中山大学软件学院本科生期末考试

考试科目：《数据库系统原理》（A 卷）

学年学期：2014 学年第 3 学期　　姓　　名：_____

学 院/系：软件学院　　学　　号：_____

考试方式：开卷　　年级专业：_____

考试时长：120 分钟　　班　　别：_____

警示 《中山大学授予学士学位工作细则》第八条："考试作弊者，不授予学士学位。"

----------以下为试题区域，共 7 道大题，总分 100 分,考生请在答题纸上作答----------

1. (10 marks) Let R = {A, B, C, D, E, F, G} and F = {AB→C, A→C, A→E, B→C, EF→EG, G→F}. Answer the following three questions.

1) **(4 marks)** Compute the minimal cover of F.

B→C
A→C
A→E
EF→G
G→F

2) **(4 marks)** Decompose R into 3NF relations.
{B, C}, {A, C, E}, {E, F, G} and {A, B, D, F} (OR {A, B, D, G})

3) **(2 marks)** Is the composition in (b2) in BCNF? Briefly explain your answer.

No. For {E, F, G} and G→F, G is not a candidate key.

2. (10 marks) Assume there is an employee database Employee (*eid*: 8 bytes, *ename*: 16 bytes, *did*: 4 bytes, *email*: 12 bytes), where *eid* and *ename* are respectively the id and name of an employee and *did* is the id of the department in which the employee works. Suppose there are 50,000 employee records and 500 departments (i.e. each department has 100 employees on average). A page size is 1,000 bytes and a pointer costs 4 bytes.

1) **(4 marks)** Assume that the employee file is sorted sequentially on *did* and there is no index. Estimate the page access cost for retrieving the records of all employees working in a department with a given *did*. (You should show your argument and the main steps of the estimation clearly in the answer.)

   Answer
   Record size = 40 bytes, 25 records per page, 2,000 pages.
   Finding the first record requires $\log_2 2000 + 3$ more pages to search the remaining records (each dept has 100 employees which are distributed in 4 pages).

2) **(6 marks)** Assume only 20 pages of main memory are available for running the external sorting of the employee file on *did*.
   - How many PASSes are needed for the external sorting?
   - In each PASS, how many runs are created?
   - What is the total cost of the sorting in terms of **pages**?

   Answer:
   3 PASSes:
   PASS 0: 2000 pages / 20 pages per run = 100 runs
   PASS 1: ceil (100 runs / 19 runs per run) = 6 runs
   PASS 2: 1 run
   Total cost: (2000 pages read per pass + 2000 pages write per pass) * 2 PASSes + 2000 pages read per pass = 10000 pages (Note: Output is not counted!)
   Or: 2000 * (2 * 2 + 1) = 10000 pages transfer.

3. (10 marks) Suppose a bookstore has the following five relational tables:
BOOK (BID, TITLE, *AID*, SUBJECT, QUANTITY_IN_STOCK)

AUTHOR (AID, NAME)
CUSTOMER (CID, NAME)
ORDER_DETAILS (*OID*, *BID*, QUANTITY)
ORDER (OID, *CID*, ORDER_YEAR)

In the above tables, keys are underlined and foreign keys are in *italics*. Each author has authored at least one book in the store. Each book has exactly one author. Each order is made by exactly one customer and has one or more associated record in ORDER_DETAILS (e.g., an order may contain different books).

Express the following query using (i) **SQL** expressions, (ii) the **relational algebra (RA)**.

Find the distinct customer IDs (CID) of customers who have purchased more than 10 identical books in one order at least once.

(i)
```
SELECT DISTINCT CID
 FROM ORDER_DETAILS od, ORDER o
WHERE QUANTITY >= 10 AND od.OID = o.OID
```

(ii) $\pi_{CID}(\sigma_{QUANTITY \geq 10}(ORDER\_DETAILS \bowtie_{OID} ORDER))$

4. (10 marks) A B+ tree with n=5 is shown in Figure 1, in which only search keys are shown and pointers to the file system are hidden. We want to insert a data entry with search key "23".

