Model Question paper

QPCODE

Reg No:________________

Name:_________________

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

FOURTH SEMESTER B.TECH DEGREE EXAMINATION, MONTH & YEAR

Course Code: CST 204

Course Name: **Database Management Systems**

Max.Marks:100 **Duration: 3 Hours**

PART A

Answer all Questions. Each question carries 3 Marks

1 List out any three salient features of a database systems.

1. Self-describing nature of a database system
2. Insulation between programs and data, and data abstraction
3. Support of multiple views of the data
4. Sharing of data and multiuser transaction processing

2 When is multi-valued composite attribute used in ER modelling?

A **multivalued attributed** in DBMS(Data Base Management System) is an attribute that can have multiple value for a single intance of an entity.

Phone number of a student: Landline and mobile.

3 For the SQL query, SELECT *A, B FROM R WHERE B='apple' AND C = 'orange'* on the table R(A, B, C, D), where A is a key, write any two equivalent relational algebra expressions.

ΠA,B(σ b='apple' and c='orange(R));

4 Outline the concept of *theta*-join.

- Theta join is a join which combines the tuples from different relations according to the given theta condition.
- The join condition in theta join is denoted by **theta(θ) symbol.** θ (theta) is one of the comparison operators {=, <, ≤, >, ≥, ≠}.
- **Notation:**$R1 \bowtie_{\theta} R2$

  Where R1 and R2 are relations such that they don't have any common attribute.

5 How is the purpose of *where* clause is different from that of having clause?

- A HAVING clause is like a WHERE clause, but applies only to groups as a whole (that is, to the rows in the result set representing groups),
- WHERE clause applies to individual rows. A query can contain both a WHERE clause and a HAVING clause.

6 What is the use of a trigger?

- A trigger in SQL is a procedural code that is automatically executed in response to certain events on a specified table.

7 When do you say that a relation is not in 1NF?

- When a relation consists of a multi-valued or composite attribute, it would be violating the 1NF

8 Given the FDs P→Q, P→R, QR→S, Q→T, QR→U, PR→U, write the sequence of Armstrong's Axioms needed to arrive at a. P → T b. PR → S

a) Transitivity Rule

If P→Q, Q→T, then P → T

b) Pseudo Transitivity Rule

If P→R, QR→S, then PR → S

9 What is meant by the lost update problem?

- This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database items incorrect.
- Suppose that transactions T1 and T2 are submitted at approximately the same time, and suppose that their operations are interleaved.

(a)

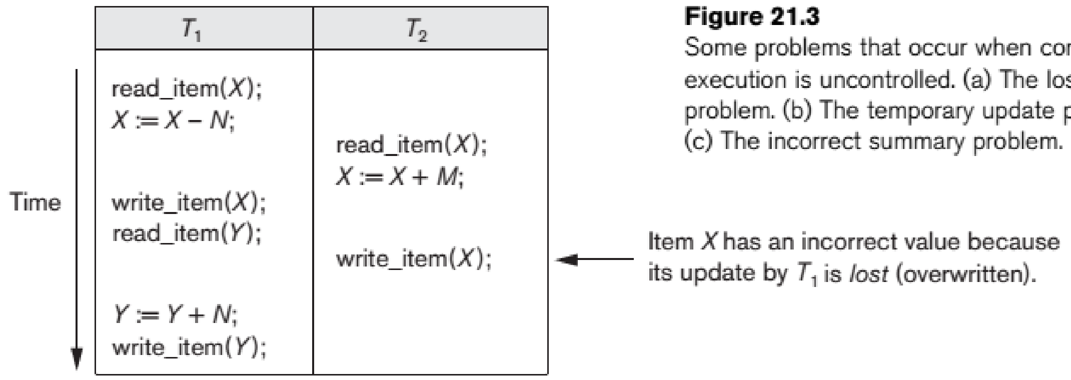


Figure 21.3
Some problems that occur when concurrent execution is uncontrolled. (a) The lost update problem. (b) The temporary update problem. (c) The incorrect summary problem.

10 What is meant by check pointing?

- Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk.

