CO251 Database Systems Assignment 2

Drishika Nadella
181ME222

June 3, 2020

Question 1

Give a set of FDs for the relation schema R(A,B,C,D) with primary key AB under which R is in 1NF but not in 2NF.
Let us take the set

$$FD : AB \rightarrow CD$$

and

$$B \rightarrow C$$

The first normal form requires each cell to contain a single value and each record to be unique. The second normal form requires each record to be 1NF and be a primary key.
Since

$$AB \rightarrow CD$$

this implies that

$$AB \rightarrow ABCD.$$

Therefore, AB is a key. But since it is the smallest subset of keys, AB is also considered the primary key.
Considering the relation

$$FD : B \rightarrow C$$

This is not a 2NF because:
B is not a super key.
It is a non-trivial functional dependency i.e.

$$B \notin C$$

C is not part of some key for R.
B is a proper subset of the key AB (this is based on transitive dependency, where changing one non-key value in a column can change the non-key value in another column).

Question 2

Give a set of FDs for the relation schema R(A,B,C,D) with primary key AB under which R is in 2NF but not in 3NF.
Let us take the set

$$FD : AB \rightarrow CD$$

and

$$C \rightarrow D$$

The second normal form requires each record to be in 1NF and be a primary key. The third normal form requires each record to be 2NF and transitive functional dependency of non-prime attribute on any super key should be removed.
Since

$$AB \rightarrow CD$$

this implies that

$$AB \rightarrow ABCD.$$

Therefore, AB is a key. But since it is the smallest subset of keys, AB is also considered the primary key.
Considering the relation

$$FD : C \rightarrow D$$

This is not a 2NF because:
C is not a super key
It is a non-trivial functional dependency i.e.

$$B \notin C$$

D is not part of some key for R.

Question 3

R(A,B,C), which has the FD:B $->$ C. If A is a candidate key for R, is it possible for R to be in BCNF? If so, under what conditions? If not, explain why not.
We know that, for a table to be in Boyce-Codd normal form, it should satisfy two conditions. First, the table must be in 3NF. Second, for a dependency

$$A \rightarrow B$$

A should be a super key.
Therefore, for R to be in BCNF, B must be a super key along with A.

Question 4

Let there be a relation R with attributes ABCDE. Let you are given the following dependencies: A − > B, BC − > E, and ED − > A.
a) List all keys for R.
b) Is R in 3NF?
c) Is R in BCNF?
a) The keys are ACD, ECD, BCD.
b) For a table to be in 3NF, each record is to be 2NF and transitive functional dependency of non-prime attribute on any super key should be removed.
In the dependencies

$$A \to B$$

$$BC \to E$$

and

$$ED \to A$$

B, E and A are all prime attributes i.e. they are all parts of keys. Therefore, R is in 3NF.
c) For a table to be in BCNF, the table must be in 3NF and for a dependency

$$A \to B$$

A must be a super key.
In the dependencies given in the question, neither A, BC nor ED are superkeys. Therefore, R is not in BCNF.

Question 5

Consider the following collection of relations and dependencies. Assume that each relation is obtained through decomposition from a relation with attributes ABCDEFGHI and that all the known dependencies over relation ABCDEFGHI are listed for each question. (The questions are independent of each other, obviously, since the given dependencies over ABCDEFGHI are different.) For each (sub) relation:
(a) State the strongest normal form that the relation is in.
(b) If it is not in BCNF, decompose it into a collection of BCNF relations.
a. R1(A,C,B,D,E), A − > B, C − >D
b. R2(A,B,F), AC − > E, B − > F
c. R3(A,D), D − > G, G − > H
d. R4(D,C,H,G), A − > I, I − > A
e. R5(A,I,C,E)
a) The strongest normal form: 1NF. BCNF Decomposition: CD, AB, ACE.
b) The strongest normal form: 1NF. BCNF Decomposition: AB, BF.
c) The strongest normal form: BCNF.
d) The strongest normal form: BCNF.
e) The strongest normal form: BCNF.

Question 6

Suppose that there is a database system that never fails. Is a recovery manager required for this system?
Recovery manager is used to back up, restore and recover database files. But apart from executing backup operations, recovery manager is also used to rollback aborted transactions. Aborted transactions are those transactions in which all the instructions present in the transactions are not executed properly. In this case, these transactions can be "rolled back" to the state it was before the transaction begins.

Question 7

Database-system implementers have paid much more attention to the ACID properties than have file-system implementers. Why might this be the case?
The ACID Properties in DBMS are:

- Atomicity: An operation or a series of operations cannot occur partially. Either all operations occur or none occur.
- Consistency: The database must be consistent before and after the transaction.
- Isolation: Multiple transactions occur independently without interference.
- Durability: The changes of a successful transaction occurs even if the system failure occurs.

