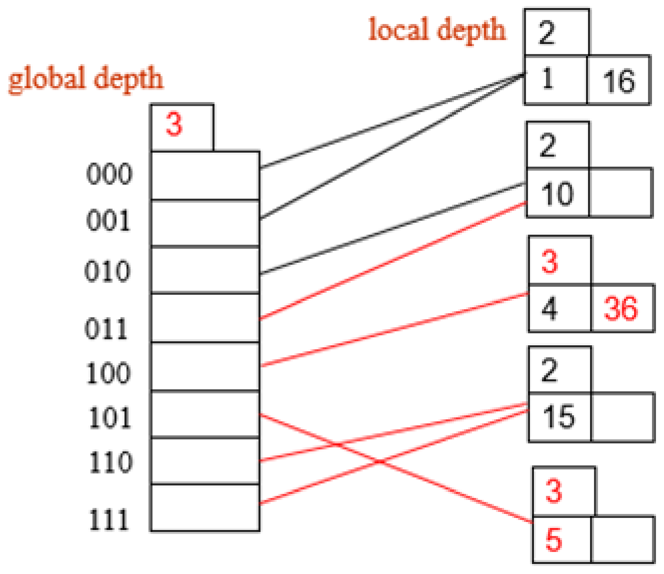
1. Inserting 10
2. Inserting 36
3. Deleting 4
4. Deleting 15

1. Inserting 10:
10 maps to 010, thus:

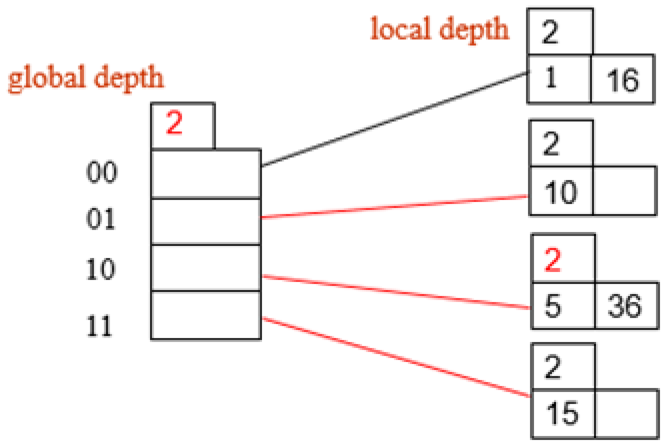


2. Inserting 36:

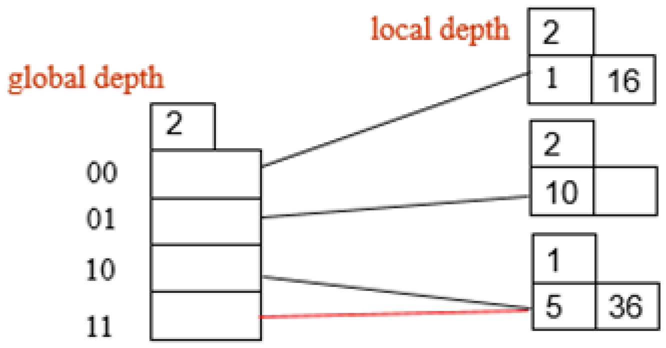
36 maps to 100, thus:



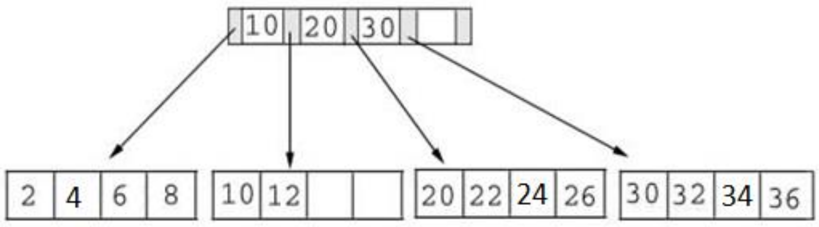
3. Deleting 4



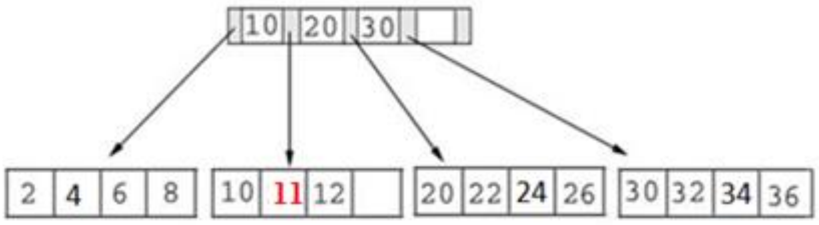
4. Deleting 15



(b) Suppose that we are using the following B+ tree index with *fanout n* = 5. The tree nodes are based on the simplified notation used in the course exercises. Update the tree by performing the sequence operations from (i) to (v) in order. You may draw only the portion of the tree that is different from the previous result for simplicity. **(8 marks)**