PART B

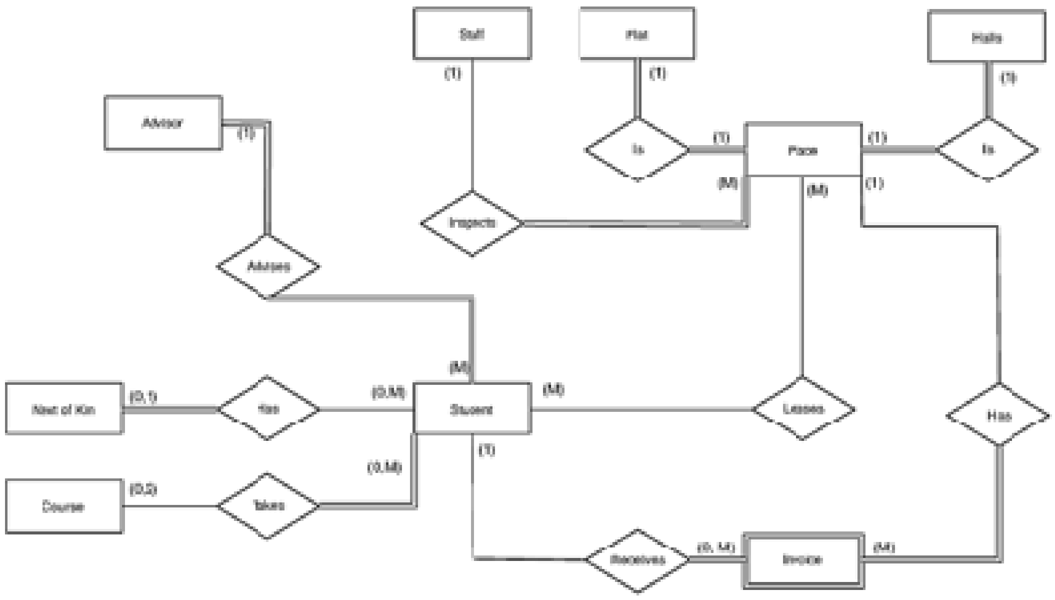
Answer any one Question from each module. Each question carries 14 Marks

11 a. Design an ER diagram for the following scenario: There is a set of teams, each team has an ID (unique identifier), name, main stadium, and to which city this team belongs. Each team has many players, and each player belongs to one team. Each player has a number (unique identifier), name, DoB, start year, and shirt number that he uses. Teams play matches, in each match there is a host team and a guest team. The match takes place in the stadium of the host team. For each match we need to keep track of the following: The date on which the game is played The final result of the match. The players participated in the match. For each player, how many goals he scored, whether or not he took yellow card, and whether or not he took red card. During the match, one player may substitute another player. We want to capture this substitution and the time at which it took place. Each match has exactly three referees. For each referee we have an ID (unique identifier), name, DoB, years of experience. One refereeis the main referee and the other two are assistant referee.

****Refer answer in the last**

OR

12 a. Interpret the the following ER diagram.



b. Distinguish between physical data independence and logical data independence with suitable examples.

Physical Data Independence:
The physical data independence is basically used to separate conceptual levels from the internal/physical levels. It is easy to achieve physical data independence. With this type of independence, user is able to change the physical storage structures or the devices which have an effect on the conceptual schema.
Examples of changes under Physical Data Independence:
- It is by the use of new storage device like Hard Drive or Magnetic Tapes
- Modifying the file organization technique in the Database
- Switching to different data structures.
- Changing the access method.
- Modifying indexes.
- To change the compression techniques or hashing algorithms.
- To change the Location of Database from say C drive to D Drive.

2. Logical Data Independence:
Logical Data Independence is used to change the conceptual scheme without changing the following things:
- External views
- External API or programs

Examples of changes under Logical Data Independence:
- To Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs
- Merging two records into one
- To break an existing record i.e to divide the record into two or more records

13 **EMPLOYEE(<u>ENO</u>, NAME, ADDRESS, DOB, AGE, GENDER, SALARY, DNUM, SUPERENO)** (14)
DEPARTMENT(<u>DNO</u>, DNAME, DLOCATION, DPHONE, MGRENO)
PROJECT(<u>PNO</u>, PNAME, PLOCATION, PCOST, CDNO)

DNUM is a foreign key that identifies the department to which an employee belongs. MGRENO is a foreign key identifying the employee who manages the department. CDNO is a foreign key identifying the department that controls the project. SUPERENO is a foreign key identifying the supervisor of each employee.

