Printed Pages: 3 **Sub Code: KCS501/NCS502**

Paper Id: | | | | | | | **Roll No.** | | | | | | | | | | | | |

B. Tech.
(SEM V) THEORY EXAMINATION 2022-23
DATABASE MANAGEMENT SYSTEM

Time: 3 Hours ***Total Marks: 100***

SECTION A

1. Attempt all questions in brief.

a) List the disadvantages of the file system over the database approach.

Ans : The disadvantages are as follows:

- Data Redundancy: It's possible that the same data is duplicated in many files. This results in data redundancy and memory waste.
- Data Inconsistency: Due to data redundancy, it's probable that data won't be consistent.
- Accessing Data Is Difficult: It's not simple to get data out of a file management system. It isn't as convenient as it should be. When a user needs to access information, they must run specialized software. For example, we have a graphical user interface in our computers installed using which we access the files stored.
- Limited Data Sharing: Data is dispersed across multiple files. Moreover, separate files may have different formats, and these files may be housed in several directories belonging to various departments. As a result of this data segregation, data sharing between applications is challenging.
- Security constraints: File system provides less security in comparison to DBMS.
- Recovery mechanism: The file system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will be lost.

b) Difference between physical and logical data independence.

Ans. Any changes made at the lower level should not affect the higher level is known as data independence. There are **two types** of data independence:

[1] Physical Data Independence
[2] Logical Data Independence

Physical Data Independence	Logical Data Independence
Any changes made at the internal level should not affect the conceptual level.	Any changes made at the conceptual level should not affect the external level.
It is easy to retrieve.	It is difficult to retrieve because the data is mainly dependent on the logical structure of the data.
As compared to logical data independence it is easy to achieve physical data independence.	As compared to physical data independence it is not easy to achieve logical data independence.

It is concerned with the Internal schema.	It is concerned with the Conceptual schema.
Examples: changes in compression techniques, Hashing algorithms, storage devices, etc.	Example: add/modify or delete a new attribute.

c) What is the difference between the DROP and DELETE commands?

DELETE	DROP
It removes some or all the tuples from a table.	It removes the entire schema, table, domain, or constraints from the database.
It's a Data Manipulation Language command.	It's a Data Definition Language command.
Actions performed by DELETE can be rolled back as it uses a buffer.	Actions performed by DROP can't be rolled back because it directly works on actual data.
Space occupied by the table in the memory is not freed even if you delete all the tuples of the table using DELETE.	It frees the table space from memory.
WHERE clause is mainly used along with the DELETE COMMAND.	No clause is required along the DROP command.
Here the main issue is of shortage of memory.	Here the main issue is of memory fragmentation.

d) What are different Integrity Constraints?

Ans. Relational Integrity constraints are referred to conditions that must be present for a valid relation. Constraints in the Relational database management system are mostly divided into **three** main categories are:

1. Domain constraints
2. Key constraints
3. Referential integrity constraints

Domain Constraints

They can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type. Domain constraints specify that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.

Example: Create DOMAIN CustomerName

CHECK (value not NULL

The example shown demonstrates creating a domain constraint such that CustomerName is not NULL.

Key constraints

An attribute that can uniquely identify a tuple in a relation is called the key of the table. The value of the attribute for different tuples in the relation has to be unique.

Example: In the given table, CustomerID is a key attribute of Customer Table. It is most likely to have a single key for one customer, CustomerID =1 is only for the CustomerName =" Google".

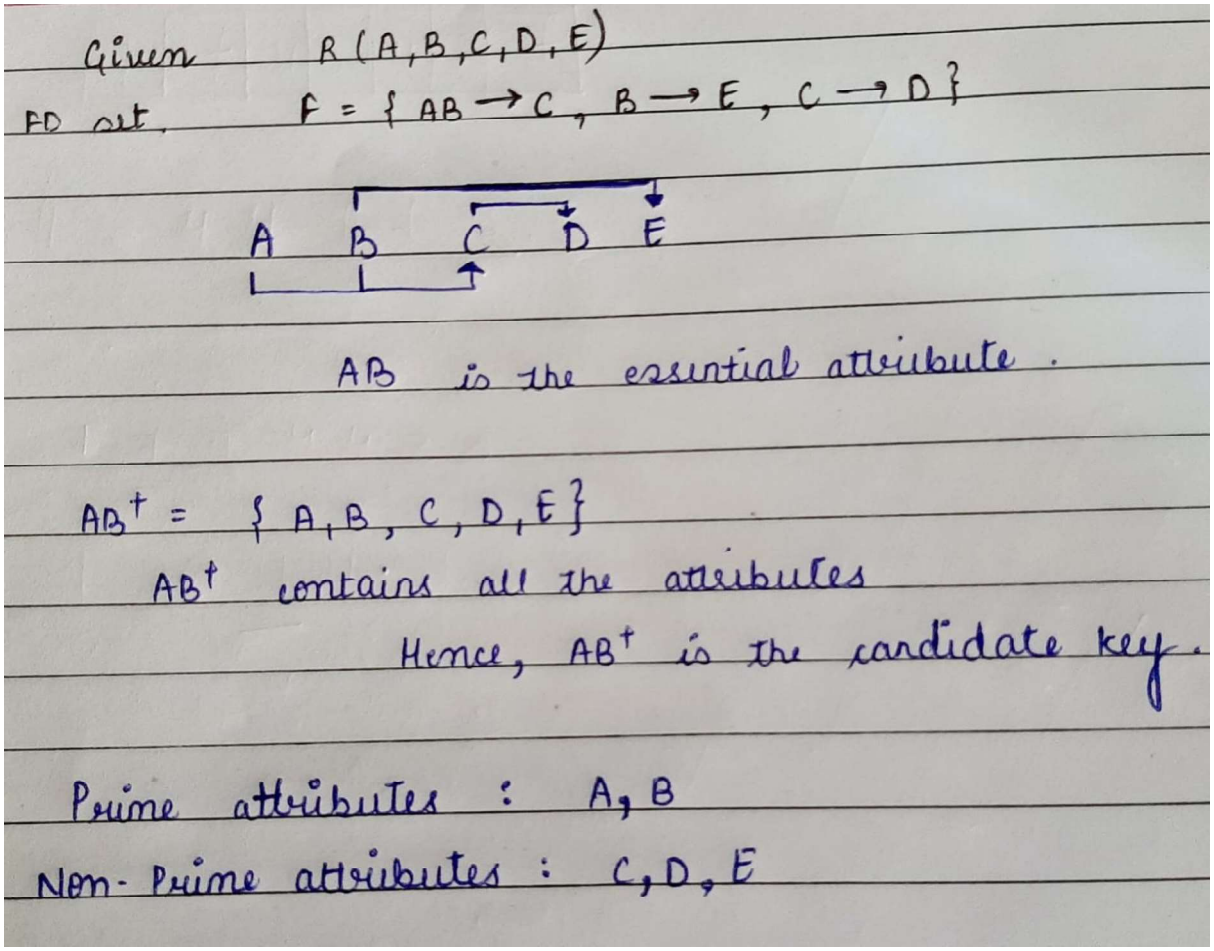
CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Referential integrity constraints

Referential integrity constraints are based on the concept of Foreign Keys. A foreign key is an important attribute of a relation that should be referred to in other relationships. A referential integrity constraint state happens where a relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

e) List all the prime and non prime attributes in relation R(A, B, C, D, E) with FD set F={ AB->C, B->E, C->D }

Ans.



f) Explain MVD with the help of a suitable example.