These ACID properties are not present in file-systems. In a real world problem, these properties ensure that there is smooth functioning of the DBMS system, where a large amount of data is present.
However, in file systems, the amount of data is not as large, so the tradeoff between the advantages of the ACID properties and the amount of money required to implement them is not enough, so the ACID properties are not implemented.

Question 8

Since every conflict-serializable schedule is view serializable, why do we emphasize conflict serializability rather than view serializability?
A schedule is called conflict serializable if it can be transformed into a serial schedule by swapping non-conflicting operations.
A schedule is called view serializable if it is view equal to a serial schedule (no overlapping transactions).
Both of these schedules are used for concurrency control, where it is ensured that DBMS operations do not conflict with each other.
Conflict serializability is advantageous over view seralizability because it is less restricted and expensive. Moreover, standard concurrency control protocols are based on conflict serializability.

Question 9

Consider a database for a bank where the database system uses snapshot isolation. Describe a particular scenario in which a non-serializable execution occurs that would present a problem for the bank.
Consider the situation where a bank imposes an integrity constraint that the amount present in the checking and savings accounts cannot be negative, to protect data integrity. Suppose that in a bank there is Rs. 500 in the checking account and Rs. 1000 in the savings account. Let the following set of transactions take place:

- Withdraw Rs. 1000 from the checking account (note that there is only Rs. 500). This works because the transaction checks for the integrity constraint by verifying both accounts' balances.
- A concurrent transaction withdraws Rs. 1000 from the checking account after checking for the integrity constraint by verifying both accounts' balances.

Both transactions check for the integrity constraint individually on their own snapshot and come to the conclusion that the balance in the accounts is Rs. 500 and hence, it does not violate the integrity constraint. Due to this snapshot isolation, no error is shown when updated as both transactions update different data items. But this method is clearly not serializable because in reality, the data integrity constraint is violated and this can lead to serious problems.

Question 10

Why do database systems support concurrent execution of transactions, in spite of the extra programming effort needed to ensure that concurrent execution does not cause any problems?
Concurrent execution of transactions is beneficial for multiple reasons:

- Improved Throughput: Having multiple transactions being executed at once allows improved throughput i.e. the number of transactions executed in a given amount of time.
- Convenient and Practical: In a big database where many end-users are involved, it is more practical to have transactions executed simultaneously to reduce delays. This is more convenient for the end-users.
- Better Resource Utilization: A transaction typically involves getting an input, processing the transaction in the CPU, and then displaying the output. So, executing several transactions parallelly makes better use of these resources. For example, while one transaction is taking input, another transaction can be processing while a third can be displaying output.
- Reduced Waiting Time: If the transactions are executed serially, some short transactions that hardly take time may be stuck between long transactions. This increases

the waiting time.But concurrent transactions solve this problem, by improving the response time as multiple transactions are processed at the same time.

Therefore, despite the extra programming effort, ultimately, concurrent execution is better.

Question 11

Consider the following two transactions:
T_{34} **: read(A);**
read(B);
if A = 0 then B := B + 1;
write(B).
T_{35} **: read(B);**
read(A);
if B = 0 then A := A + 1;
write(A).
Add lock and unlock instructions to the above transactions , so that they observe a two-phase locking protocol. Can the execution of these transactions result in deadlock?
Adding locks to T_{34}:
T_{34}: lock-S(A) (Shared lock)
read(A)
lock-X(B) (Exclusive lock)
read(B)
if A = 0
then B:= B + 1
write(B)
unlock(A)
unlock(B)

Adding locks to T_{35}: T_{35}: lock-S(B) (Shared lock)
read(B)
lock-X(A) (Exclusive lock)
read(A)
if B = 0
then A := A + 1
write(A)
unlock(B)
unlock(A)

Question 12

In timestamp ordering,W-timestamp(Q) denotes the largest timestamp of any transaction that executed write(Q) successfully. Suppose that, instead, we defined it to be the timestamp of the most recent transaction to execute write(Q)

successfully.Would this change in wording make any difference? Explain your answer.
The wording in this makes no difference. This is because the most recent transaction is the one which has the largest timestamp. So, essentially, both definitions of W-timestamp(Q) mean the same.