Write relational algebra expressions for the following queries:-

(a) Names of female employees whose salary is more than 20000.
(b) Salaries of employee from 'Accounts' department
(c) Names of employees along with his/her superviser's name
(d) For each employee return name of the employee along with his department name and the names of projects in which he/she works
(e) Names of employees working in all the departments

a)πName (σGender='female' AND Salary> '20000' (Employee))

b)πName,Salary (Employee ⋈Dnum=Dno AND Dname='Accounts' Department)

c)πEmplyee.Name,Emp1.Name (Employee ⋈Employee.SuperEno=Emp1.Eno1 AND Dname=' Accounts') ρ_Emp1 (Employee))

c) Requires additional info like an employee works for one project or more than one project, a project is worked by one employee or more than one employee

d) Whether we need to assume eno not be the primary key

OR

14 a.Write SQL DDL statements for the the following (Assume suitable domain types): (10)

i. Create the tables STUDENT(<u>ROLLNO</u>, NAME, CLASS, SEM, ADVISER), FACULTY(<u>FID</u>, NAME, SALARY, DEPT). Assume that ADVISER is a foreign key referring FACUTY table.

SQL>Create tables STUDENT(ROLLNO int primary key, NAME text not null, CLASS varchar(10) ,SEM int , ADVISER text , Foreign key (Adviser) references faculty(name));

ii. Delete department with name 'CS' and all employees of the department.

SQL>DELETE FROM DEPARTMENT WHERE NAME='CS';

iii. Increment salary of every faculty by 10%.

SQL> UPDATE DEPARTMENT SET Salary =.01*Salary+Bal ;

b. illustrate foreign key constraint with a typical example.(4)

- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the <u>PRIMARY KEY</u> in another table.
- The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.
- **Persons Table**

PersonID	Last Name	First Name	Age
1.	Hansen	Ola	30
2.	Svendson	Tove	23
3.	Pettersen	Kari	20

- **Orders Table**

OrderID	OrderNumber	PersonID
1.	77895	3
2.	44678	3
3.	22456	2
4,	24562	1

- In the Orders Table PersonID act as the foreign key because it is refereed from primary key of Persons' table PersonID

15 For the relation schema below, give an expression in SQL for each of the queries that follows: (14)

employee (<u>employee-name</u>, street, city)
works(<u>employee-name, company-name</u>, salary)
company(<u>company-name</u>, city)
manages(<u>employee-name, manager-name</u>)

a) Find the names, street address, and cities of residence for all employees who work for the Company ‘RIL Inc.' and earn more than $10,000.

SQL>select employee.employee-name, employee.street, employee.city from employee, works where employee.employee-name=works.employee-name and company-name = ' RIL Inc.' and salary > 10000)

b) Find the names of all employees who live in the same cities as the companies for which they work.

SQL>select e.employee-name from employee e, works w, company c where e.employee-name = w.employee-name and e.city = c.city and w.company-name = c.company-name;

c) Find the names of all employees who do not work for ‘KYS Inc.’. Assume that all people work for exactly one company.

SQL>select employee-name from works where company-name <> ‘KYS Inc.';

d) Find the names of all employees who earn more than every employee of ‘SB Corporation'. Assume that all people work for at most one company.

SQL>select employee-name from works where salary > all (select salary from works where company-name = 'SB Corporation');

e) List out number of employees company-wise in the decreasing order of number of employees.
SQL>SELECT employee-name, COUNT(*) FROM works GROUP BY company-name ORDER BY company-name DESC;

OR

16 a. Consider an EMPLOYEE file with 10000 records where each record is of size 80 bytes. The file is sorted on employee number (15 bytes long), which is the primary key. Assuming un-spanned organization and block size of 512 bytes compute the number of block accesses needed for selecting records based on employee number if, (9)

i. No index is used
ii. Single level primary index is used
iii. Multi-level primary index is used

Assume a block pointer size of 6 bytes.