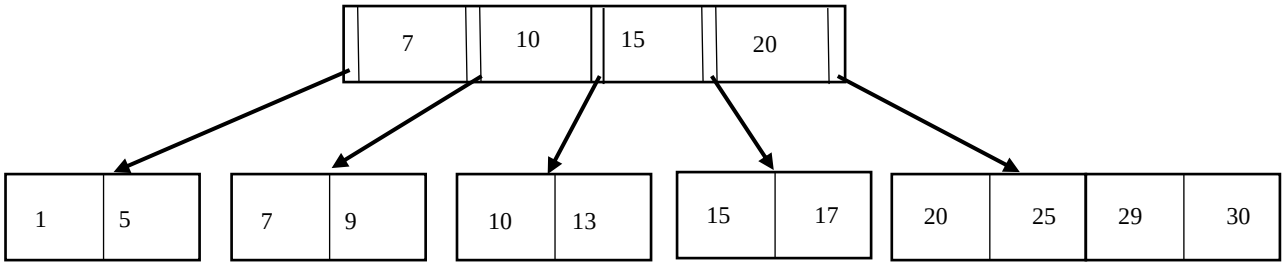


Figure 1. A B+ Tree Structure

1) Which of the following descriptions about the insertion operation is correct?

A. The B+ tree contains 2 levels after insertion. 2 node splits are needed during insertion. The root node contains search key “15”.
B. The B+ tree contains 3 levels after insertion. 1 node split is needed during insertion. The root node contains search key “20”.
C. The B+ tree contains 3 levels after insertion. 2 node splits are needed during insertion. The root node contains search key “15”.
D. The B+ tree contains 3 levels after insertion. 2 node splits are needed during insertion. The root node contains search key “20”.

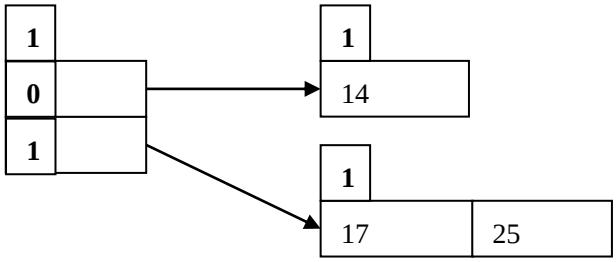
Answer: C

2) We want to delete the data entry with search key “7”. How many leaf nodes store only two data values after deletion?
   A. 2
   B. 3
   C. 4
   D. 5

Answer: **A**

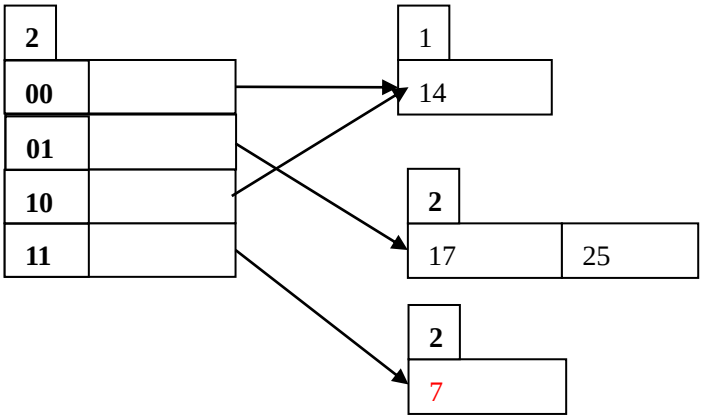
5. (20 marks) You are given an initial hash structure with three keys already inserted as below. The hash function is h(x) = x mod 16. Draw *five* extendable hash structures corresponding for each insertion of the following search key values *K*: 7, 15, 20, 37, 18. Assume each bucket can hold two keys and the search key values arrive in the given order (i.e. 7 being the first coming key and 18 being the last one).

You should follow the convention used by lecture slides: binary hash indices starting from **the least significant bit**. (E.g. 1 is the least significant bit of the 4 digit binary number 0001.)