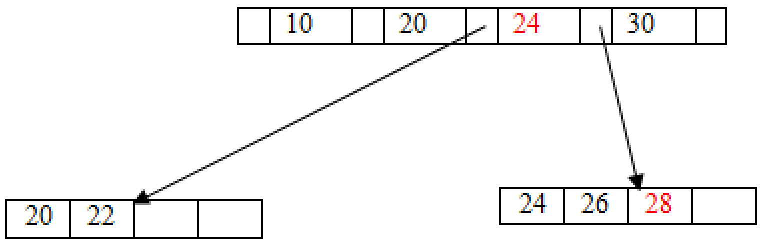


(i) Insert key = 11



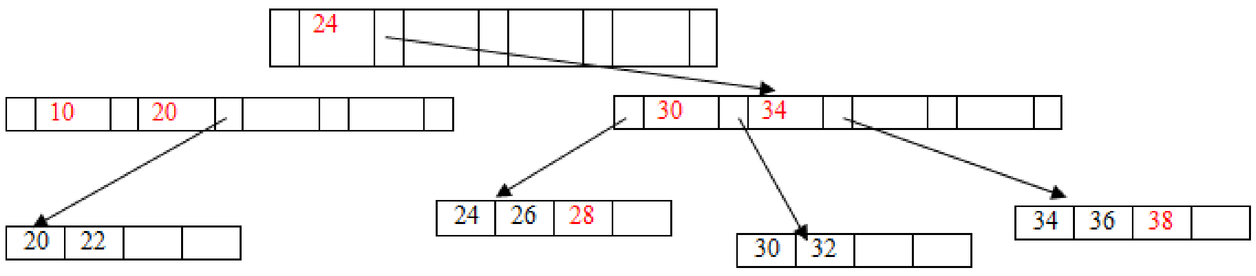
(ii) Insert key = 28

Split the third leaf and insert 28 in root node



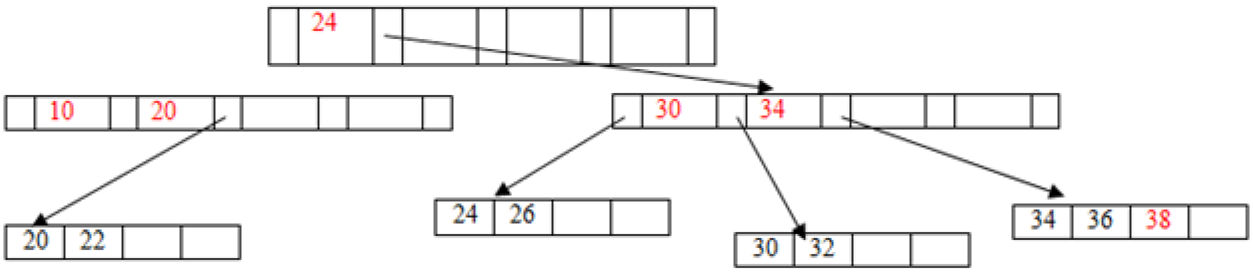
(iii) insert key = 38

Split the fourth leaf, split the root node and create a new root node



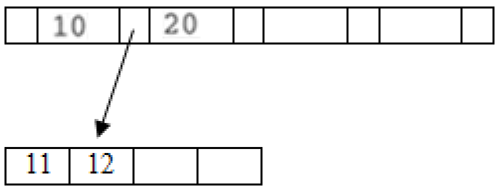
(iv) delete key = 28

delete 28 from fourth leaf, no other change



(v) delete key = 10

delete 10 from second leaf, no change



For reference: according to the algorithms, no change in parent node if the insert or delete does not have overflow or underflow.

Q4 (16 marks) Query Optimization

Consider a database consisting of the following three tables:

- CUSTOMERS (<u>CID</u>, Cname, job, age, street, city)
- BRANCHES (<u>BID</u>, Bname, city)
- ACCOUNTS (<u>*CID, BID*</u>, balance)

The meaning of the attributes in the above schemas is self-explanatory. For example, Bname is the name of the branch. Keys are underlined and foreign keys are in italics.

- There is no index on all the relations.
- The relation CUSTOMERS has 20,000 tuples and 20 tuples of CUSTOMERS fit into one block.
- The relation BRANCHES has 4,000 tuples and 40 tuples of CUSTOMERS fit into one block.
- The relation ACCOUNTS has 24,000 tuples and 30 tuples of ACCOUNTS fit on one block.

(a) Assume that there are 12% of branches which is located in NEW YORK.

Estimate the required size in terms of numbers of tuples $\pi_{BID}(\sigma_{city=\text{"NY"}}\text{BRANCHES})$

Size = _____12 % * 4,000 = 480_______________ tuples. **(4 marks)**

(b) Consider the (natural) join CUSTOMERS $\bowtie$ BRANCHES $\bowtie$ ACCOUNTS.

The following are two strategies for computing the join:

Strategy 1: (CUSTOMERS $\bowtie$ BRANCHES) $\bowtie$ ACCOUNTS
Strategy 2: (CUSTOMERS $\bowtie$ ACCOUNTS) $\bowtie$ BRANCHES

Which strategy is better? Explain the reason(s) with numerical support of your choice.
(If the conclusion stated in the first line is wrong, we will deduct all the marks) **(6 marks)**

Strategy _____2______ is better.

Reasons:
Because in Strategy 1, CUSTOMERS join BRANCHES is equal to the cross-product of the two relations and the size of the join result will become as large as 20,000 * 4,000 = 80,000,000 tuples. This intermediate result is very large and later when joining this intermediate result with ACCOUNTS, the cost is also large.

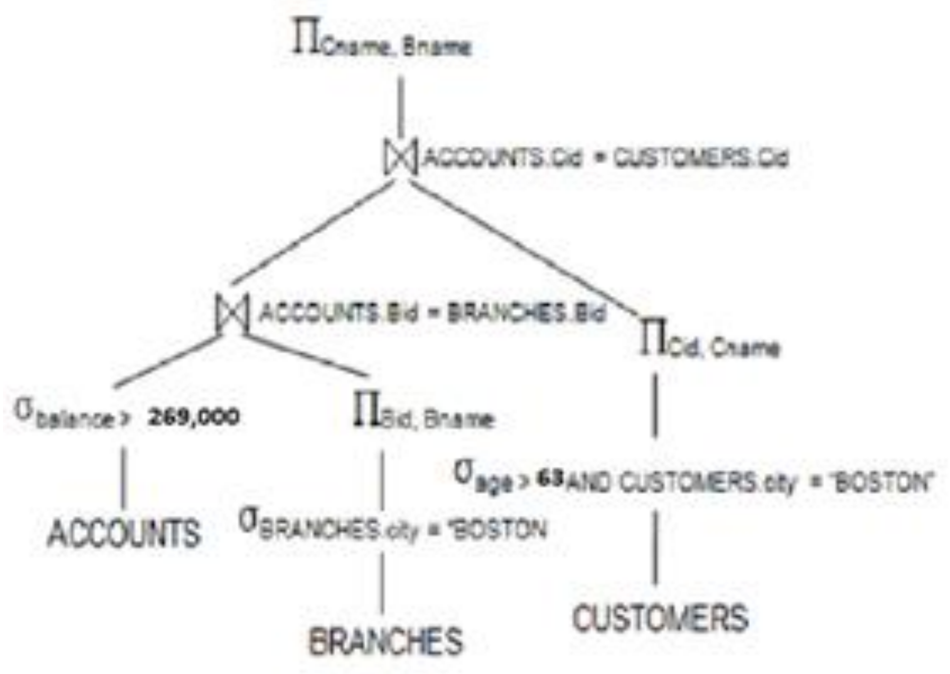
In comparison, in Strategy 2, CUSTOMERS join ACCOUNTS has only 24,000 tuples. And later when joining this intermediate result with BRANCHES, the cost is also small. (Just need to answer the key points).