(i) <u>No index is used</u>

Block size=512 bytes
Record Size= 80 bytes
Block Factor = Block Size/Record Size= 512/80 = 6
Hence 6 records per block
Total number of Blocks = 10000/6=1667 blocks
To find a record in the block we can use binary search=O(log2 Bi)
i.e log2 1667= 11

<u>ii. Single level primary index is used</u>

The primary key is 15 bytes and the block pointer size is 6 bytes
The indexing size will be 15+6=21
Block size=512 bytes
Indexing=512/21=24
Total blocks formed in indexing =1667/24=70
To find a record in the block =O(log2 Bi)
i.e log2 (70+1) = 6
(+1 is for the data access part)

<u>iii. Multi-level primary index is used</u>

Number of second-level index = number of Single level index/Bfr=70/6=12
Number of the third levels index= 12/6=2
Number of the fourth levels index=2/6=1
The number of levels in the Multi-level primary index is 4

b. Illustrate correlated and non-correlated nested queries with real examples. (5)

Subqueries can be categorized into two types:

- A noncorrelated (simple) subquery obtains its results independently of its containing the (outer) statement.
- A correlated subquery requires values from its outer query in order to execute.

Correlated Subqueries

The correlated sub-query usually gets values from its external query before it starts. When the subquery returns, it relays its results to an external query.

In the following example, the subquery needs values from the addresses. state column in the outer query:

```
SELECT name, street, city, state FROM addresses
    WHERE EXISTS (SELECT * FROM states WHERE states.state = addresses.state);
```

- The query extracts and evaluates each address. state value in the outer subquery records.
- Then the query—using the EXISTS predicate—checks the addresses in the inner (correlated) subquery.
- Because it uses the EXISTS predicate, the query stops processing when it finds the first match.

Non correlated Sub queries

A noncorrelated sub-query is signed without an external query. The subquery starts first and then transmits its results to an external query
For example:

17 a. Illstrate3NF and BCNF with suitable real examples. (6)

b. Given a relation R(A1,A2,A3,A4,A5) with functional dependencies A1→A2A4 and A4→A5, check if the decomposition R1(A1,A2,A3), R2(A1,A4), R3(A2,A4,A5) is lossless.

a)3NF
A relation is in the third normal form if there is no transitive dependency for non-prime attributes as well as it is in the second normal form.
Two Conditions for 3NF

- It should be in 2NF
- No transitive Dependency
  - What is Transitive dependency?
  - If a Non-prime attribute->Non-prime attribute, then it is called a Transitive dependency

Example:

- R(A,B,C,D)
  - FD:- (*A->B,B->C,C->D*)
  - Check whether the above relation is in 3NF

Solution

- Find the attribute closure of all the attributes
  - $ABCD^+$={A,B,C,D}
  - $ABCD^+$ is a superkey

Find out the candidate key

  - Discard the element by checking the FD
  - ABCD
  - Check A is a superkey

- A^+={A, B, C, D} ->A is a candidate key because there is no proper subset for identifying a superkey

- Check out more candidate keys are possible

  Check whether the prime attribute A is present on the right-hand side of the FD
  - In this problem, no prime attribute is there
  - Hence no more candidate key
- Now what we have identified.
  - A is the only candidate key
  - Prime Attribute is A
- Now check all the FD whether satisfies the following conditions

It should be in 2NF

- No transitive Dependency

  A->B,B->C,C->D

Hence the given relation is not in 3NF and it is in 2NF

BCNF

- A relation is in BCNF iff
  - It is in 3NF
  - For each non-trivial FD XY, X must be Super key

Example:

- *R(A,B,C)*
  - *FD:- (A->B,B->C,C->A*
  - *Check whether the above relation is in BCNF*

Check out more candidate keys are possible
- Check whether the prime attribute A is present on the right-hand side of the FD
- Yes prime attribute is there in the RHS of the *FD:- (A→B, B→C, C→A)*
- C, B
- Hence the candidate keys are A, B, C
- Hence prime attributes are A, B, C

- Check for BCNF Conditions
  - It is in 3NF
  - For each non-trivial FD X→Y, X must be super key

Hence the given relation is BCNF

b)

Based on property 1:
R1 U R2 U R3 = R
R1(A1,A2,A3) U R2(A1,A4) U R3(A2,A4,A5) = R(A1,A2,A3,A4,A5)
Property 1 satisfies
Based on property 2 common attributes are there in R1, R2 and R3
Based on property 3:
R1(A1,A2,A3), R2(A1,A4)

Common attribute for R1 and R2 is A1. Find the Attribute closure for A1
A1+ = {A1,A2,A4,A5}
A1 is candidate key for R2

R1 ⋈ R2=A1,**A2**,A3,**A4**
Now check with R3 (**A2,A4**,A5)
Here A2 and A4 are the common attribute and therefore find attribute closure for A2A4
A2A4+ = A2,A4,A5
A2A4 is the candidate key for R3 and

Hence it is Lossless

R1 ⋈ R2=A1,**A2**,A3,**A4**
Now check with R3 (**A2,A4**,A5)
Here A2 and A4 are the common attribute and therefore find attribute closure for A2A4
A2A4+ = A2,A4,A5
A2A4 is the candidate key for R3 and

Hence it is Lossless

18 a. Consider the un-normalized relation R(A, B, C, D, E, F, G) with the FDs A→B , AC→G, AD→EF, EF→G, CDE→AB. Trace the normalization process to reach 3NF relations.

