1. a) 19-20(1-a)

b) What is the difference between a Candidate key, a primary key & Composite key? What considerations might influence the choice of a primary key?

| Primary | Candidate | Composite (Fname, Lname) |
|---|---|---|
| i. There can be only one primary key in any table/relation | i. There can be multiple/more Then one Candidate key in a table/relation | i. Two or more attributes (Columns) that together can uniquely identify a tuple (record) in a table and work as PK. |
| ii. The primary key is unique & not null. | ii. whereas a Candidate unique & alternate keys can be null. | ii. Combination of Columns |
| iii. Only one Candidate key can be primary key | iii. Each CK can qualify as PK | iii. Creating more than one primary key is jointly known as Composite key. |
| iv. Ex: Student(ID, Name, address) PK - ID | iv. Ex: Stu(ID, name, email) CK: ID, email (minimal super keys are) | |

(✓To denote a CK

✓Chosen with care by DBA

✓Never or very rarely changed

✓CK with NULL value is NOT the primary key.

✓ PK = Unique + NOT + NULL

b) 19-20

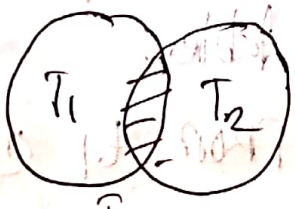
c) What are the join type and condition that are permitted in SQL?

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

Here are different type of join in SQL:

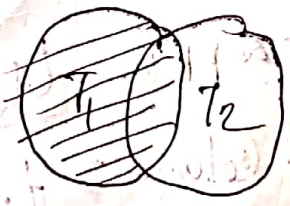
i. INNER join: Return records that have matching values in both tables

Syntax: SELECT C_name(s) FROM t1 INNER JOIN t2 ON t1.C_name = t2.C_name;



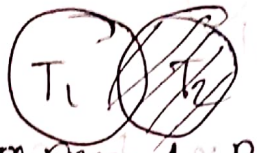
Inner

ii. Left (Outer): Returns all records from the left table, and the matched records from the right table.




Syntax: SELECT * FROM t1 LEFT JOIN t2 ON ...

✓ Right (Outer): Returns all records from the right table and the matched records from the left table




Syntax: SELECT * FROM t1 RIGHT JOIN t2 ON ...

✓ Full (Outer): Return all records when there is a match in either left or right table.


Syn: SELECT * FROM t1 FULL JOIN t2 ON ;>

✓ CROSS join: Returns the Cartesian product of two tables. ie all possible combination of rows from both tables.

Syn: SELECT * FROM t1 CROSS JOIN t2;

✓ SELF JOIN: Join a table with itself, often used when working with hierarchical data or to compare rows within the same table.

Synt: SELECT * FROM t1 INNER JOIN t2 ON t1.column = t2.column;

Natural join: Performs a join based on columns with the same name in both tables.

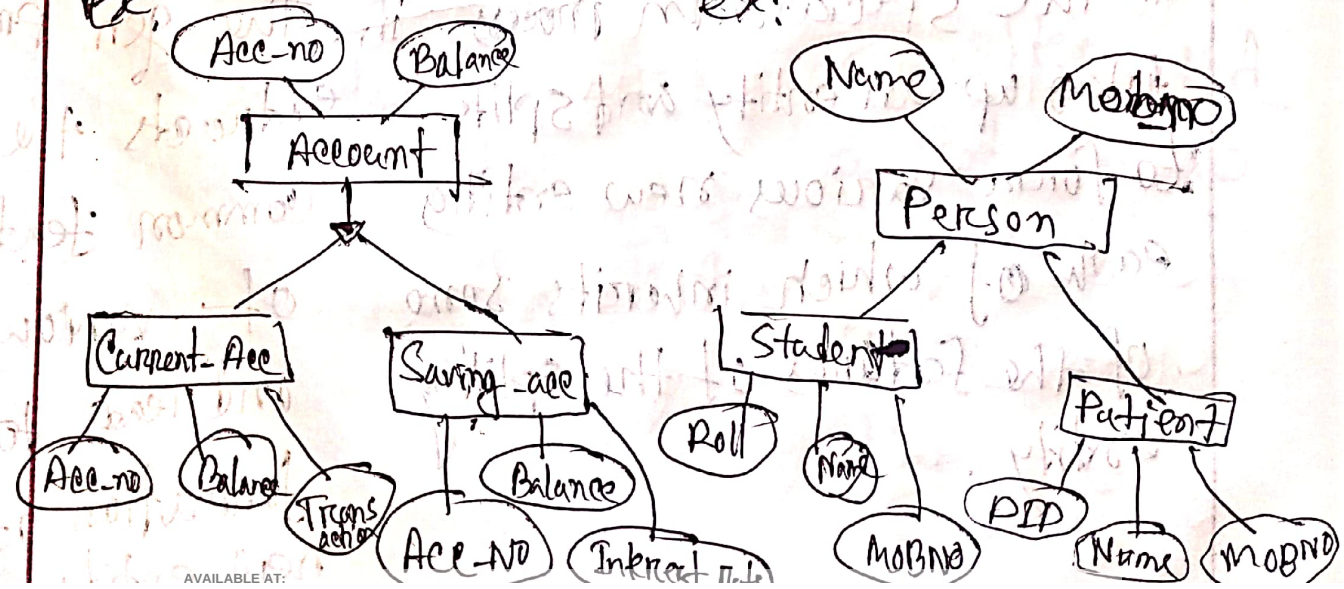
Syn: SELECT * FROM t1
NATURAL JOIN t2;

a) diff between Specialization & generalization.