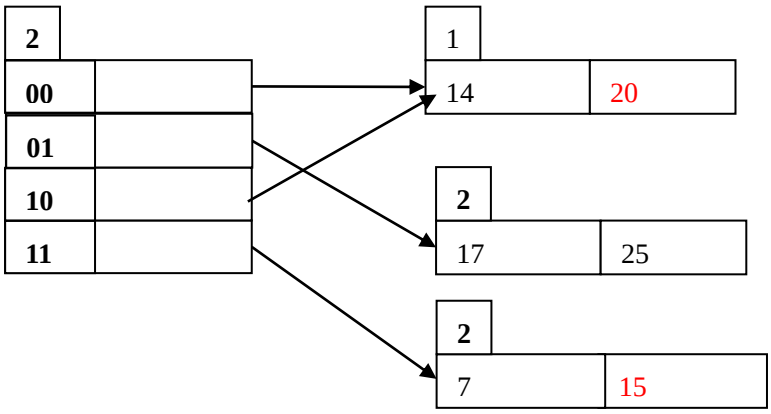


Answer

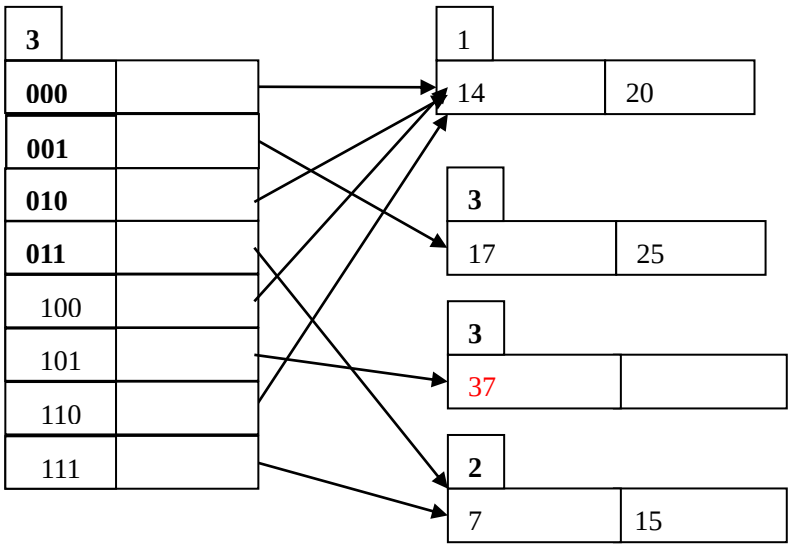
Insert 7



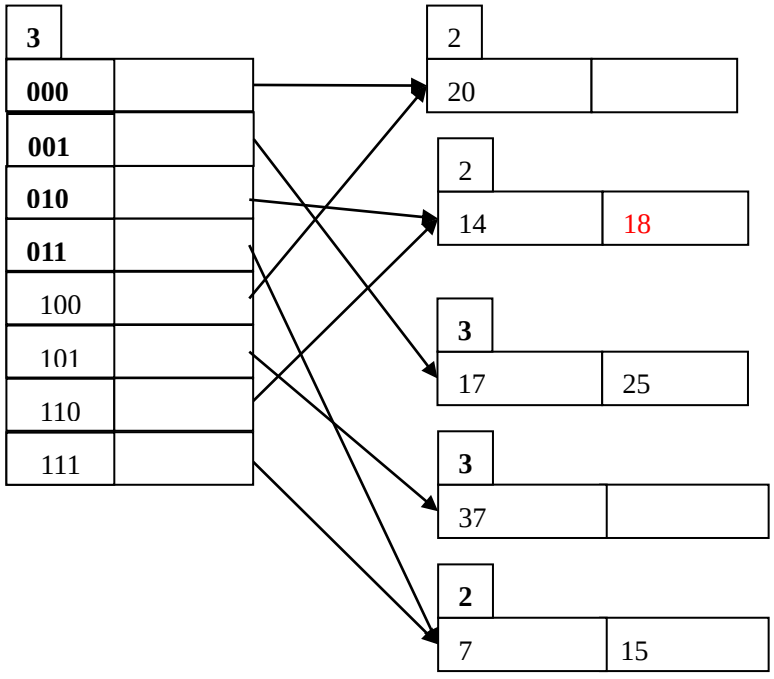
Insert 15 and 20



Insert 37



Insert 18



6. (25 Marks) Consider a database consisting of the following three relation schemas:

SAILORS (sid, sname, rating, age)

BOATS (bid, bname, color)

RESERVES (*sid*, *bid*, date, rname)

The meaning of the attributes in the above schemas is self-explanatory. For example, *sid* is the sailor identity number and *bname* is the name of the boat. The primary keys of the relations are underlined. The attribute *sid* in RESERVES is a foreign key referencing SAILORS. The attribute *bid* in RESERVES is a foreign key referencing BOATS.