ABCDEFG+={A,B,C,D,E,F,G} is a super key

ACD+={A,C,D,B,G,E,F} is a super key

Proper subset of ACD+ = {AC,CD,AD,A,C,D}

- AC+ is not a super key
- CD+ is not a super key
- AD+ is not a superkey
- A+ is not a super key
- C+ is not a super key
- D+ is not a super key

Hence **ACD** is a candidate key

Now prime attributes are A,C,D

CDE+={C,D,E,A,B,E,F,G} is a superkey

Proper subset of ACD+ = {AC,CD,AD,A,C,D}

- CD+ is not a super key

- DE+ is not a super key
- CE+ is not a super key
- C+ is not a super key
- D+ is not a super key
- E+ is not a super key

Hence **CDE** is a candidate key

Now prime attributes are A,C,D,E

Now check for transitive dependency

A→B --> No transitive dependency

AC→G --> No transitive dependency

AD→EF --> No transitive dependency

EF→G --> No transitive dependency

CDE→AB --> No transitive dependency

Hence relation R(A, B, C, D, E, F, G) with the FDs A→B , AC→G, AD→EF, EF→G, CDE→AB is in 3NF

b. Illustrate Lossless Join Decomposition and Dependency PreservingDecomposition with typical examples.

Lossless Join Decomposition

If we decompose a relation R into relations R1 and R2,

- Decomposition is lossy if R1 ⋈ R2 ⊃ R
- Decomposition is lossless if R1 ⋈ R2 = R

To check for lossless join decomposition using FD set, following conditions must hold:

1. Union of Attributes of R1 and R2 must be equal to attribute of R. Each attribute of R must be either in R1 or in R2.
   Att(R1) U Att(R2) = Att(R)
2. Intersection of Attributes of R1 and R2 must not be NULL.
   Att(R1) ∩ Att(R2) ≠ Φ
3. Common attribute must be a key for at least one relation (R1 or R2)
   Att(R1) ∩ Att(R2) -> Att(R1) or Att(R1) ∩ Att(R2) -> Att(R2)

For Example, A relation R (A, B, C, D) with FD set{A->BC} is decomposed into R1(ABC) and R2(AD) which is a lossless join decomposition as:

1. First condition holds true as Att(R1) U Att(R2) = (ABC) U (AD) = (ABCD) = Att(R).
2. Second condition holds true as Att(R1) ∩ Att(R2) = (ABC) ∩ (AD) ≠ Φ
3. Third condition holds true as Att(R1) ∩ Att(R2) = A is a key of R1(ABC) because A->BC is given.

Dependency Preserving Decomposition

If we decompose a relation R into relations R1 and R2, All dependencies of R either must be a part of R1 or R2 or must be derivable from combination of FD's of R1 and R2.

For Example, A relation R (A, B, C, D) with FD set{A->BC} is decomposed into R1(ABC)

and R2(AD) which is dependency preserving because FD A->BC is a part of R1(ABC).

Example
Consider a schema **R(A,B,C,D)** and functional dependencies **A->B and C->D** which is decomposed into **R1(AB) and R2(CD)**
This decomposition is **dependency preserving decomposition** because

- **A->B** can be ensured in **R1(AB)**
- **C->D** can be ensured in **R2(CD)**

19.a. Discuss the four ACID properties and their importance. (7)

A transaction is a single logical unit of work which accesses and possibly modifies the contents of a database.
  – Transactions access data using read and write operations.
- In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called ACID properties