Ans. MVD is represented with the symbol "->->" in DBMS. A multivalued dependency prevents fourth normal form.

X->Y relates one value of X to one value of Y.

X->->Y (read as X multi determines Y) relates one value of X to many values of Y.

A multivalued dependency X→→Y specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: If two tuples t1 and t2 exist in r such that t1[X] = t2[X], then two tuples t3 and t4 should also exist in r with the following properties, 15 where we use Z to denote (R – (X ∪ Y)):

■ t3[X] = t4[X] = t1[X] = t2[X].

■ t3[Y] = t1[Y] and t4[Y] = t2[Y].

■ t3[Z] = t2[Z] and t4[Z] = t1[Z].

Whenever X→→Y holds, we say that X multi determines Y. Because of the symmetry in the definition, whenever X →→ Y holds in R, so does X →→ Z. Hence, X →→ Y implies X→→Z, and therefore it is sometimes written as X→→Y|Z. An MVD X →→ Y in R is called a trivial MVD if

(i) Y is a subset of X,

or

(ii) X ∪ Y = R

Example: Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

BIKE_MODEL	MANUF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

Here columns COLOR and MANUF_YEAR are dependent on BIKE_MODEL and independent of each other.

In this case, these two columns can be called as multivalued dependent on BIKE_MODEL. The representation of these dependencies is shown below:

```
BIKE_MODEL → → MANUF_YEAR
BIKE_MODEL → → COLOR
```

g) Discuss the Consistency and Isolation property of a transaction.

Ans. Consistency

The integrity constraints are maintained so that the database is consistent before and after the transaction. The execution of a transaction will leave a database in either its prior stable state or a new stable state. The consistent property of the database states that every transaction sees a consistent database instance. The transaction is used to transform the database from one consistent state to another consistent state.

For example, The total amount must be maintained before or after the transaction.

1. Total before T occurs = 600+300=900 2.
2. Total after T occurs= 500+400=900 Therefore, the database is consistent. In the case when T1 is completed but T2 fails, then inconsistency will occur.

Isolation

It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed. In isolation, if the transaction T1 is being executed and using the data item X, then that data item can't be accessed by any other transaction T2 until the transaction T1 ends.

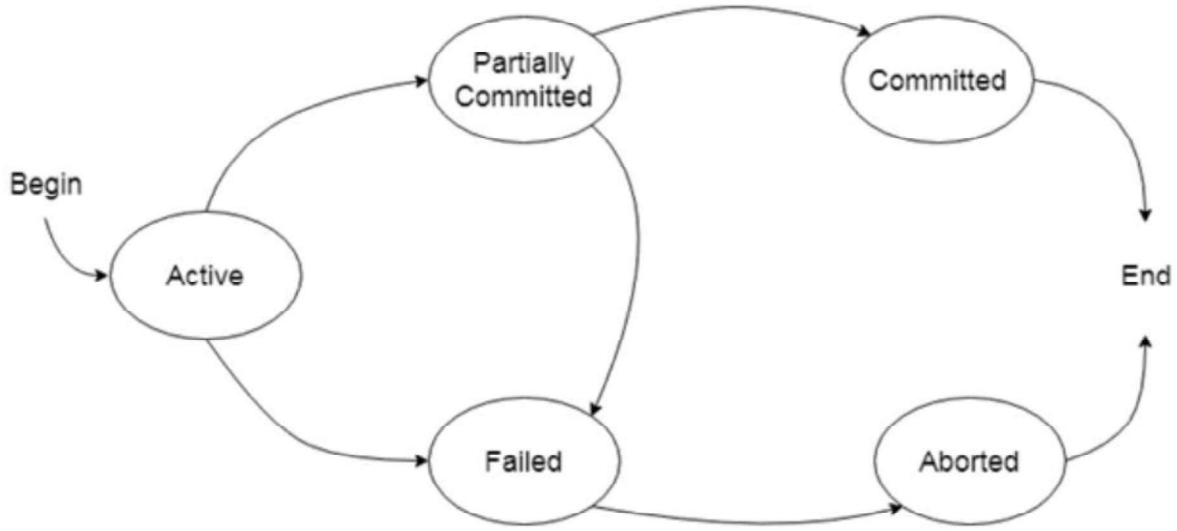
The concurrency control subsystem of the DBMS enforced the isolation property.

h) Draw a state diagram and discuss the typical states that a transaction goes through during execution.

Ans.

States of Transaction

In a database, the transaction can be in one of the following states -



Active state

The active state is the first state of every transaction. In this state, the transaction is being executed.

For example, Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

Partially committed

In the partially committed state, a transaction executes its final operation, but the data is still not saved to the database. In the total mark calculation example, a final display of the total marks step is executed in this state.

Committed

A transaction is said to be in a committed state if it executes all its operations successfully. In this state, all the effects are now permanently saved on the database system.

Failed state

If any of the checks made by the database recovery system fails, then the transaction is said to be in a failed state.

In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

Aborted

If any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous consistent state. If not then it will abort or roll back the transaction to bring the database into a consistent state. If the transaction fails in the middle of the transaction then before executing the transaction, all the executed transactions are rolled back to its consistent state.

After aborting the transaction, the database recovery module will select one of the two operations:
1. Re-start the transaction

2. Kill the transaction

i) Discuss Conservative 2PL and Strict 2PL.

Ans. Strict 2PL

Cascading rollbacks can be avoided by a modification of two-phase locking called the strict two-phase locking protocol. This protocol requires not only that locking be two phase, but also that all exclusive-mode locks taken by a transaction be held until that transaction commits. This requirement ensures that any data written by an uncommitted transaction are locked in exclusive mode until the transaction commits, preventing any other transaction from reading the data.

T_5	T_6	T_7
lock-X(A)		
read(A)		
lock-S(B)		
read(B)		
write(A)		
unlock(A)		
	lock-X(A)	
	read(A)	
	write(A)	
	unlock(A)	
		lock-S(A)
		read(A)

Conservative 2 PL

Conservative 2PL is a less restrictive form of 2PL than strict 2PL and rigorous 2PL. In conservative 2PL, a transaction is allowed to release any lock at any time, regardless of whether it will need the lock again.

The advantage of conservative 2PL is that it allows for maximum concurrency, as transactions are able to release locks at any time. This can lead to the best performance in terms of throughput and response time.

The disadvantage of conservative 2PL is that it does not guarantee serializability and can lead to inconsistent results if not implemented carefully. Additionally, it does not prevent deadlocks which could cause transaction to hang.

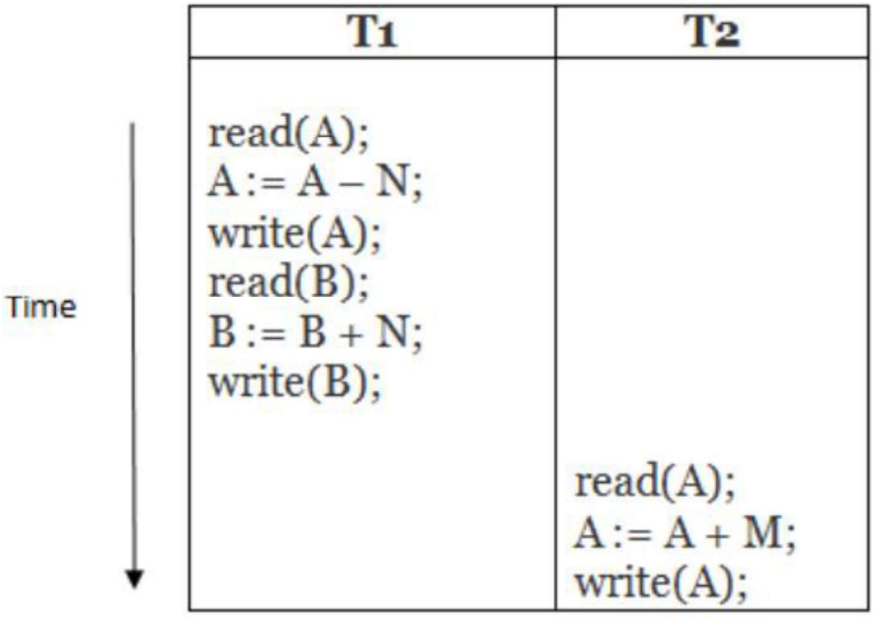
j) Describe how serializability is related to conflict serializability.