(c) Consider the following SQL query:

```
SELECT     Cname, Bname
FROM       CUSTOMERS, BRANCHES, ACCOUNTS
WHERE      CUSTOMERS.cid = ACCOUNTS.cid
AND        BRANCHES.bid = ACCOUNTS.bid
AND        age > 63
AND        CUSTOMERS.city = “BOSTON”
AND        BRANCHES.city = “BOSTON”
AND        balance > 269,000
```

Assume that 30% of customers are older than 63, 20% of customers live in BOSTON, 10% of branches are located in BOSTON and 30% of balance are greater than 269,000.

Using the above information and the heuristic rules for optimizing a query that you have learned in the course, **draw the most efficient query tree for the SQL query**. You should clearly indicate all the nodes and branches in the tree. **(6 marks)**



$\Pi_{Cname, Bname}$
$\bowtie_{ACCOUNTS.Cid = CUSTOMERS.Cid}$
$\bowtie_{ACCOUNTS.Bid = BRANCHES.Bid}$
$\Pi_{Cid, Cname}$
$\sigma_{balance > 269,000}$
$\Pi_{Bid, Bname}$
$\sigma_{age > 63 \text{ AND } CUSTOMERS.city = \text{'BOSTON'}}$
ACCOUNTS
$\sigma_{BRANCHES.city = \text{'BOSTON}}$
BRANCHES
CUSTOMERS

For reference only (no mark is counted for the explanations):

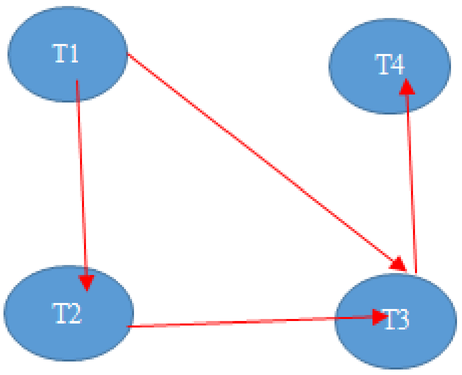
The selectivity of the operation age > 63 on CUSTOMERS is 30%.
The selectivity of the operation CUSTOMERS.city = “BOSTON” on CUSTOMERS is 20%.
The selectivity of the operation BRANCHES.city = “BOSTON” on BRANCHES is 10%.
The selectivity of the operation age > 63 AND city = “BOSTON” on CUSTOMERS is 30%*20% = 6%.
The selectivity of the operation balance > 269,000 on ACCOUNTS is 30%.
The estimated size of $\sigma_{\text{age > 63 AND CUSTOMERS.city = “BOSTON”}}$ CUSTOMERS is 6% * 20, 000 = 1200 tuples.
The estimated size of $\sigma_{\text{BRANCHES.city = “BOSTON”}}$ BRANCHES is 10% * 4,000 = 400 tuples.
The estimated size of $\sigma_{\text{balance > 100,000}}$ ACCOUNTS is 30% * 24,000 = 7,200 tuples.
The estimated size of CUSTOMERS join ACCOUNTS is min(1200 , 24.000) = 1,200 tuples.
The estimated size of BRANCHES join ACCOUNTS is min(7.200 , 400) = 400 tuples.
The estimated size of CUSTOMERS join BRANCHES = 1200 * 400 = 480,000 tuples.
The intermediate result should be as small as possible, so we join BRANCHES and ACCOUNTS first and then join the intermediate result with CUSTOMERS. This is the trickiest part of the question: join which two relations out of the three first.
Note: the left and right hand sides of a join operator can be switched.