Atomicity

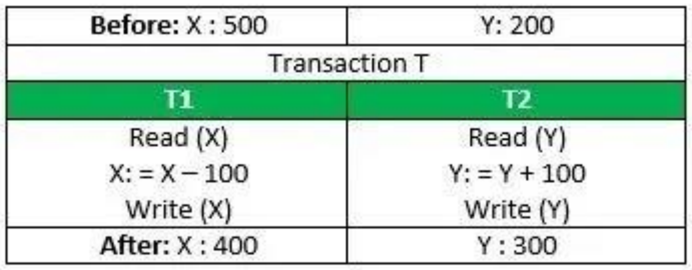
- By this, we mean that either the entire transaction takes place at once or doesn't happen at all.
- There is no midway i.e. transactions do not occur partially.
- Each transaction is considered as one unit and either runs to completion or is not executed at all.
- It involves the following two operations.
  —**Abort**: When a job is terminated, changes made to database are not visible.
  —**Commit**: If a transaction commits, changes made are visible.
  Atomicity is also known as the 'All or nothing rule'.

Atomicity Example

- Consider the following transaction **T** consisting of **T1** and **T2**: Transfer of 100 from account **X** to account **Y**.

If the transaction fails after completion of **T1** but before completion of **T2**.(say, after **write(X)** but before **write(Y)**), then amount has been deducted from **X** but not added to **Y**.
- This results in an inconsistent database state.

- Therefore, the transaction must be executed in entirety in order to ensure correctness of database state.

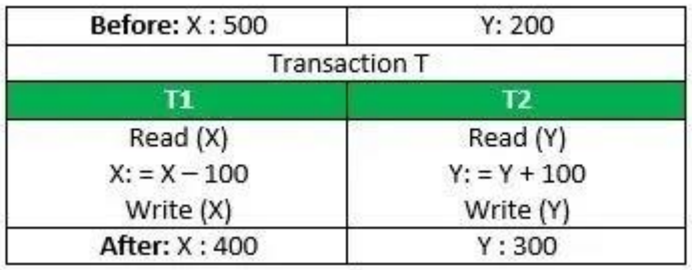


Before: X : 500	Y: 200
Transaction T	
T1	T2
Read (X) X: = X − 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

Consistency

- This means that integrity constraints must be maintained so that the database is consistent before and after the transaction.
- It refers to the correctness of a database.
- Referring to the same example,
  - The total amount before and after the transaction must be maintained.
  - Total **before T** occurs = **500 + 200 = 700**.
  - Total **after T occurs** = **400 + 300 = 700**.
  - Therefore, database is **consistent**.
  - Inconsistency occurs in case **T1** completes but **T2** fails.

As a result T is incomplete.



Before: X : 500	Y: 200
Transaction T	
T1	T2
Read (X) X: = X − 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

Isolation

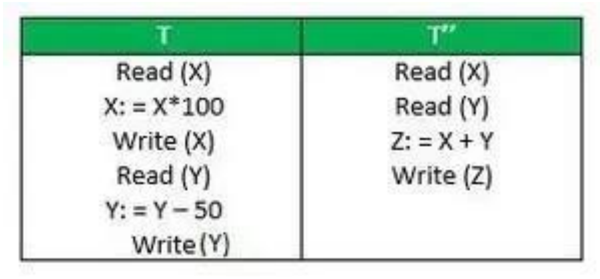
- This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of database state.
- Transactions occur independently without interference.
- Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed.
- This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order.

Isolation Example

- Let **X**= 500, **Y** = 500.
  Consider two transactions **T** and **T”.**
- Suppose **T** has been executed till **Read (Y)** and then **T’’** starts.

- As a result , interleaving of operations takes place due to which **T’’** reads correct value of **X** but incorrect value of **Y** and sum computed by
  - **T’’: (X+Y = 50, 000+500=50, 500)**
- is thus not consistent with the sum at end of transaction:
  - **T: (X+Y = 50, 000 + 450 = 50, 450)**.

This results in database inconsistency, due to a loss of 50 units. Hence, transactions must take place in isolation and changes should be visible only after they have been made to the main memory.



T	T’’
Read (X) X: = X*100 Write (X) Read (Y) Y: = Y – 50 Write (Y)	Read (X) Read (Y) Z: = X + Y Write (Z)

Durability

- This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs.
- These updates now become permanent and are stored in non-volatile memory.
- The effects of the transaction, thus, are never lost.