The relation SAILORS has 100,000 tuples and 100 tuples of SAILORS fit into one page. The relation BOATS has 50,000 tuples and 25 tuples of BOATS fit into one page. The relation RESERVES has 10,000 tuples and 20 tuples of RESERVES fit on one page. We assume all attribute values and pointers in these three relations, if needed to be considered, are of the same size.

(a) **(10 marks)** Assume that we use Indexed Nested Loop Join to compute SAILORS ⋈ RESERVES using SAILORS as the outer relation. RESERVES have a primary B+-tree index with 2 levels on the join attribute. Estimate the cost of the join in terms of **pages**.

Number of SAILORS **pages**: br = 100,000/100 = 1,000

Number of SAILORS **Tuples**: nr = 100,000.

The cost is br+ c * nr = 1,000 + (2+1) * 100,000 = 301,000.

(b) **(5 mark)** Assume that 26% of the sailors have the rating bigger than 5. Estimate the result size of $\pi_{sid}(\sigma_{rating>5}$ SAILORS) in terms of **pages**.

Size = 26% * 100,000 / 100/4 = 65 pages

260 divided by 4, for there is projection and all the attributes have same size

(c) **(10 marks)** Consider the following two strategies to compute the join operation SAILORS ⋈ BOATS ⋈ RESERVES.

Strategy 1: (SAILORS⋈ BOATS) ⋈ RESERVES

Strategy 2: (SAILORS⋈ RESERVES) ⋈ BOATS

Which strategy is better? Explain the reason(s) of your choice based on the size of the intermediate result using the above strategies.

Strategy 2 is better.

Because in Strategy 1, SAILORS ⋈ BOATS is equal to the cross-product of the two relations and the size of the join result will become as large as 100,000 * 50,000 = 5,000,000,000 tuples. This intermediate result is very large and later when joining this intermediate result with RESERVES, the cost is also large.

In comparison, in Strategy 2, SAILORS ⋈ RESERVES has only 10,000 tuples. And later when joining this intermediate result with BOATS, the cost is also small.

7. (15 Marks) Consider a schedule S which consists of four transactions as follows:

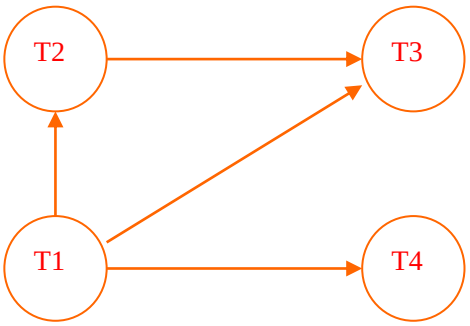
S = <T3_R(U), T2_R(X), T2_W(X), T3_R(X), T1_R(Y), T1_W(Y), T3_W(X), T1_R(Z), T4_R(Z), T4_W(Z), T2_W(Y), T3_R(Y)>
The notation is self-explanatory. For example, T1_R(X) means that transaction T1 reads item X.

(a) **(5 marks)** Fill in the following table representing S with the usual notations in lecture slides. The first operation R(U) has been shown in the table. Show clearly all *conflicting pairs* with downward arrows on the operations.

T1	T2	T3	T4
		R(U)	
	R(X)		
	W(X)		
		R(X)	
R(Y)			
W(Y)			
		W(X)	
R(Z)			
			R(Z)
			W(Z)
	W(Y)		
		R(Y)	

(b) **(5 marks)** Construct the precedence graph of S. Explain why or why not the schedule is conflict-serializable.

Precedence Graph of S:



No cycle.

(c) **(5 marks)** Suppose the format of the "commit" operation is Ci where i = 1, 2, 3, or 4. For example, the operation C1 means that the transaction T1 commits. Append all the commit statements to S so that the schedule becomes **recoverable**. For example, one append can be SC4C3C2C1 which means running S and then C4, C3, C2, C1. (Note that you should NOT change the sequence of the operations in S other than appending S with the four commit statements to make the schedule recoverable.)

Recoverable (but not cascadeless) schedule: SC1C2C3C4 or (SC1C2C4C3) or (SC1C4C2C3) (Note: any permutation of Ci satisfies the commit order constraints: C1→C2, C1→C3, C1→C4, C2→C3 is correct)