UNIVERSITY of TORONTO
SCARBOROUGH

CSCD43 Exam Solution
Database Systems Technology

Duration – 3 Hours

Examination Aids: No Aids Allowed

Student #: |_|_|_|_|_|_|_|_|_|

Last Name:_____ **First Name:**_____

Do **not** turn this page until you have received the signal to start. In the meantime, please fill out the identification section above, and read the instructions below carefully.

This term test consists of 6 questions on 12 pages (including this one), printed on one side of the paper. When you receive the signal to start, please make sure that your copy of the examination is complete.

Answer each question directly on the examination paper, in the space provided, and use the reverse side of the pages for rough work. If you need more space for one of your solutions, use the reverse side of the page and indicate clearly the part of your work that should marked.

#1: Short Questions	**_____/12**
#2: E/R Diagram	**_____/12**
#3: Transactions	**_____/10**
#4: Indexing & Hashing	**_____/25**
#4: Recovery	**_____/16**
#6: Concurrency & Deadlocks	**_____/25**
	***Total*:_____/100**

Question 1: Short Questions [*12 marks total*]

True	False	Chocolate is the flavor in a chocolate cake **(Note: worth 2 marks)**
True	False	Every relationship in an E-R diagram must translate to an individual relation in the relational model.
True	False	Secondary indexes are always dense.
True	False	When a sparse index is used, each record must have an entry in the index.
True	False	In undo logging, it is not always possible to recover some consistent state of a database system if the system crashes during recovery.
True	False	Two-phase locking prevents deadlocks.
True	False	ACR (avoids cascading rollback) schedules are always serializable.
True	False	Recoverable schedules sometimes require cascading rollback.
True	False	Hash index is efficient in answering range queries, such as finding products with price higher than 100".
True	False	Both the wait-die and wound-wait schemes guarantee that every transaction eventually completes (i.e. no starvation).
True	False	For any data file, it is possible to construct two separate dense first level indexes on different keys.

Question 2: E/R Diagram [*12 marks total*]

Your task is to design a database for an online video service that offers hit TV series. The following is the description of the application:

• Each series has a name and a description.

• We would like to record information on the regular cast of the series. In other words, for each main actor/actress in the series, we want to record his/her name and address. You may assume that the cast doesn't change.

• Each series has many episodes. Each episode has an episode number, the year and date it was first aired, and the length of the episode in minutes. The episode number uniquely identifies an episode with respect to the series, but two different series can have the same episode number.

• Registered viewers can comment on any episodes. For each comment, we want to record its post date and content.

• For each registered viewer, we want to record his/her userID and password. If

– The name of each series is unique, so name is a key for Series.

– The name of actors/actresses is unique, so name is a key for Cast.

– userID is unique, so userID is a key for Users.

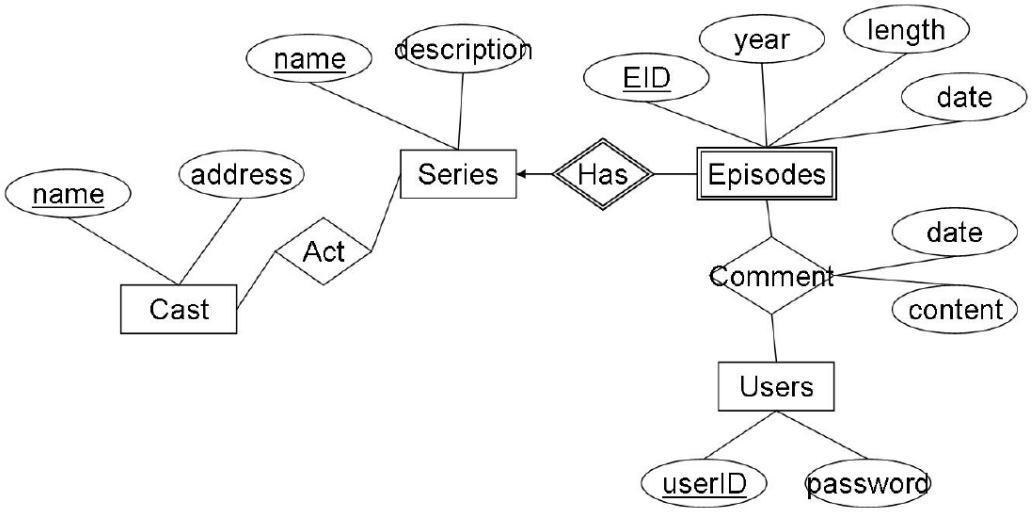
– Many actors/actresses can star in one series. One actor/actress can star in multiple different series. Hence, Act is many-many.

