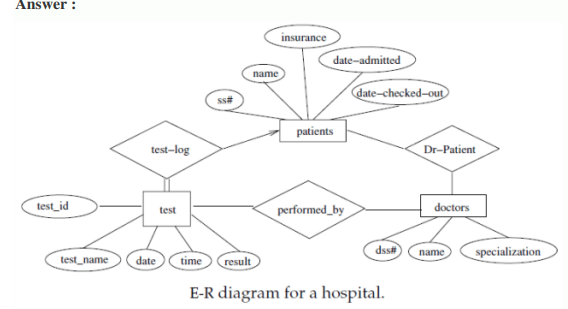
**IT252 Database Systems End Sem Exam Quetions and Answers.**

1) Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted. 6M



2) Write SQL query for the following database schema:

Employee(employee\_name, street, city)

Works(employee\_name, company\_name, salary)

Company(company\_name, city)

Manager(employee\_name, Manager\_name)

a. Find the names, street address, and cities of residence for all employees who work for 'First

Bank Corporation' and earn more than $10,000.

b. Find the names of all employees in the database who do not work for 'First Bank

Corporation'. Assume that all people work for exactly one company.

c. Find the names of all employees in the database who earn more than every employee of

'Small Bank Corporation'. Assume that all people work for at most one company.

d. Find the name of the company that has the smallest payroll.

e. Find the names of all employees in the database who live in the same cities and on the

same streets as do their managers. 10M

a. Find the names, street address, and cities of residence for all employees who work for 'First

Bank Corporation' and earn more than $10,000.

select employee.employee-name, employee.street, employee.city from employee, works

where employee.employee-name=works.employee-name and company-name =

'First Bank Corporation' and salary > 10000)

b. Find the names of all employees in the database who do not work for 'First Bank

Corporation'. Assume that all people work for exactly one company.

Select employee-name from works where company-name not in( 'First Bank Corporation');

c. Find the names of all employees in the database who earn more than every employee of

'Small Bank Corporation'. Assume that all people work for at most one company.

select employee-name from works where salary > all (select salary from works where

company-name = 'Small Bank Corporation')

d. Find the name of the company that has the smallest payroll.

Select C.comany\_name, min(salary) as “smallest payroll”

from Employee E, Works W, Company C

where E.employee\_name = W.employee\_name and

W.company\_name=C.company\_name

Group by C.company\_name ;

e. Find the names of all employees in the database who live in the same cities and on the

same streets as do their managers.

select p.employee-name from employee p, employee r, manages m

where p.employeename = m.employee-name and m.manager-name =

r.employee-name and p.street = r.street and p.city = r.city

3) For each of the schedules below, indicate whether they are conflict serializable. If they are,

then give the equivalent serial order of the transactions.

(i) Schedule S1: R1(A), R1(B), W1(A), R2(B), W2(D), R3(C), R3(B), R3(D), W2(B),

W1(C), W3(D)

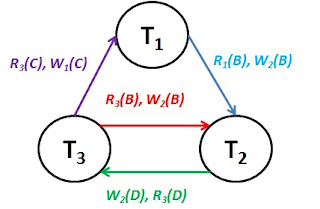
(ii) Schedule S2: R1(A), R1(B), W1(A), R2(B), W2(A), R3(C), R3(B), R3(D), W2(B),

W1(C), W3(D) 10M

**Solution:**

**(i) Schedule S1:**

Schedule S1 is not conflict serializable because the precedence graph for S1 forms a cycle (refer the image below).



**You could see two cycles here; T1 → T2 → T3 → T1 and T2 → T3 → T2**. Second one is discussed below;



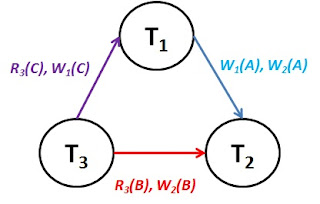
The conflicting instructions are highlighted in color below;

R1(A), R1(B), W1(A), R2(B), **W2(D)**, R3(C), **R3(B)**, **R3(D)**, **W2(B)**, W1(C), W3(D)

In the first conflict (**green color**), transaction T2 is writing D and T3 is reading D. In the second one, transaction T3 is reading B and T2 is writing B. And these two conflicts end up in creating a cycle. **Hence, schedule S1 is not conflict-serializable**.