Ans. The serial schedule is a type of schedule where one transaction is executed completely before starting another transaction. In the serial schedule, when the first transaction completes its cycle, then the next transaction is executed.

For example: Suppose there are two transactions T1 and T2 which have some operations. If it has no interleaving of operations, then there are the following two possible outcomes:

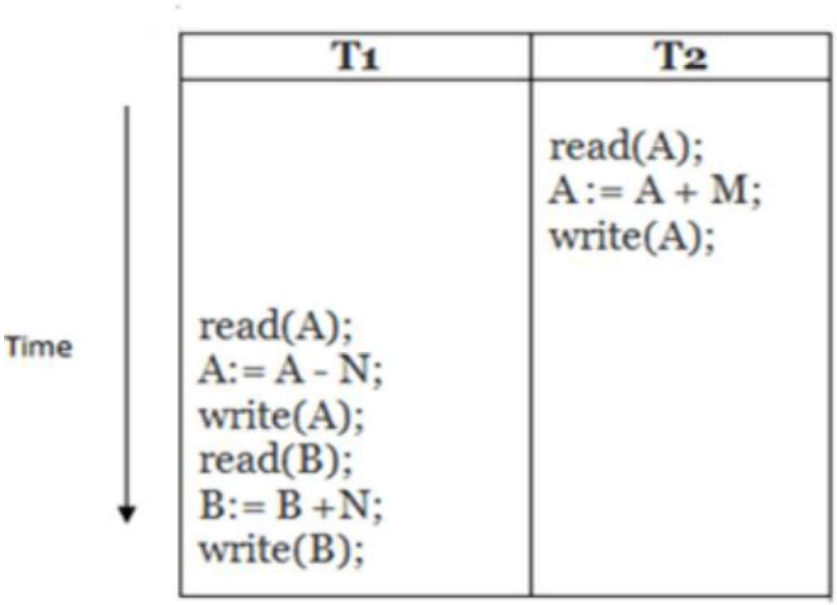
1. Execute all the operations of T1 which was followed by all the operations of T2.

2. Execute all the operations of T1 which was followed by all the operations of T2.



Schedule A

Schedule A shows the serial schedule where T1 followed by T2



Schedule B

Schedule B shows the serial schedule where T2 followed by T1.

Conflict Serializable Schedule

A schedule is called conflict serializability if after swapping of non-conflicting operations, it can transform into a serial schedule. The schedule will be a conflict serializable if it is conflict equivalent to a serial schedule.

Conflicting Operations

The two operations become conflicting if all conditions satisfy:

1. Both belong to separate transactions.

2. They have the same data item.

3. They contain at least one write operation.

Example: Swapping is possible only if S1 and S2 are logically equal

1. T1: Read(A) T2: Read(A)

T1	T2
Read(A)	
	Read(A)

Swapped

T1	T2
	Read(A)
Read(A)	

Schedule S1 **Schedule S2**

Here, S1 = S2. That means it is non-conflict.

2. T1: Read(A) T2: Write(A)

T1	T2
Read(A)	
	Write(A)

Swapped

T1	T2
	Write(A)
Read(A)	

Schedule S1 **Schedule S2**

Here, S1 ≠ S2. That means it is conflict.

SECTION B

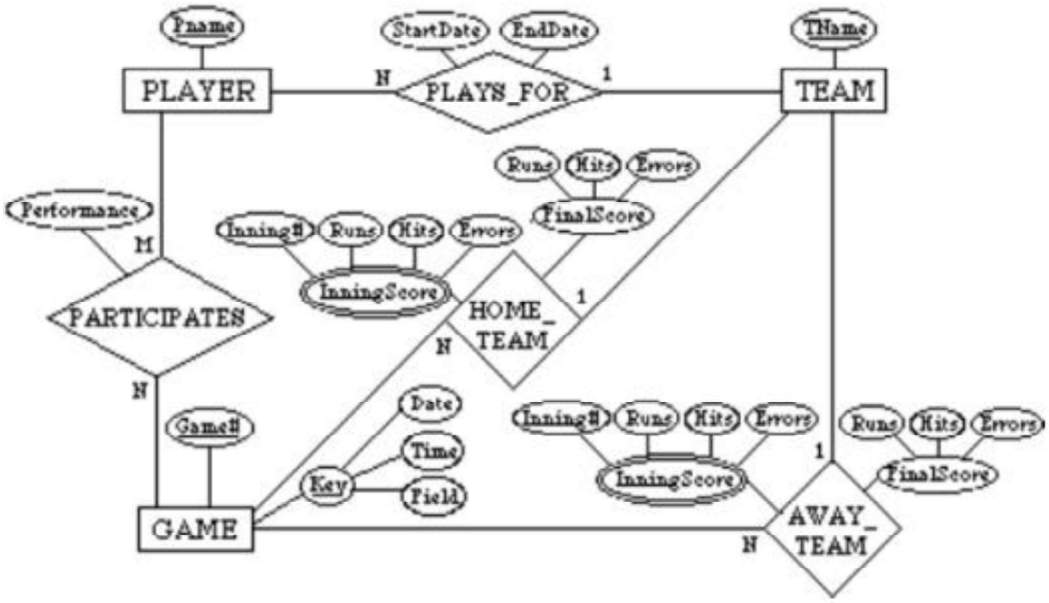
2. Attempt any three of the following :

(a) A database is being constructed to keep track of the teams and games of a sport league. A team has a number of players, not all of whom participate in each game. It is desired to keep track of players participating in each game for each team, the positions they play in that game and the result of the game.

(i) Design an E-R schema diagram for this application.

(ii) Map the E-R diagram into relational model.

Ans : E-R Diagram can be this :



PLAYER

Pname	StartDate	EndDate	TName

TEAM

TName

Inning_Score

Inning#	Runs	Hits	Errors

Final_Score

Runs	Hits	Errors

PARTICIPATES

Pname	Performance	Game#

GAME

Game#	Key	Home_Inn_score	Away_Inn_score	Home_final	Away_final

Key

Date	Time	Field

(b) What are Joins? Discuss all types of Joins with the help of suitable examples.