Question 13

Explain why the following technique for transaction execution may provide better performance than just using strict two-phase locking: First execute the transaction without acquiring any locks and without performing any writes to the database as in the validation-based techniques, but unlike the validation techniques do not perform either validation or writes on the database. Instead, rerun the transaction using strict two-phase locking. (Hint: Consider waits for disk I/O.)
Since a transaction waits on disk I/O as well as lock acquisition, the proposed transaction involves the following two phases:
1. The transaction is executed without using locks and writing anything to the database. The data blocks are read from the disk or memory. Therefore, it accounts for most of the disk I/O waiting time.
2. The transaction is re-executed with strict two-phase locking. In this phase, almost all of the waiting time is on obtaining locks. Only in certain rare situations is there some amount of disk I/O waiting time if a disk block required for a transaction is saved to memory before this phase begins.
Such a transaction entails two significant advantages:

- Better concurrency: In the second phase of the transaction, where the locks are acquired, there is little to no waiting time on disk I/O since most of that occurs in the first phase of the transaction. Therefore, transaction time is saved and leads to improved concurrency.
- Increases disk throughput: In certain situations, a disk may be idle while the transaction is acquiring a lock. However, this wastage is not present in the proposed method of transaction, increasing disk throughput.

Such a transaction works better if it is not computationally intensive, because otherwise, there is wasted work.

Question 14

Consider the timestamp-ordering protocol, and two transactions, one that writes two data items p and q, and another that reads the same two data items. Give a schedule whereby the timestamp test for a write operation fails and causes the first transaction to be restarted, in turn causing a cascading abort of the other transaction. Show how this could result in starvation of both transactions.

(Such a situation, where two or more processes carry out actions, but are unable to complete their task because of interaction with the other processes, is called a livelock.)
Consider the transaction table:

T_1	T_2
write(p)	
	read(p)
	read(q)
write(q)	

Let the timestamp of T_2 be greater than that of T_1. Let the timestamp test for every operation apart from write(q) be successful.
When transaction T_1 does the timestamp test for write(q), then TS(T1) < R-timestamp(q) since TS(T1) < TS(T2) and R-timestamp(q) = TS(T2).
Therefore, the write operation fails. T_1 rolls back. This results in transaction T_2 also being rolled back due to cascading as it requires the value of p that is written by transaction T_1.
It is possible for a livelock in such a situation i.e. a transaction has to wait for a indefinite period of time to acquire a lock if this repeats every time the transactions are restarted.

Question 15

Consider the following locking protocol: All items are numbered, and once an item is unlocked, only higher-numbered items may be locked. Locks may be released at any time. Only X-locks are used. Show by an example that this protocol does not guarantee serializability.
Assume there are two transactions T_1 and T_2, and two data items A and B. Let us assume B is higher numbered than A.
Consider the following schedule.

- Lock A in T_1: l_1(A)
- Write in A in T_1: w_1(A)
- Unlock A in T_1: u_1(A)
- Lock A in T_2: l_2(A)
- Read A in T_2: r_2(A)
- Lock B in T_2: l_2(B)
- Write in B in T_2: w_2(B)
- Unlock B in T_2: u_2(B)
- Lock B in T_1: l_1(B)

- Read B in T_1: r_1(B)
- Unlock B in T_1: u_1(B)

T_1 writes in A before T_2 reads A. Also, T_2 writes in B before T_1 reads in B.
This results in a cycle in the schedule. Therefore, the protocol does not guarantee serializability.

Question 16

Explain the purpose of the checkpoint mechanism. How often should checkpoints be performed? How does the frequency of checkpoints affect:
A. System performance when no failure occurs?
B. The time it takes to recover from a system crash?
C. The time it takes to recover from a media (disk) failure?
Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk. Checkpoint declares a point before which the DBMS was in consistent state, and all the transactions were committed. It is used to reduce recovery time after a crash. Using a checkpoint mechanism, redoing the transactions after a crash is not necessary.
The more often checkpoints are performed, lesser the number of updates that need to be performed during the recovery process. In other words, performing more checkpoints improves the recovery system.

A. If no failure occurs, then performing the checkpoint mechanism becomes unnecessary and hence the cost of performing it needs to be taken under consideration. Therefore, in a situation where no failure occurs, lesser number of checkpoints are ideal.

B. The more the frequency of checkpoints, the faster it takes for the system to recover from a crash. This is because when a checkpoint is created, redoing the transactions after a crash is not necessary. Therefore, in a situation where the time to recover after a system crash needs to be shortened, the frequency of checkpoints need to be increased.

C. The time taken to recover from a disk crash is not affected by the frequency of checkpoints.

Question 17

Explain how the database may become inconsistent if some log records pertaining to a block are not output to stable storage before the block is output to disk.

- Let us consider a transaction which modifies the data items of two records, A and B.
- This system crashes after this transaction commits, but prior to the log records being output to stable storage.

- The time of crash was such that the modification to record A had been sent to the disk, but the buffer page containing the record B modification had not been written to the disk yet.
- Later when the system comes back it is in a inconsistent state as there are no log records related to the transactions update on record B in the stable storage making recovery impossible.

In such a situation, when some log records pertaining to a block are not output to stable storage before the block is output to disk, data may become inconsistent.