Q5 (16 Marks) Transaction Management and Recovery Management

(a) Consider the following schedule S. **(8 marks)**

T1	T2	T3	T4
Write(x)			
Read(x)			
		Write(z)	
	Read(x)		
	Read(y)		
			Read(z)
		Read(x)	
		Write(y)	

(i) Draw a precedence graph for schedule S.



(ii) If S is serializable, write down an equivalent serial schedule. If not, explain why.

Yes, S is serializable and the equivalent serial schedule is T1, T2, T3, T4.

(iii) Consider the schedule S: r1(X); r2(Y); w2(X); w1(Y). Does deadlock occur when using the two-phase locking protocol? Explain why.

Yes, deadlock occurs with such an interleaving of the actions of these transactions. T1 can gain read lock on X and T2 can gain read lock on Y, but T2 cannot gain write lock on X because it has conflict with read lock of transaction T1, so T2 has to wait. Similarly, T1 cannot gain write lock on Y due to the conflict of read lock of the transaction T1. So, T2 cannot proceed as well and they will wait forever.

(b) Transactions T1 and T2 access data objects A, B, C, D, E. Consider the following sequence of log records for deferred database modification scheme: **(8 marks)**

```
<START T1>
<T1, A, 1>
<START T2>
<T2, B, 2>
<T1, C, 3>
<T2, D, 4>
<COMMIT T2>
<T1, E, 5>
<COMMIT T1>
```

(i) Suppose the last record that appears on disk at the time of a crash is <COMMIT T2>. What will the recovery manager do to recover from this crash in terms of updates to the disk and to the log?

It will go backwards from <COMMIT T2>.

Since T1 has not committed, nothing for T1 needs to be redone. Recovery manager will realize that T2 has been committed, so it will redo that by setting D to 4 on disk. It will skip <T1, C, 3> <T1, A, 1> and <START T1>, and then redo <T2, B, 2> by setting B to 2 on disk.

Done.

(ii) After a transaction is rolled back under the timestamps ordering protocol, it is usually assigned a new timestamps when it starts again. Can it keep its old timestamps? Explain briefly the reason.

The re-submitted transaction cannot be assigned to its old timestamps, simply because it will probably be rolled back again. For all its updates it will be too late for others to read.

Q6 (16 marks) SQL and PL/SQL

The following database contains three relational tables.

- Student(stu_id, full_name(Not NULL), major, GPA)
- Book(book_id, title, author, publication_date, storage_quantity)
- Borrow(*stu_id*, time_stamp, *book_id*)

Where

- Keys are underlined and foreign keys are in italics.
- In **Borrow** table foreign key *stu_id* refers to *stu_id* in **Student** table, and *book_id* refers to *book_id* in **Book** table.
- The datatype of *stu_id*, *book_id*, *GPA and storage_quantity* are **number(10)**; the datatype of *publication_date* is **date**, while all the other attributes for all tables are **varchar(100).**
- The publication_year attribute is in the default format of date in Oracle.
- The minimal possible value for *storage_quantity* is 0.
- Different students could have the exact same *full_name*.
- Assume the Oracle environment is identical to the lab environment in this course.