Ans : A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

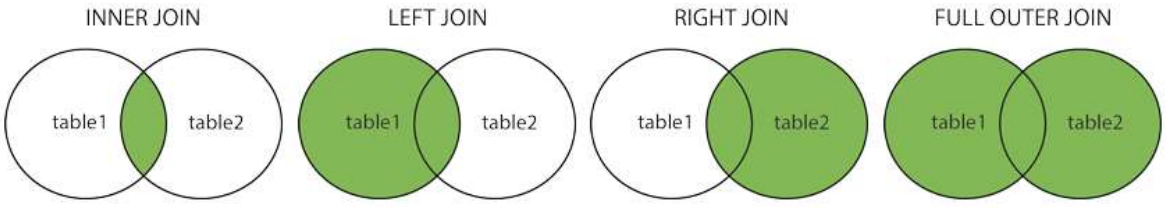
OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column.
Then, we join them, that selects records that have matching values in both tables:

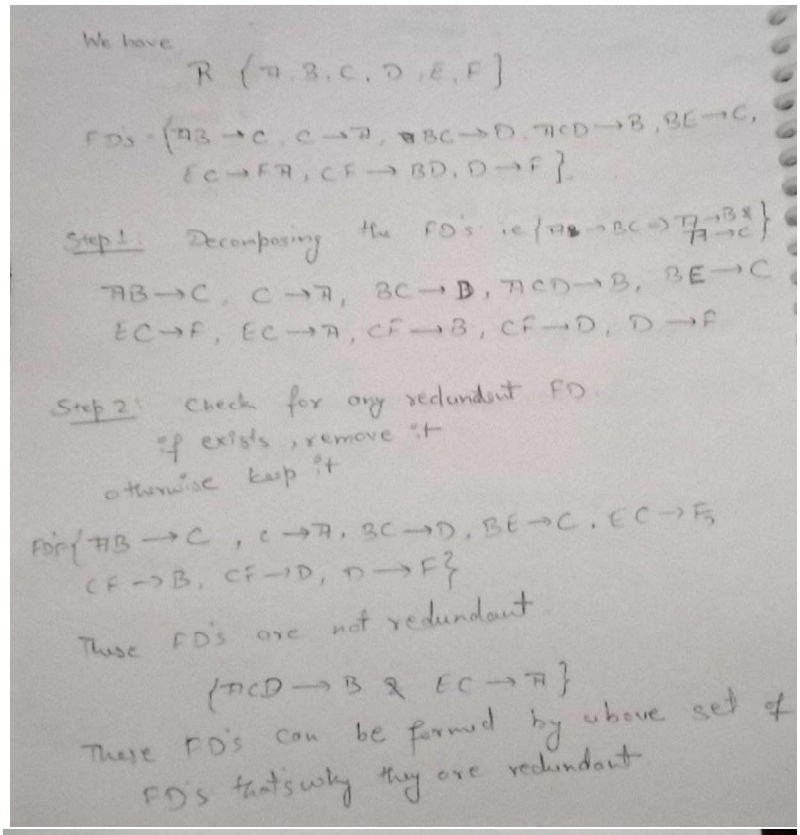
OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglunds snabbköp	8/12/1996

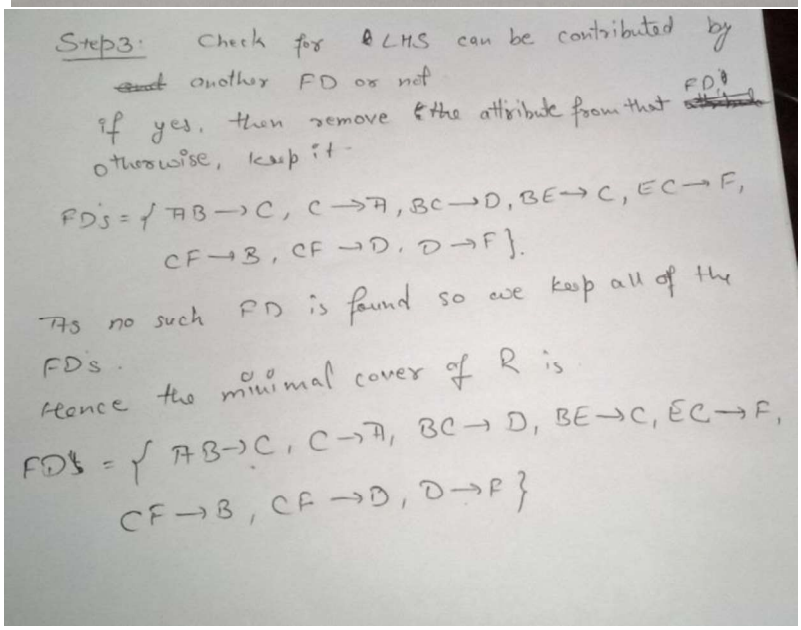
Different Types of SQL JOINs
Here are the different types of the JOINs in SQL:
(INNER) JOIN: Returns records that have matching values in both tables
LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table



(c) A set of FDs for the relation R {A, B, C, D, E, F} is AB → C, C → A, BC → D, ACD → B, BE → C, EC → FA, CF → BD, D → E. Find a minimum cover forth is set of FDs.

Ans :





(d) What is a schedule? Define the concepts of recoverable, cascade less and strict schedules, and compare them in terms of their recoverability.

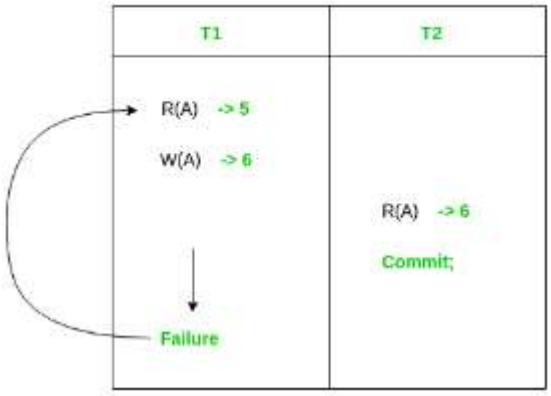
Ans : A series of operation from one transaction to another transaction is known as schedule.
It is used to preserve the order of the operation in each of the individual transaction.
Schedules are of three types :

1. Serial Schedules : The serial schedule is a type of schedule where one transaction is executed completely before starting another transaction. In the serial schedule, when the first transaction completes its cycle, then the next transaction is executed.

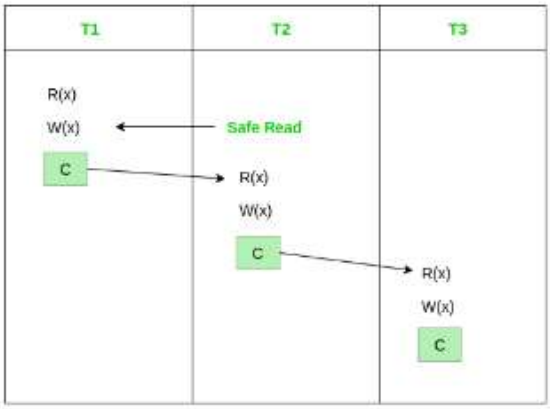
2. Non-Serial Schedules : If interleaving of operations is allowed, then there will be non-serial schedule. It contains many possible orders in which the system can execute the individual operations of the transactions.
3. Serializable Schedules : The serializability of schedules is used to find non-serial schedules that allow the transaction to execute concurrently without interfering with one another. It identifies which schedules are correct when executions of the transaction have interleaving of their operations A non-serial schedule will be serializable if its result is equal to the result of its transactions executed serially.