– Many users can comment on one episode. One user can comment on different episodes. Hence, Comment is many-many.

State assumptions in your solution clearly.

(a) [*6 marks*] Draw an ER diagram for this application. Be sure to mark the multiplicity of each relationship of the diagram. Decide the key attributes and identify them on the diagram.

Solution:
There can be multiple solutions for this problem, depending on students' assumptions. If the assumptions are reasonable and the ER diagram is drawn correctly based on the assumptions, we will give full credit.



Possible solutions: Students may have Comment as an entity instead of a relation. Also, students may have one entity for actors and one for actresses and these entities are ISArelated to Cast.
Note: If your assumptions are reasonable, the ER diagram corresponds to the assumptions, and the ER diagram is correctly drawn, we will accept the solution.

(b) [*6 marks*] Convert the above ER diagram into a relational schema. Merge relations where appropriate. Your solution should have as few relations as possible, but they do not need to be normalized. Specify the key of each relation in your schema.

Solution:
We will grade 2b) according to student's solution in 2a). If the translation is correct, we will give full credit.
– Cast(name, address)
– Act(castName, seriesName)
– Series(name, description)
– Episodes(EID, seriesName, year, date, length)
– Comment(userID, EID, seriesName, date, content)
– Users(userID, password)

Question 3: Transactions [*10 marks total*]

What are the differences between the four levels of transaction isolation?

Solution:

• Read uncommitted: Transactions do not need to acquire any locks before reading data. Transactions may thus read data written by other transactions that have not yet committed. The value read may thus later be changed further or rolled-back. This problem is called the dirty read problem. This level of isolation also suffers from all the problems of the more restrictive isolation levels below.

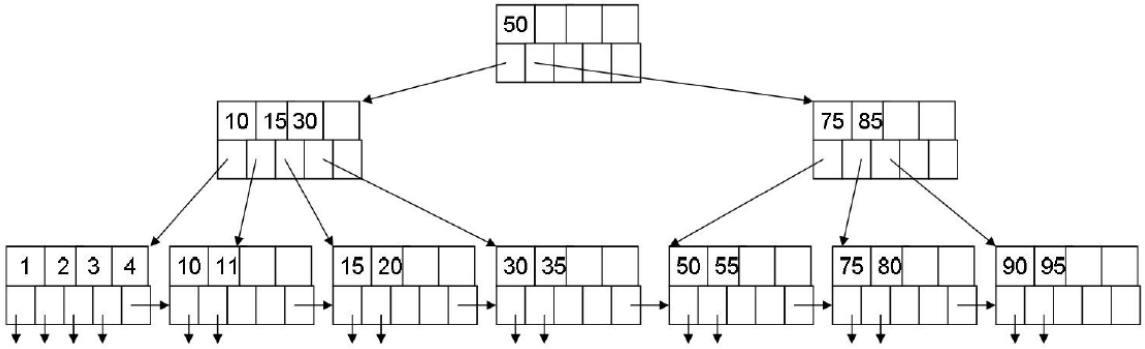
• Read committed: Transactions must acquire shared locks before reading data. They may release these locks as soon as they read the data (short duration read locks). This level of isolation guarantees that the transaction never reads uncommitted data by other transactions. However, it doesn't ensure the data will not change until the end of the transaction. If a transaction reads the same data item twice, it can see two different values. This problem is called the non-repeatable read problem. This level of isolation also suffers from all the problems of the more restrictive isolation levels below.

• Repeatable read: Transactions must acquire long duration read locks on the individual data items that they read. This level of isolation provides all the guarantees of the read committed level. It also ensures that data seen by a transaction does not change until the end of the transaction: i.e., it provides repeatable reads. However, because locks are held on individual data items, transactions may experience the phantom problem. If a transaction reads twice a set of tuples that satisfy a predicate, it only locks the individual data items that match the predicate. If another transaction inserts a tuple that matches the predicate between the two read operations, that new tuple will appear as a result of the second read.