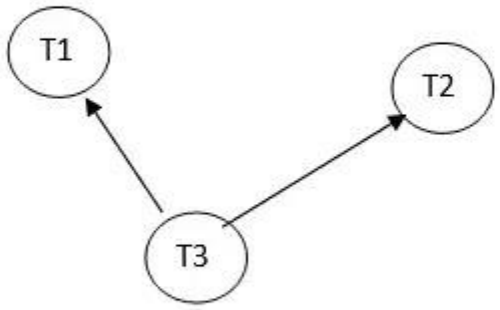
b. Determine if the following schedule is conflict serializable. Is the schedule recoverable? Is the schedule cascade-less? Justify your answers. (7)

r1(X), r2(Z), r1(Z), r3(X), r3(Y), w1(X), c1, w3(Y), c3, r2(Y), w2(Z), w2(Y), c2

(Note: ri(X)/wi(X) means transaction Ti issues read/write on item X; ci means transaction Ti commits.)

Transaction 1	**Transaction 2**	**Transaction 3**
r1(x)		
	r2(Z)	
r1(Z)		
		R3(x)
		R3(y)
w1(X)		
C1		W3(y)
		C3
	r2(Y)	
	w2(Z)	

Transaction 1	Transaction 2	Transaction 3
	w2(Y)	
	C2	



Since there is no cycle of transaction in the above schedule, it is conflict serializable.

Because all of the transactions T1, T2, and T3 commit after the transactions that wrote the data, this **schedule is recoverable**.

It is cascade-less. T1,T2,T3 read the data after all of the previous transactions that wrote the data had committed.

OR

20.a. Discuss the main characteristics of Key-value DB and Graph DB. (7)

Key-value DB

In Key-value DB data is recorded (added, updated, and deleted) and retrieved based on its key / retrieve key. A key-value website links a value to a key, which is used to track an item. In its simplest form, a key value store is similar to a dictionary / list / object item as it exists in most system paradigms, but is stored in a sustainable manner and managed by the Database Management System.

► As the name suggests, this type of NoSQL database implements a hash table to store unique keys along with the pointers to the corresponding data values.

► The values can be of scalar data types such as integers or complex structures such as JSON, lists, BLOB, and so on.

► A value can be stored as an integer, a string, JSON, or an array—with a key used to reference that value.

► It typically offers excellent performance and can be optimized to fit an organization’s needs.

► Key-value stores have no query language but they do provide a way to add and remove key-value pairs.

► Values cannot be queried or searched upon. Only the key can be queried.

Graph DB

A graphical database uses nodes, relationships between nodes and key values structures instead of tables to represent information. This model is usually very fast in integrated data sets and uses a schema-less and low-top model ready to capture ad hoc and fast-changing data.

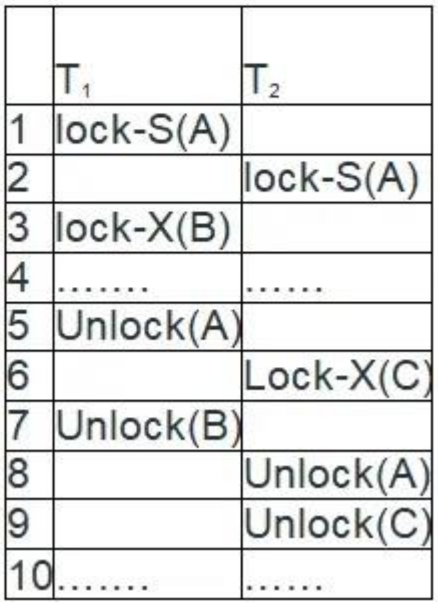
Graph database is designed for the purpose of maintaining and navigating relationships. Relationships are first-class citizens on a graph site, and most of the value of graph information is derived from these relationships. Graphical data uses nodes to store data entities, and margins to maintain relationships between businesses.

b)Illustrate two-phase locking with a schedule containing three transactions. Argue that 2PL ensures serializability. Also argue that 2Pl can lead to deadlock.

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

1. . Growing Phase: New locks on data items may be acquired but none can be released.
2. . Shrinking Phase: Existing locks may be released but no new locks can be acquired.