Recoverability of Schedules

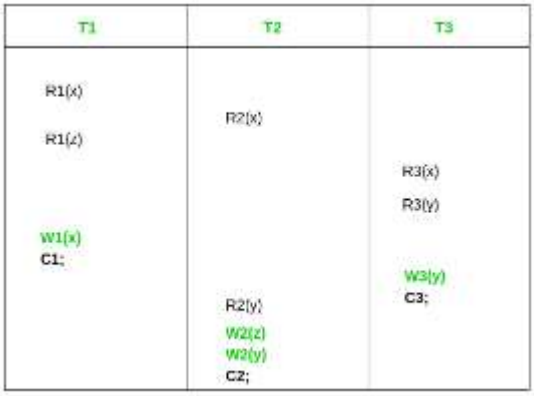
Recoverable Schedule : A schedule is said to be recoverable if it is recoverable as name suggest. Only reads are allowed before write operation on same data. Only reads (Ti->Tj) is permissible.



Cascadeless Schedule : When no read or write-write occurs before execution of transaction then corresponding schedule is called cascadeless schedule.



Strict Schedule : if schedule contains no read or write before commit then it is known as strict schedule. Strict schedule is strict in nature.



(e) Discuss the immediate update recovery techniques in both single-user and multiuser environments. What are the advantages and disadvantages of immediate update?

Ans : If a failure occurs in a single-user system, the executing transaction at the time of failure may have recorded some changes in the database. The effect of all such operations must be undone as part of the recovery process. Hence, the recovery algorithm needs an UNDO procedure, to undo the effect of certain write operations that have been applied to the database following examination of their system log entry.

Immediate update in a multi-user environment-When concurrency execution is permitted, the recovery process again depends on the protocols used for concurrency control. The procedure below outlines a recovery technique for concurrent transactions with immediate update.

Advantage of immediate update is that the changes are applied directly to the database. The disadvantage of this method is that there are frequent I/O operations while the transaction is active. In a single-user enviroment there si no concurrency control is required but log is maintained under WAL.

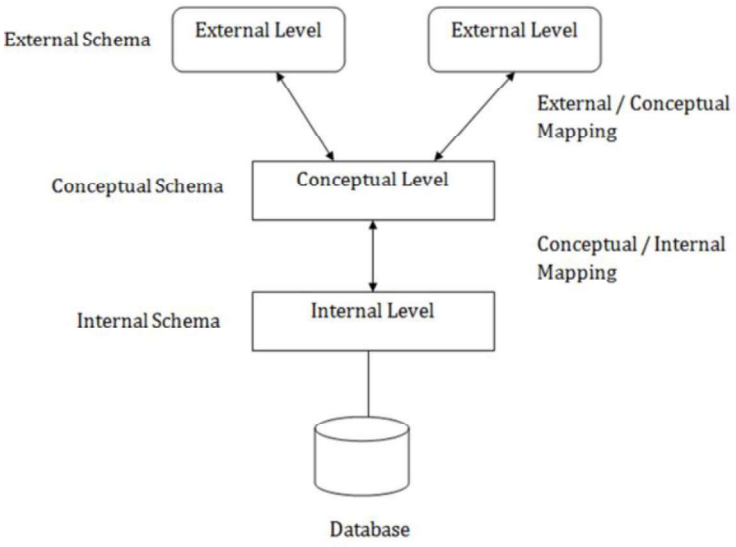
SECTION C

3. Attempt any one part of the following:

a) Describe the three-schema architecture. Why do we need mappings between schema levels? How do different definition languages support this architecture?

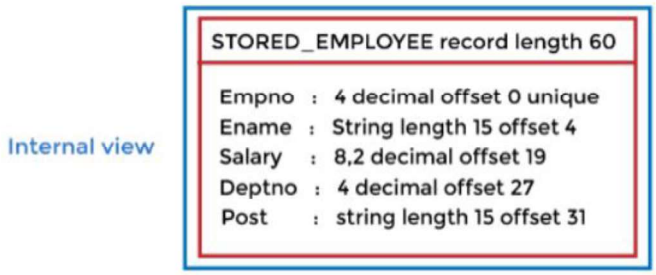
Ans. The three schema architecture is also used to separate the user applications and physical database. The three schema architecture contains three levels. It breaks the database down into three different categories.

The three-schema architecture is as follows:



Internal level:

It has an internal schema which describes the physical storage structure of the database. The internal schema is also known as a physical schema. It uses the physical data model. It is used to define that how the data will be stored in a block. The physical level is used to describe complex low-level data structures in detail.

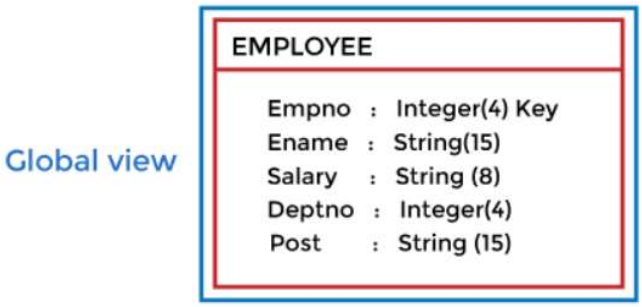


The internal level is generally concerned with the following activities:

i) Storage space allocation
ii) Access paths (eg. Specification of primary and secondary keys, indexes, pointers and sequencing)
iii) Data compression and encryption techniques
iv) Representation of stored fields

Conceptual level:

It describes the design of a database at the conceptual level. The conceptual level is also known as the logical level. The conceptual schema describes the structure of the whole database. The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data. At the conceptual level, internal details such as an implementation of the data structure are hidden. Programmers and database administrators work at this level.



External level:

At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database. An external schema is also known as view schema. Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group. The view schema describes the end user interaction with database systems.

Mapping between Views

The three levels of DBMS architecture don't exist independently of each other. There must be correspondence between the three levels i.e. how they actually correspond with each other. DBMS is responsible for correspondence between the three types of schema. This correspondence is called Mapping.

There are basically two types of mapping in the database architecture:

- Conceptual/ Internal Mapping
- External / Conceptual Mapping

Conceptual/ Internal Mapping

The Conceptual/ Internal Mapping lies between the conceptual level and the internal level. Its role is to define the correspondence between the records and fields of the conceptual level and files and data structures of the internal level.

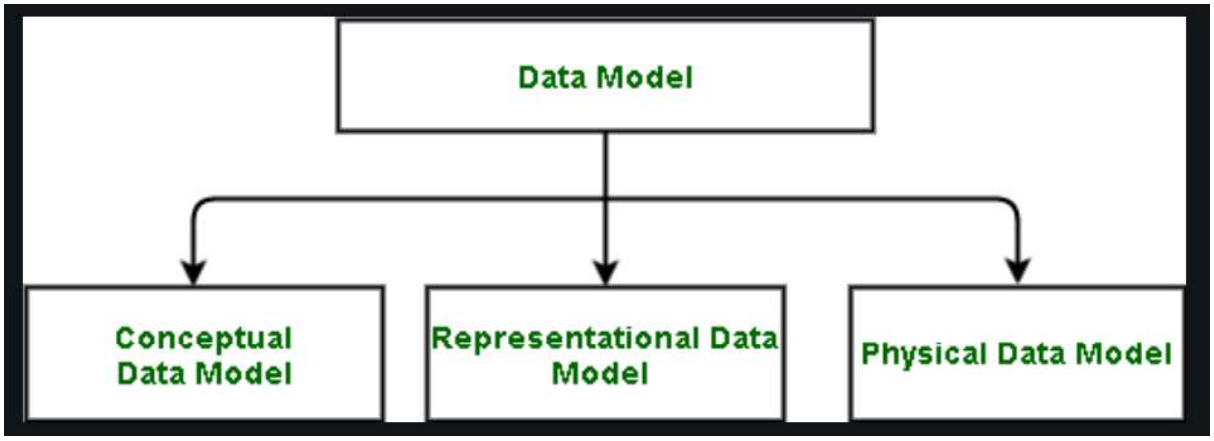
External/ Conceptual Mapping

The external/Conceptual Mapping lies between the external level and the Conceptual level. Its role is to define the correspondence between a particular external and the conceptual view.

b) What are the different types of data models in DBMS? Explain them.