• Serializable: Transactions must acquire long duration read locks on predicates as well as on individual data items. This level of isolation protects against all the problems of the less restrictive levels. It ensures serializability.

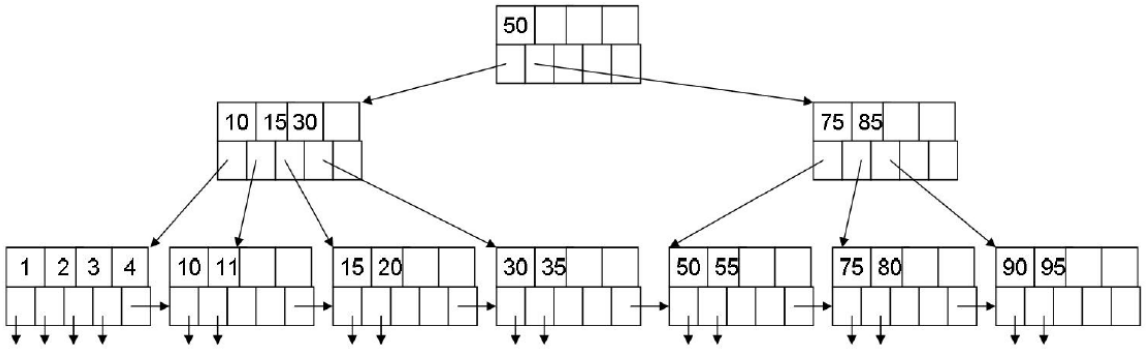
Question 4: Indexing & Hashing [*25 marks total*]

(a) [*15 marks*] Consider the following B+ tree of order 4 (i.e., n = 4, each index node can hold n keys and n + 1 pointers)



(i) [*5 marks*] Based on the tree in above figure, show the resulting tree after inserting key 5.

Note:
The above diagram needs correction, the right side node in the second level has 75 and 90 and not 75 and 85. Students were informed of this during the exam.



(ii) [*5 marks*] Based on the tree in above figure, show the resulting tree after deleting key 90.

Note:
The above diagram needs correction, the right side node in the second level has 75 and 90 and not 75 and 85. Students were informed of this during the exam.

(iii) [*5 marks*] Based on the tree in above figure, show the steps in executing the following operation: Lookup all records in the range 40 to 100 (including 40 and 100).

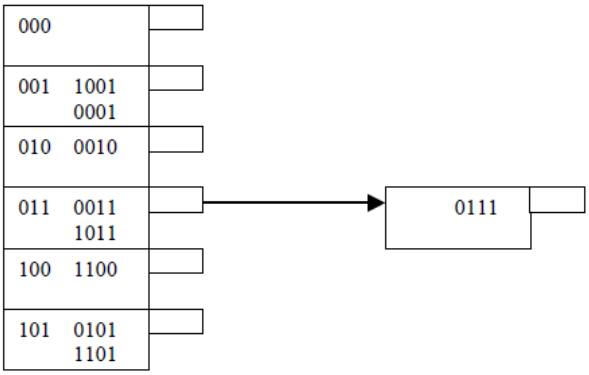
(b) [*10 marks*] Consider a linear hashing index with buckets that can hold up to X records (or key/ptr. pairs). Say that *m*, the id of the last active bucket, is increased whenever the "utilization" exceeds 80%. In other words, if N is the total number of records stored in the index, then *m* is increased whenever N/(X*(*m*+1)) > 0.8.

Illustrate the final state of a linear hashing index with X = 2 when records with the following hash values are inserted, in order:

0011, 1100, 0101, 1011, 0010, 1101, 0111, 1001, 0001

Assume that the index starts out empty with one bucket allocated and with *m* = 0.

Solution:



Question 5: Recovery [*16 marks total*]

Consider a database with data items {A, B, C, D}. The system uses an undo/redo scheme and has the following logs. Note that an entry <T, X, old, new> means transaction T changes the value of X from old to new. We consider recovery using this undo/redo log.