(a) Write the SQL statement to select the second highest *GPA* of all the students in the **Student** table. *You may use **ONLY** these keywords in your query: "SELECT, FROM, WHERE, DISTINCT and COUNT"*. (The output of the query should be a single number which represents the second highest *GPA*. You will get 0 marks if you use any SQL keywords other than the listed 5.) **(3 marks)**

```
Select distinct GPA
from Student s1
where 2 = (Select count(distinct GPA) from Student s2 where s1.GPA<=s2.GPA);
```

(b) Write an equivalent SQL query ***__*without using DISTINCT keyword*__*** (You cannot use DISTINCT keyword in this question.) for the following SQL query. **(3 marks)**

```
SELECT DISTINCT GPA
FROM Student;
```

```
SELECT GPA FROM STUDENT s1
WHERE s1.stu_id = ( SELECT MAX(s2.stu_id)
FROM Student s2
WHERE s1.GPA = s2.GPA)
```

(c) Write a SQL query that returns the full names of all students who have borrowed some books by the author 'Mark Twain' which is published in March the 3rd year 2017 and have never borrowed any book written by author 'Shakespeare'. The result should not contain duplicates but does not need to be ordered. **(4 marks)**

```
select distinct s.full_name
From Student s, Book b, Borrow bo
Where b.author = 'Mark Twain' and b.publication_date = '03-MAR-17' and b.book_id =
bo.book_id and bo.stu_id = s.stu_id and s.stu_id NOT IN
(Select bo2.stu_id
from Borrow bo2, Book b2
Where b2.author = 'Shakespeare' and bo2.book_id = b2.book_id);
```

(d) Write a PL/SQL script to create a trigger named *'No_Book_Left'* that will check if any operation is trying to update the value for *storage_quantity* attribute to a number smaller than 0. Do nothing if the new *storage_quantity* attribute is still greater than 0. If the new *storage_quantity* attribute is smaller than 0, then:

- Throw a user defined EXCEPTION named *'book_out_of_storage'*, and after catching the exception print out *'Operation Failed!:No book left.'* on the SQL/PLUS console.
- Keep the original vaule for *storage_quantity* before the current update operation.

You may presume all the operations to change the *storage_quantity* in table **Book** are like this:

```
UPDATE Book SET storage_quantity = -1 WHERE book_id = 12234;
```

(3 marks)

```
CREATE OR REPLACE TRIGGER No_Book_Left
BEFORE UPDATE ON Book
FOR EACH ROW
DECLARE
    book_out_of_storage EXCEPTION;
BEGIN
    IF (:new.storage_quantity < 0) THEN
        :new.storage_quantity := :old.storage_quantity;
        RAISE book_out_of_storage;
    END IF;
EXCEPTION
    WHEN book_out_of_storage THEN
        DBMS_OUTPUT.PUT_LINE('Operation Failed!:No book left.');
END;
```

(d) Complete the following PL/SQL script to find all the *book_id* of the books with 0 *storage_quantity* in the **Book** table, and insert those *book_id* into a new table PopularBooks(*book_id*), where *book_id* is of number(10) data type, the primary key of the table and a foreign key with reference to *Book.book_id*. (You need to create **PopularBooks** table in the script, you may presume that **PopularBooks** table doesn't exist in the database before you run your script). Please also print all the book_id inserted to **PopularBooks** table once on the SQL/PLUS console. (Hint: Follow the instructions in the comments quoted by /* */.) **(3 marks)**

```
/*Create PopularBooks table*/
CREATE table PopularBooks(
      book_id number(10) references Book(book_id),
      primary key(book_id));

DECLARE
     var_bookid Book.book_id%TYPE;
     CURSOR book_cursor
/*select the book_id with 0 storage_quantity*/
    IS SELECT book_id from Book WHERE storage_quantity = 0;
BEGIN
           OPEN book_cursor;
LOOP
      FETCH book_cursor INTO var_bookid;
      EXIT WHEN
/*complete this statement*/
                                        (var_bookid);
      INSERT INTO PopularBooks VALUES (var_bookid);

END LOOP;
      CLOSE book_cursor;
END;
/
```

- End of the Exam Paper-