Ans. Data Models

Data Model is the modeling of the data description, data semantics, and consistency constraints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction.



Conceptual Data Model:

The conceptual data model describes the database at a very high level and is useful to understand the needs or requirements of the database. It is this model, that is used in the requirement-gathering process i.e. before the Database Designers start making a particular database. One such popular model is the entity/relationship model (ER model). The E/R model specializes in entities, relationships, and even attributes that are used by the database designers. In terms of this concept, a discussion can be made even with non-computer science(non-technical) users and stakeholders, and their requirements can be understood.

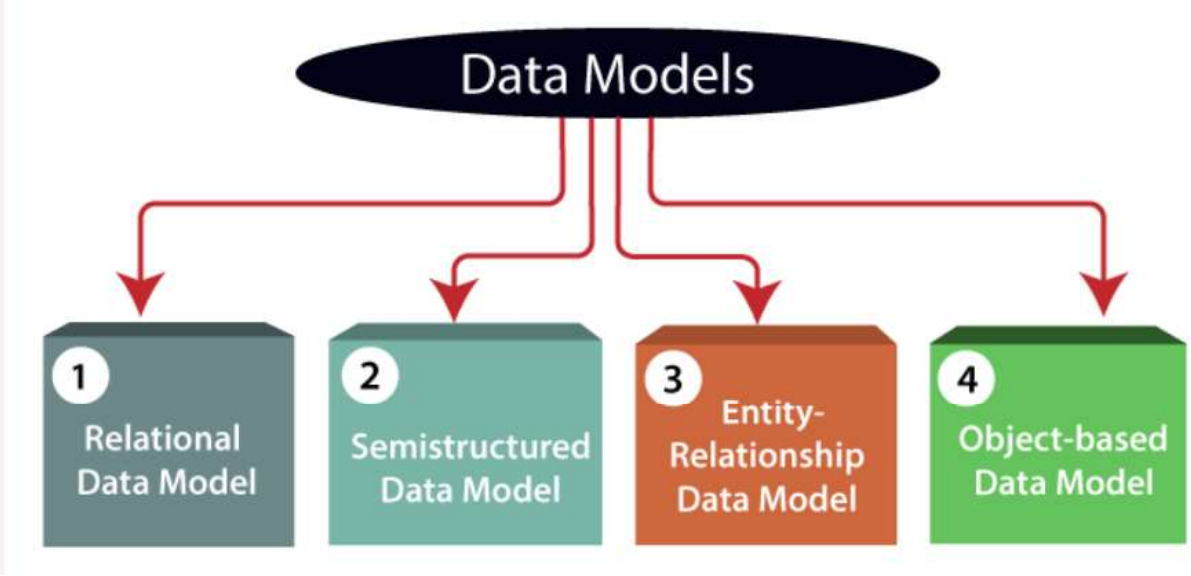
Representational Data Model:

This type of data model is used to represent only the logical part of the database and does not represent the physical structure of the database. The representational data model allows us to focus primarily, on the design part of the database. A popular representational model is Relational model.

Physical Data Model:

Ultimately, all data in a database is stored physically on a secondary storage device such as discs and tapes. This is stored in the form of files, records and certain other data structures. It has all the information of the format in which the files are present and the structure of the databases, presence

of external data structures and their relation to each other.



1) Relational Data Model**:** This type of model designs the data in the form of rows and columns within a table. Thus, a relational model uses tables for representing data and in-between relationships. Tables are also called relations. This model was initially described by Edgar F. Codd, in 1969. The relational data model is the widely used model which is primarily used by commercial data processing applications.

2) Entity-Relationship Data Model**:** An ER model is the logical representation of data as objects and relationships among them. These objects are known as entities, and a relationship is an association among these entities. This model was designed by Peter Chen and published in 1976 papers. It was widely used in database design. A set of attributes describe the entities. For example, student_name, and student_id describe the 'student' entity. A set of the same type of entities is known as an 'Entity set', and the set of the same type of relationships is known as a relationship set.

3) Object-based Data Model**:** An extension of the ER model with notions of functions, encapsulation, and object identity, as well. This model supports a rich type system that includes structured and collection types. Thus, in the 1980s, various database systems following the object-oriented approach were developed. Here, the objects are nothing but the data carrying their properties.

4) Semistructured Data Model**:** This type of data model is different from the other three data models (explained above). The semistructured data model allows the data specifications at places where individual data items of the same type may have different attribute sets. The Extensible Markup Language, also known as XML, is widely used for representing semistructured data. Although XML was initially designed for including the markup information in the text document, it gains importance because of its application in the exchange of data.

4. Attempt any one part of the following:

(a) Consider the following schema for institute library:
Student (RollNo, Name, Father_Name, Branch)
Book (ISBN, Title, Author, Publisher)
Issue (RollNo, ISBN, Date-of-Issue)

Write the following queries in SQL and relational algebra:

(i) List roll number and name of all students of the branch 'CSE'.

(ii) Find the name of student who has issued a book published by 'ABC' publisher.

(iii) List title of all books and their authors issued to a student 'RAM'.

(iv) List title of all books issued on or before December 1, 2020.

(v) List all books published by publisher 'ABC'

Ans :

(i) Select RollNo, Name from Student where branch = 'CSE';

(ii) Select S.Name from Student S where RollNo = (Select RollNo from Issue I joins Book B on I.ISBN = B.ISBN where Publisher = 'ABC') ;

(iii) Select Title, Author from Book where ISBN = (select I.ISBN from Issue I joins Student S on I.RollNo = S.RollNo where S.Name = 'RAM') ;

(iv) Select Title from Book where ISBN = (select I.ISBN from Issue I where I.Date-of-Issue <= '01-12-2020') ;

(v) Select Title from Book where publisher = 'ABC';

(b) Explain different types of Triggers in SQL/PL SQL.

Ans : A stored program that is fired by default or by some events is called a trigger.

Types Of Triggers In PL/SQL

1) Categorization on the trigger level.

- **ROW Level trigger**: It gets executed for each record that got updated by a DML statement.
- **STATEMENT Level trigger**: It gets executed only once by the event statement.

2) Categorization of the trigger timing.

- **BEFORE trigger**: It gets executed prior to the specific event that has taken place.
- **AFTER trigger**: It gets executed post the specific event that has taken place.
- **INSTEAD OF trigger**: It is a special type of trigger and it gets executed for each record that got updated by a DML statement.

3) Categorization of the trigger event.

- **DML trigger**: It gets executed if a DML event like an UPDATE, INSERT, or DELETE is performed.
- **DDL trigger**: It gets executed if a DDL event like a DROP, ALTER, or CREATE is performed.
- **DATABASE trigger**: It gets executed if a database event like SHUTDOWN, STARTUP, LOGOFF, and LOGON has taken place.

5. Attempt any one part of the following:

(a) Given the following set of FD's on Schema R (V, W, X, Y, Z) {Z→V, W→Y, XY→Z, V→WX} State whether the following decomposition are loss-less-decompositions or not.

(i) R1 = (V, W, X), R2 = (V, Y, Z)

(ii) R1 = (V, W, X), R2 = (X, Y, Z)

Ans : To determine if a decomposition is lossless join or not, we need to check if the original relation can be reconstructed from the decomposed relations using a natural join operation.