Example



	T_1	T_2
1	lock-S(A)	
2		lock-S(A)
3	lock-X(B)	
4	…….	……
5	Unlock(A)	
6		Lock-X(C)
7	Unlock(B)	
8		Unlock(A)
9		Unlock(C)
10	…….	……

Transaction T1:

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

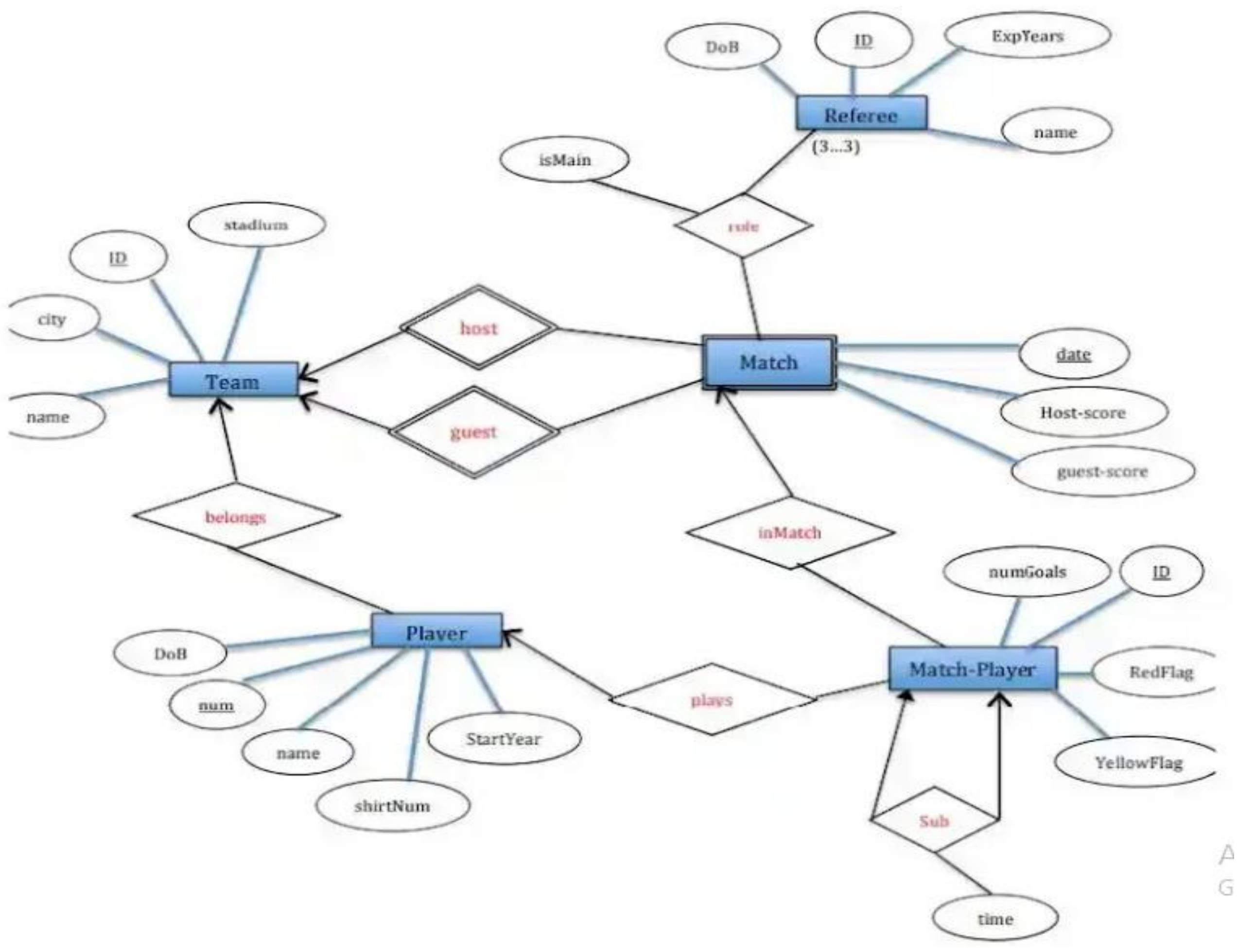
Transaction T2:

- 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.
    - **Deadlock in 2-PL**
    - Consider this example. We have two transactions T1 and T2.
    - **Schedule:** Lock-X1(A) Lock-X2(B) Lock-X1(B) Lock-X2(A)
    - Drawing the precedence graph, the loop is detected. So Deadlock is also possible in 2-PL.


DoB
ID
ExpYears
Referee
name
(3...3)
isMain
role
stadium
ID
city
Team
name
host
guest
Match
date
Host-score
guest-score
belongs
inMatch
numGoals
ID
Player
DoB
num
name
shirtNum
StartYear
plays
Match-Player
RedFlag
YellowFlag
Sub
time

A
G