**(ii) Schedule S2:**

Schedule S2 is conflict serializable because the precedence graph for S3 doesn’t have a cycle in it (refer the image below).



Serialization order for this schedule is T3, T1, T2. That is, the execution of this schedule will be same as executing transactions in the order T3, T1 and T2.

4) Consider the following relational database schema consisting of the four relation schemas:

**passenger**( pid, pname, pgender, pcity)

**agency**( aid, aname, acity)

**flight**(fid, fdate, time, src, dest)

**booking**(pid, aid, fid, fdate)

Answer the following questions using relational algebra queries;

i) Get the complete details of all flights to New Delhi.

ii) Find only the flight numbers for passenger with pid 123 for flights to Chennai before

06/11/2020.

iii) Find the passenger names for those who do not have any bookings in any flights.

iv) Get the details of flights that are scheduled on both dates 01/12/2020 and 02/12/2020 at

16:00 hours.

 v) Find the details of all male passengers who are associated with Jet agency. 10M

i) σ *destination = “New Delhi”*(flight)

ii) Π*fid*(σ*pid = 123*(booking) ⨝ σ*dest = “Chennai” ^ fdate < 06/11/2020*(flight))

iii) Π*pname*(passenger ⨝ booking)

iv) (σ*fdate = 01/12/2020 ^ time = 16:00*(flight)) ∩ (σ*fdate = 02/12/2020 ^ time = 16:00*(flight))

v) Π*passengers.pid, pname, pcity*(σ*pgender = “Male” ^ aname = ‘Jet’*(passengers ⨝ booking ⨝ agency))

5) a)Consider the given schedule S1 with transactions T1 and T2; if the value of X at the beginning of the transactions is 100, what will be the value of X at the end of the transactions? Also, find the problem with the given schedule. 6M

|  |  |
| --- | --- |
| **Transaction T1** | **Transaction T2** |
| *READ X*  *X:=X - 20*  *WRITE X* | *READ X*  *X:=X - 30*  *WRITE X* |

**Solution:**

The value of X at the end of the transactions is 70. This value is wrong. From the given schedule, it is very clear that both transactions are reading the initial value of X as 100, both calculates the new value to be written, and both writes the new value of X. here, T1 finishes transaction first and writes it’s new X value 80. Then, T2 overwrites the value written by T1. Hence, we will see only one result which is written later.

This problem is called **Lost Update** problem.

**Lost update problem** – it happens when two transactions access the same resource in an interleaved fashion (*i.e., few instructions from T1 and few from T2 and so on*) and one of them overwrites the update made by the first.

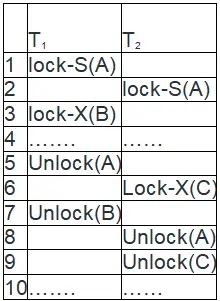
b) Illustrate two-phase locking with a schedule containing three transactions. Argue that 2PL ensures serializability. 10M

**A transaction is said to follow the Two-Phase Locking protocol if Locking and Unlocking can be done in two phases.**

**.**Growing Phase: New locks on data items may be acquired but none can be released.

.  Shrinking Phase: Existing locks may be released but no new locks can be acquired.