For the given decomposition:
R1 = (V, W, X) R2 = (V, Y, Z)
To check if this is a lossless join decomposition, we need to check if the join of R1 and R2 using the common attribute V produces the original relation R.
(R1 ⋈ R2) = (V, W, X) ⋈ (V, Y, Z)
Using the common attribute V to join R1 and R2, we get:
V | W | X | Y | Z --+---+---+---+-- | | | |
Since there are no tuples in the resulting relation, we cannot reconstruct the original relation R. Hence, this is not a lossless join decomposition.

(b) Consider the universal relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies F = {{A, B}→{C}, {A}→{D, E], {B}→{F}, {F}→{G, H], {D}→{I, J}}. What is the key for R? Decompose R into 2NF and then 3NF relations.

Ans : To find the key for R, we need to determine the attributes that functionally determine all other attributes in the relation. From the given set of functional dependencies, we can see that:

- A and B together functionally determine C.
- A alone functionally determines D and E.
- B alone functionally determines F.
- F alone functionally determines G and H.
- D alone functionally determines I and J.

To combine all of these functional dependencies and find the key, we can start with the set of all attributes and eliminate any attributes that can be functionally determined by the others:

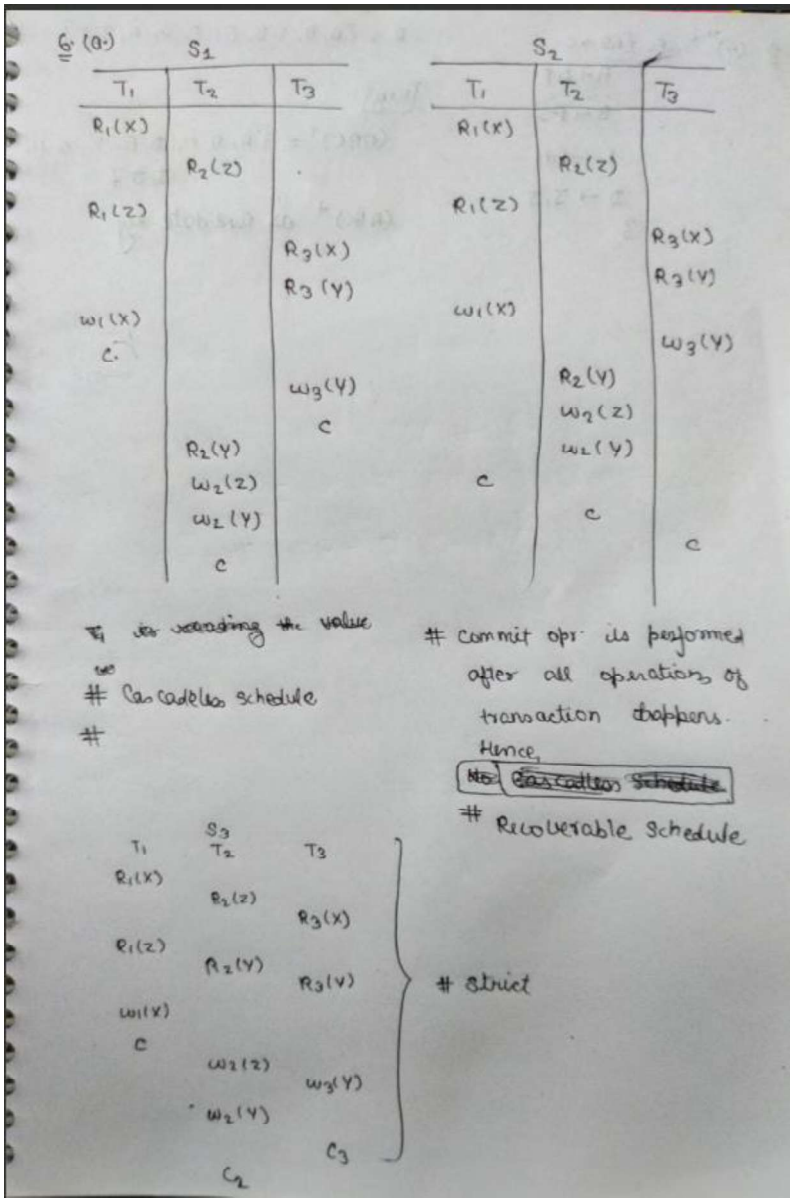
- AB determines C, so we can eliminate C from the set of attributes.
- A determines D and E, so we can eliminate both D and E from the set of attributes.
- B determines F, so we can eliminate F from the set of attributes.
- F determines G and H, so we can eliminate both G and H from the set of attributes.
- D determines I and J, so we can eliminate both I and J from the set of attributes.

The remaining attributes are A and B, so the key for R is {A, B}.
To decompose R into 2NF, we need to identify any partial dependencies and create separate relations for them. From the given set of functional dependencies, we can see that A and B together determine C, which means that C depends on a part of the key. To remove this partial dependency, we can create a new relation that includes C and the attributes that functionally determine it, which are A and B:
R1 (A, B, C)
The remaining attributes (D, E, F, G, H, I, J) all depend on the key {A, B} and can be placed in a separate relation:
R2 (A, B, D, E, F, G, H, I, J)
To decompose R into 3NF, we need to identify any transitive dependencies and create separate relations for them. From the given set of functional dependencies, we can see that F functionally determines G and H, but F is not part of the key. This means that there is a transitive dependency and we need to create a new relation for F and its functional dependencies:
R3 (F, G, H)
The remaining attributes (A, B, C, D, E, I, J) can be placed in a separate relation:
R4 (A, B, C, D, E, I, J)
This decomposition is in 3NF since all relations have no transitive dependencies and all functional dependencies are either on the key or the whole relation.