1. <T2 start>
2. <T2, B, 10, 11>
3. <T1 start>
4. <T2 commit>
5. <T1, A, 20, 21>
6. <Checkpoint start; Active= {T1}>
7. <T3 start>
8. <T3, C, 30, 31>
9. <T4 start>
10. <T4, B, 40, 41>
11. <T4 commit>
12. <Checkpoint end>
13. <T3, D, 50, 51>
14. <Checkpoint start; Active= {T1, T3}>
15. <T1, C, 31, 32>
16. <T5 start>
17. <T5, D, 51, 52>
18. <T3 commit>
19. <T6 start>
20. <T6, C, 32, 33>
21. <T5 commit>
22. System failed

Note:
Step 10 needs correction. It should be <T4, B, 11, 41>**. Students were informed of this during the exam.**

(a) [*4 marks*] List all possible values of A, B, C and D. That is, what are the possible data values on the disk at the point of failure (after action 21)?

Solution: A= 21, B= 11 or 41, C = 30 or 31 or 32 or 33, D = 50, 51, 52

(b) [*4 marks*] During recovery, what are the transactions that need to be undone?

Solution: T1, T6.

(c) [*4 marks*] During recovery, what are the transactions that need to be redone?

Solution: T3, T4, T5

(d) [*4 marks*] What are the values of A, B, C, D after recovery? Explain why?

Solution: A = 20 B = 41 C = 31 D = 52
A= 20 because T1 is redone.
B= 41 because T4 is redone.
C= 31 because both T1 and T6 are undone and T3 is redone.
D = 52 because T5 is redone.

Question 6: Concurrency & Deadlocks [*25 marks total*]

(a) [*5 marks*] Briefly explain why it is not necessarily desirable to execute multiple transactions as a serial schedule in a database system.

Solution:
We can improve the throughput of a system by performing some tasks at CPU while doing I/O activities. Also, we can reduce average waiting time if we can execute a short transaction while executing a long transaction.

(a) [*5 marks*] When we use timestamps as a concurrency method, the scheduler of a database system assigns each transaction T a unique timestamp $T_S(T)$. The scheduler also associates each database element X with the read time of X, $R_T(X)$, and the write time of X, $W_T(X)$. The scheduler aborts a transaction T reading a database element X if $T_S(T) < W_T(X)$. Explain why.

Solution:
If $T_S(T) < W_T(X)$, it means that a newer transaction T′ has already written some value on X. When we use a concurrency method with timestamps, we consider a serial schedule where all the actions of each transaction are considered to be done at the starting time of that transaction. Therefore, we must swap T's read action on X with T′'s write action on the same element X. However, this is a conflicting swap, and thus we cannot convert the actual schedule to the serial schedule.

(c) [*5 marks*] Consider the schedule

S: w1(X); w3(X); w2(X); w4(X); r4(Y); w1(Y).

Draw the precedence graph for schedule S. Is it conflict-serializable? Explain why.

Solution:
It is not conflict-serializable since the graph has a cycle.

(d) [*5 marks*] Consider the schedule

S: r1(X); r2(Y); w2(X); w1(Y).

Does deadlock occur using the two-phase locking rule? Explain why.

Solution:

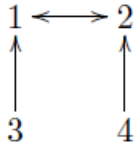
Yes, deadlock occurs with such an interleaving of the actions of these transactions. T1 can gain read lock on X and T2 can gain read lock on Y, but T2 cannot gain write lock on X because it has conflict with read lock of transaction T1, so T2 has to wait. Similarly, T1 cannot gain write lock on Y due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

(e) [*5 marks*] For the sequence of actions below, assume that locks are requested immediately before each read and write action. However, if a transaction is already holding a lock on a database element, it does not need to obtain that lock again to execute another action on that element. Also, unlocks occur immediately after the final action that a transaction executes. Tell which locking actions are denied, and whether a deadlock occurs, by drawing a wait-for graph.

r1(A); r2(B);w1(C);w2(D); r3(C);w1(B);w4(D);w2(A);

Solution:

l3(C) denied (C locked by 1)
l1(B) denied (B locked by 2)
l4(D) denied (D locked by 2)
l2(A) denied (A locked by 1)
wait-for graph is:



Yes, a deadlock occurs, because of a cycle between 1 and 2 in the wait-for graph.

END OF EXAM