**Example**



Transaction T1:

* The growing Phase is from steps 1-3.
* The shrinking Phase is from steps 5-7.
* Lock Point at 3

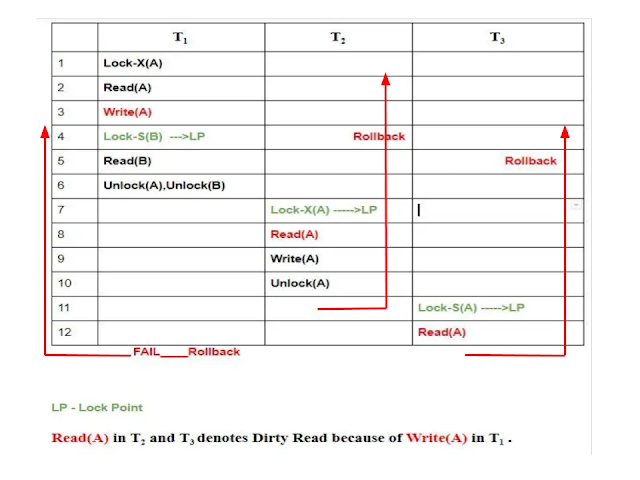
Transaction T2:

* The growing Phase is from steps 2-6.
* The shrinking Phase is from steps 8-9.
  + Lock Point at 6

LOCK POINT is the Point at which the growing phase ends, i.e., when a transaction takes the final lock it needs to carry on its work

**drawbacks of 2-PL**

* Cascading Rollback is possible under 2-PL.
* Deadlocks are possible.



On analyzing the schedule, Dirty Read in T2 and T3 in lines 8 and 12 respectively, when T1 failed we have to roll back others also. Hence, Cascading Rollbacks are possible in 2-PL.

6) Write a program in PL/SQL to update the salary of a specific employee by 8% if the salary exceeds the mid range of the salary against this job and update up to mid range if the salary is less than the mid range of the salary, and display a suitable message. 10M

DECLARE

emp\_min\_salary NUMBER(6,0);

emp\_max\_salary NUMBER(6,0);

emp\_mid\_salary NUMBER(6,2);

tmp\_salary EMPLOYEES.SALARY%TYPE;

tmp\_emp\_id EMPLOYEES.EMPLOYEE\_ID%TYPE := 167;

tmp\_emp\_name EMPLOYEES.FIRST\_NAME%TYPE;

BEGIN

SELECT min\_salary,

max\_salary

INTO emp\_min\_salary,

emp\_max\_salary

FROM JOBS

WHERE JOB\_ID = (SELECT JOB\_ID

FROM EMPLOYEES

WHERE EMPLOYEE\_ID = tmp\_emp\_id);

-- calculate mid-range

emp\_mid\_salary := (emp\_min\_salary + emp\_max\_salary) / 2;

-- get salary of the given employee

SELECT salary,first\_name

INTO tmp\_salary,tmp\_emp\_name

FROM employees

WHERE employee\_id = tmp\_emp\_id;

-- update salary

IF tmp\_salary < emp\_mid\_salary THEN

UPDATE employees

SET salary = emp\_mid\_salary

WHERE employee\_id = tmp\_emp\_id;

ELSE

UPDATE employees

SET salary = salary + salary \* 8 /100

WHERE employee\_id = tmp\_emp\_id;

END IF;

--display message

IF tmp\_salary > emp\_mid\_salary THEN

DBMS\_OUTPUT.PUT\_LINE('The employee '||tmp\_emp\_name||' ID ' || TO\_CHAR(tmp\_emp\_id) ||

' works in salary ' || TO\_CHAR(tmp\_salary) ||

' which is higher than mid-range of salary ' || TO\_CHAR(emp\_mid\_salary));

ELSIF tmp\_salary < emp\_mid\_salary THEN

DBMS\_OUTPUT.PUT\_LINE('The employee '||tmp\_emp\_name||' ID ' || TO\_CHAR(tmp\_emp\_id) ||

' works in salary ' || TO\_CHAR(tmp\_salary) ||

' which is lower than mid-range of salary ' || TO\_CHAR(emp\_mid\_salary));

ELSE

DBMS\_OUTPUT.PUT\_LINE('The employee '||tmp\_emp\_name||' ID ' || TO\_CHAR(tmp\_emp\_id) ||

' works in salary ' || TO\_CHAR(tmp\_salary) ||

' which is equal to the mid-range of salary ' || TO\_CHAR(emp\_mid\_salary));