6. Attempt any one part of the following:

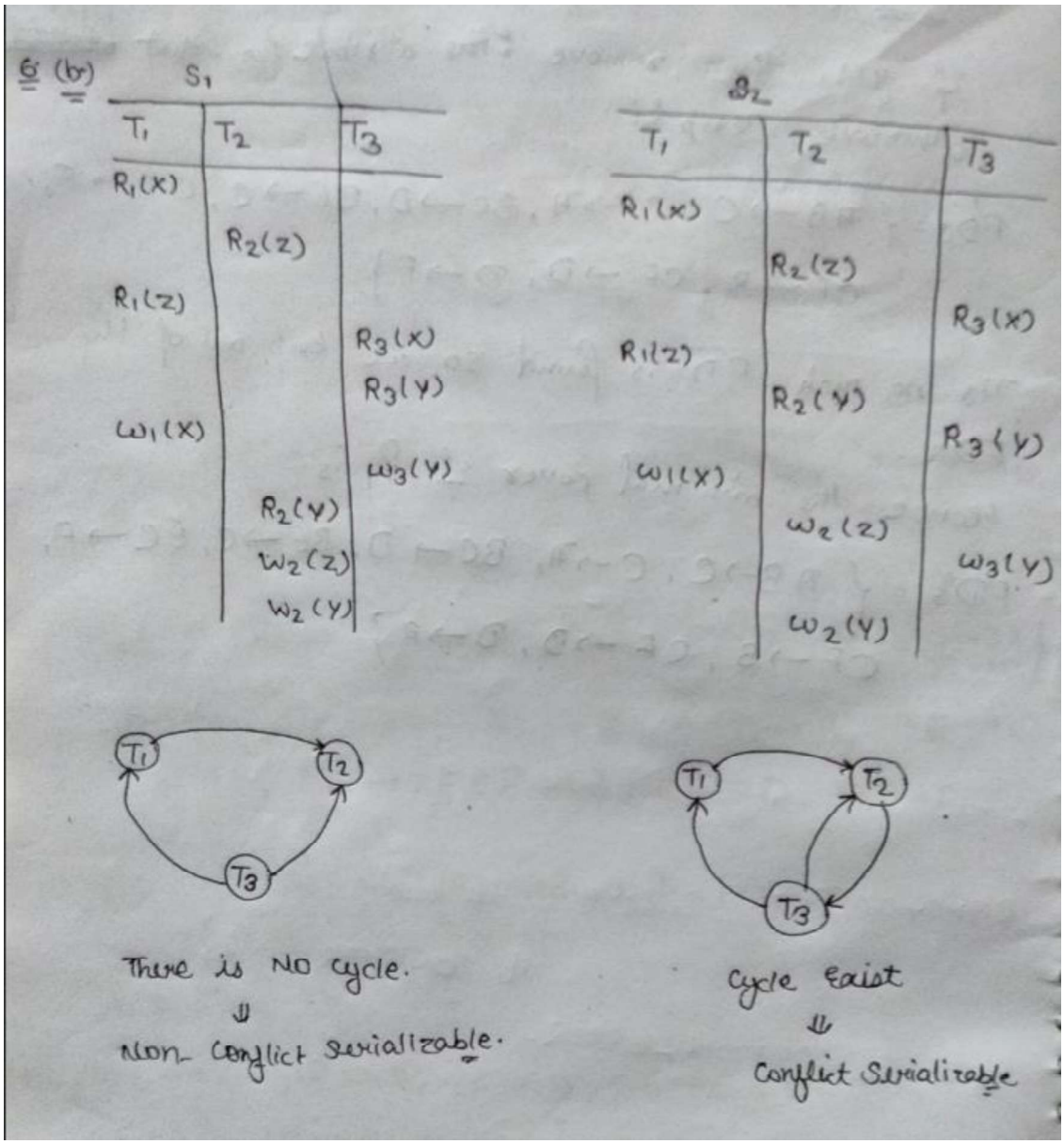
(a) Consider schedules S1, S2, and S3 below. Determine whether each schedule is strict.cascade less, recoverable, or non recoverable. (Determine the strictest recoverability condition that each schedule satisfies.)
S1 :r1(X); r2(Z); r1(Z); r3(X); r3(Y); w1(X); c1; w3(Y); c3; r2(Y); w2(Z); w2(Y); c2
S2 : r1 (X): r2 (Z), r1 (2); r3 (X); r3 (Y); w1 (X); w3 (Y); r2 (Y); w2 (Z); w2 (Y); c1; c2; c3;
S3: r1 (X); r2 (Z): 13 (X); r1 (Z); 12 (Y); 13 (Y); w1 (X); c1; w2 (Z): w3 (Y); w2 (Y); c3; c2;

Ans :



(b) Consider the three transactions T1, T2, and T3. and the schedules S1 and S2given below. State whether each schedule is serializable or not. If a schedule is serializable. write down the equivalent serial schedule(s).
T1: r1(X); r1(Z); w1(X);
T2: r2(Z); r2(Y); w2(Z); w2(Y);
T3: r3(X); r3(Y); w3(Y);
S1: r1(X); r2(Z); r1(Z); r3(X); r3(Y); w1(X); w3(Y); r2(Y); W2(Z); w2(Y);
S2: r1(X); r2(Z); r3(X); r1(Z); r2(Y); r3(Y); w1(X); w2(Z); w3(Y); w2(Y);

Ans :



7. Attempt any one part of the following:

(a) Discuss the timestamp ordering protocol for concurrency control. How does strict timestamp ordering differ from basic timestamp ordering?

Ans : Timestamp-based Protocols

The most commonly used concurrency protocol is the timestamp based protocol. This protocol uses either system time or logical counter as a timestamp.Lock-based protocols manage the order between the conflicting pairs among transactions at the time of execution, whereas timestamp-based protocols start working as soon as a transaction is created.

Every transaction has a timestamp associated with it, and the ordering is determined by the age of the transaction. A transaction created at 0002 clock time would be older than all other transactions that come after it. For example, any transaction 'y' entering the system at 0004 is two seconds younger and the priority would be given to the older one.

In addition, every data item is given the latest read and write-timestamp. This lets the system know when the last 'read and write' operation was performed on the data item.

Timestamp Ordering Protocol

The timestamp-ordering protocol ensures serializability among transactions in their conflicting read and write operations. This is the responsibility of the protocol system that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.

- The timestamp of transaction Ti is denoted as TS(Ti).
- Read time-stamp of data-item X is denoted by R-timestamp(X).
- Write time-stamp of data-item X is denoted by W-timestamp(X).

Timestamp ordering protocol works as follows −

If a transaction Ti issues a read(X) operation –

- If TS(Ti) < W-timestamp(X) : Operation rejected.
- If TS(Ti) >= W-timestamp(X) : Operation executed.
  All data-item timestamps updated.

If a transaction Ti issues a write(X) operation –

- If TS(Ti) < R-timestamp(X) : Operation rejected.
- If TS(Ti) < W-timestamp(X)
  - Operation rejected and Ti rolled back.
- Otherwise, operation executed.

Strict Timestamp Ordering (TO). A variation of basic TO called strict TO ensures that the schedules are both strict (for easy recoverability) and (conflict) serializable. In this variation, a transaction T that issues a read_item(X) or write_item(X) such that TS(T) > write_TS(X) has its read or write operation delayed until the transaction T that wrote the value of X (hence TS(T) = write_TS(X)) has committed or aborted. To implement this algorithm, it is necessary to simulate the locking of an item X that has been written by transaction T until T is either com-mitted or aborted. This algorithm does not cause deadlock, since T waits for T only if TS(T) > TS(T).

(b) How do optimistic concurrency control techniques differ from other concurrency control techniques? Why they are also called validation or certification techniques? Discuss the typical phases of an optimistic concurrency control method.

Ans : The validation scheme automatically guards against cascading rollbacks, since the actual writes take place only after the transaction issuing the write has committed. However, there is a possibility of starvation of long transactions, due to a sequence of conflicting short transactions that cause repeated restarts of the long transaction. To avoid starvation, conflicting transactions must be temporarily blocked, to enable the long transaction to finish. This validation scheme is called the optimistic concurrency control scheme since transactions execute optimistically, assuming they will be able to finish execution and validate at the end. In contrast, locking and timestamp ordering are pessimistic in that they force a wait or a rollback whenever a conflict is detected, even though there is a chance that the schedule may be conflict serializable.

The optimistic method of concurrency control is based on the assumption that conflicts of database operations are rare and that it is better to let transactions run to completion and only check for conflicts before they commit. An optimistic concurrency control method is also known as validation or certification methods. No checking is done while the transaction is executing. The optimistic method does not require locking or timestamping techniques. Instead, a transaction is executed without restrictions until it is committed. In optimistic methods, each transaction moves through the following phases:

1. Read phase. During this phase, the system executes transaction Ti. It reads the values of the various data items and stores them in variables local to Ti. It performs all write operations on temporary local variables, without updates of the actual database.

2. Validation phase. Transaction Ti performs a validation test to determine whether it can copy to the database the temporary local variables that hold the results of write operations without causing a violation of serializability.

3. Write phase. If transaction Ti succeeds in validation (step 2), then the system applies the actual updates to the database. Otherwise, the system rolls back Ti.