END IF;

END;

/

7) Why concurrency control and recovery needed in DBMS? Explain Types of Problems that may occur when two simple transactions run concurrently. 10M

DBMS has a Concurrency Control sub-system to assure database remains in consistent state despite concurrent execution of transactions The DBMS must not permit some operations of a transaction T to be applied to the database while other operations of T are not, because the whole transaction is a logical unit of database processing. If a transaction fails after executing some of its operations but before executing all of them, the operations already executed must be undone and have no lasting effect. Types of Failures. Failures are generally classified as transaction, system, and media failures. There are several possible reasons for a transaction to fail in the middle of execution:

1. A computer failure (system crash). A hardware, software, or network error occurs in the computer system during transaction execution. Hardware crashes are usually media failures—for example, main memory failure.

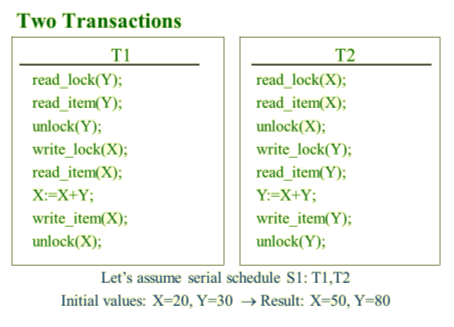
2. A transaction or system error. Some operation in the transaction may cause it to fail, such as integer overflow or division by zero. Transaction failure may also occur because of erroneous parameter values or because of a logical programming error. Additionally, the user may interrupt the transaction during its execution.

3. Local errors or exception conditions detected by the transaction. During transaction execution, certain conditions may occur that necessitate cancellation of the transaction. For example, data for the transaction may not be found. An exception condition, such as insufficient account balance in a banking database, may cause a transaction, such as a fund withdrawal, to be cancelled. This exception could be programmed in the transaction itself, and in such a case would not be considered as a transaction failure.

4. Concurrency control enforcement. The concurrency control may abort a transaction because it violates serializability ,or it may abort one or more transactions to resolve a state of deadlock among several transactions (see Section Transactions aborted because of serializability violations or deadlocks are typically restarted automatically at a later time.

5. Disk failure. Some disk blocks may lose their data because of a read or write malfunction or because of a disk read/write head crash. This may happen during a read or a write operation of the transaction.

6. Physical problems and catastrophes. This refers to an endless list of problems that includes power or air-conditioning failure, fire, theft, sabotage, overwriting disks or tapes by mistake, and mounting of a wrong tape by the operator.



8) Consider a relation R(A, B, C, D) with the set of functional dependencies F = {AB → C, BC → D, CD → A}. Assume that R is decomposed into R1(A, B, C) and R2(A, C, D). Find whether the given decomposition is lossless or not. 8M

**Solution:**

Lossless join decomposition implies that the result of joining all the decomposed relations will create the base relation again without any loss/gain in data.

If one of the following is true, then the decomposition is said to be lossless;

* (R1 ∩ R2) → R1
* (R1 ∩ R2) → R2

If we apply intersection between R1 and R2, we shall get,

(R1 ∩ R2) = {A, B, C} ∩ {A, C, D} = AC.

There is no functional dependency in F such that the AC is alone on the left hand side. Hence, this decomposition is lossless.

**Example:**

Let us populate R with sample data and try the experiment;

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **D** |
| a1 | a2 | a3 | a4 |
| a1 | a4 | a3 | a2 |

According to the decomposition, we shall get R1 and R2 as follows;

|  |  |  |
| --- | --- | --- |
| **R1** | | |
| **A** | **B** | **C** |
| a1 | a2 | a3 |
| a1 | a4 | a3 |
|  |  |

|  |  |  |
| --- | --- | --- |
| **R2** | | |
| **A** | **C** | **D** |
| a1 | a3 | a4 |
| a1 | a3 | a2 |

Join back R1 and R2 must result in R if the decomposition is lossless.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| R1 | ⋈ | R2 | = | R’ |
| |  |  |  | | --- | --- | --- | | **A** | **B** | **C** | | a1 | a2 | a3 | | a1 | a4 | a3 | | ⋈ | |  |  |  | | --- | --- | --- | | **A** | **C** | **D** | | a1 | a3 | a4 | | a1 | a3 | a2 | | = | |  |  |  |  | | --- | --- | --- | --- | | **A** | **B** | **C** | **D** | | a1 | a2 | a3 | a4 | | a1 | a2 | a3 | a2 | | a1 | a4 | a3 | a2 | | a1 | a4 | a3 | a4 | |

9) Consider a relation Movies\_Screened with attributes Theatre, Movie, Day, Time, and

Certificate. Sample tuples are as follows:

***Sathyam, 'Slumdog Millionaire', Wed, 18:00, 15***

***Sathyam, 'Slumdog Millionaire', Wed, 20:00, 15***

***PVR, 'Slumdog Millionaire', Wed, 20:30, 15***

***PVR, 'Vicky Christina Barcelona', Wed, 20:30, 12A***

Each movie is assigned a certificate by the Indian Board of Film Certification; the

certificate value 15 means that nobody younger than 15 years of age can see this movie in a

cinema. The same theatre can show a movie on multiple times during a day, and may show

different movies at the same time (on different screens).

(a) Does this relation violate the second normal form requirements? Explain.

(b) Decompose this relation into BCNF, and explain why the resulting relations are in BCNF. 10M

**Solution:**  
a) To check for 2NF, first we need to find the candidate keys for MOVIES\_SCREENED.

Let us find the functional dependencies (FDs) of MOVIES\_SCREENED.

* THEATRE cannot determine any attributes as a theatre screens more than one movie, it screens on different days, different timings, and different certification movies.
* MOVIE can determine the CERTIFICATE value as a movie will be given only one certificate. Hence, we can include MOVIE **→** CERTIFICATE.
* Likewise, DAY, TIME and CERTIFICATE cannot determine the other attributes uniquely.

We get the set of FDs for this relation as follows;

F = { MOVIE **→** CERTIFICATE, (THEATRE, MOVIE, DAY, TIME) **→** CERTIFICATE }

To find the candidate key, we need to find the closure of left hand side attributes of the FDs.

(THEATRE, MOVIE, DAY, TIME)+ = THEATRE, MOVIE, DAY, TIME, CERTIFICATE.

Hence, the composite key (THEATRE, MOVIE, DAY, TIME) is the candidate key for the relation MOVIES\_SCREENED.

**To be in 2NF, a relation should not have partial functional dependency**.

In our relation, a non-key attribute CERTIFICATE is determined by MOVIE, which is part of a candidate key (THEATRE, MOVIE, DAY, TIME). So the given relation is not in 2NF.

**The relation MOVIES\_SCREENED violates second normal form**.   
  
b) As discussed, the relation violates 2NF. To normalize to 2NF, we decompose the relation

using the violating functional dependency MOVIE **→** CERTIFICATE.

It results in the following relations;

Movie\_Screens (THEATRE, MOVIE, DAY, TIME)

Movies (MOVIE, CERTIFICATE).

Both relations are in 2NF because no partial dependency exists [see the keys underlined].

Both relations are in 3NF too because no transitive dependencies found.

Also, both are in BCNF because in the Movie\_Screens relation, no subset of the attributes determines any other attribute, and the only non-trivial dependency in MOVIES is from MOVIES to CERTIFICATE.

10) Consider a file with 2,00,000 records stored in a disk with fixed length block of size 256

byte. Each record is of size 50 bytes. The primary key is 4 bytes and block pointer is 6

bytes. Compute the following , assuming that multi-level primary index is used as access

path:

1. Blocking factor for data records
2. Blocking factor for index records
3. Number of data blocks
4. Number of First level index blocks
5. Number of level of multi level index 10M