| Specialization | generalization |
|---|---|
| i. This process operates in a top-down manner | i. This process operates in a bottom manner |
| ii. The entity of higher-level must always have the lower level entity (generalization) | ii. The entities of the higher level might not consist of the entities of the lower level. (specialization) |
| iii. The specialization process takes up an entity and splits to form various new entities each of which inherits some of the features of the splitting entity. | iii. The gen process extracts the most common features of various entities and leads to the formation of a new entity. |

| Specialization | Generalization |
| --- | --- |
| iv. A user can easily apply a generalization to a group of different entities | iv. We can only apply specialization to a single entity. |
| v. It leads to a reduction in the size of a schema | v. It leads to an increase in the size of a scheme |
| vi. This process leads to the formation of a single entity out of multiple entities | vi. This process leads to the formation of multiple entities from one single entity. |
| vii. There is inheritance in Specialization | vii. There is no inheritance in Generalization |

ex:



b) Same as 2019-20 (2-b)

c) Same as 2018-19 (2-b)

d) Draw the ER diagram for ...

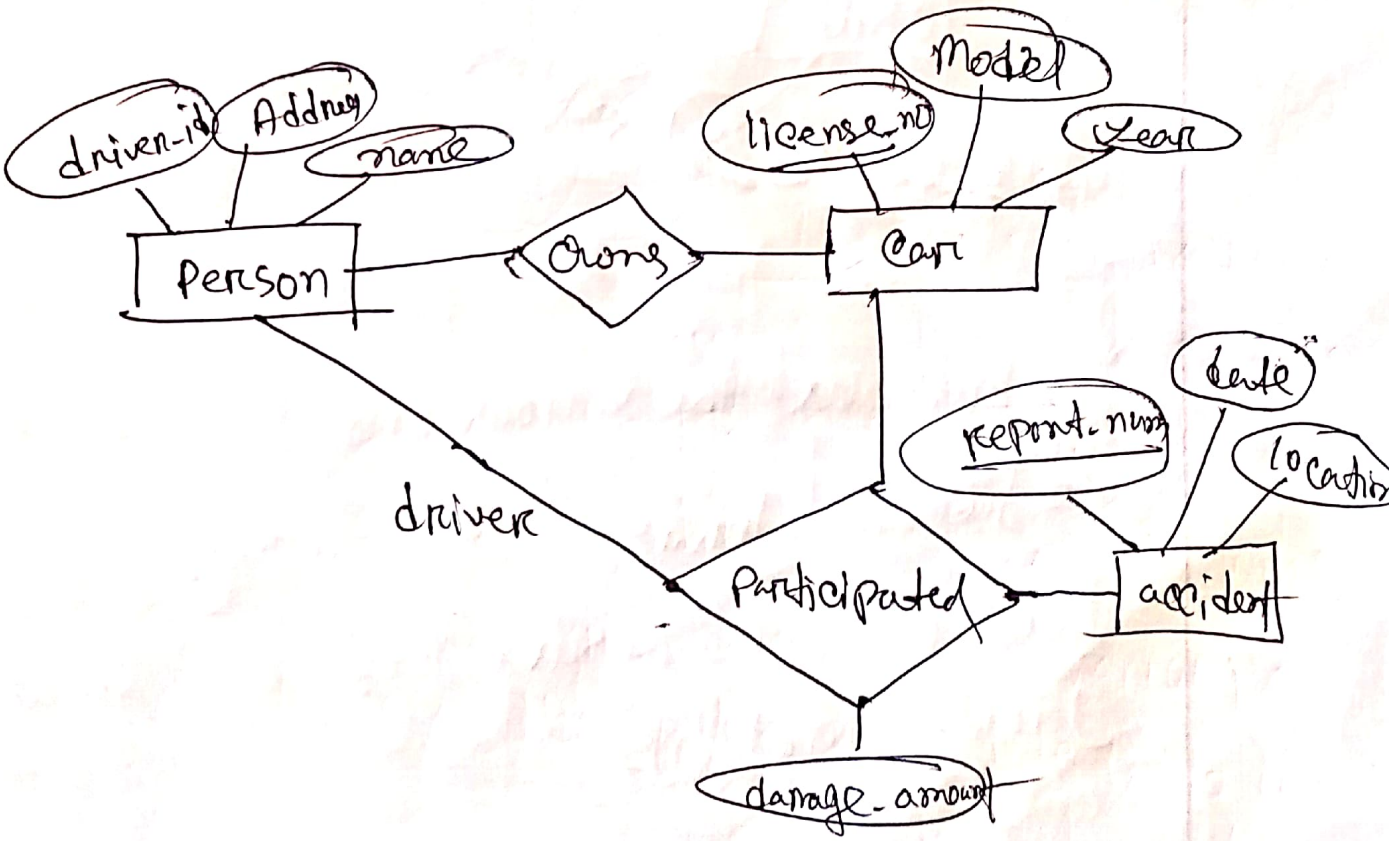
Person (driver_id, name, address)

Car (license no, year, model)

Accident (report number, date, location)

owns (driver_id, license_no)

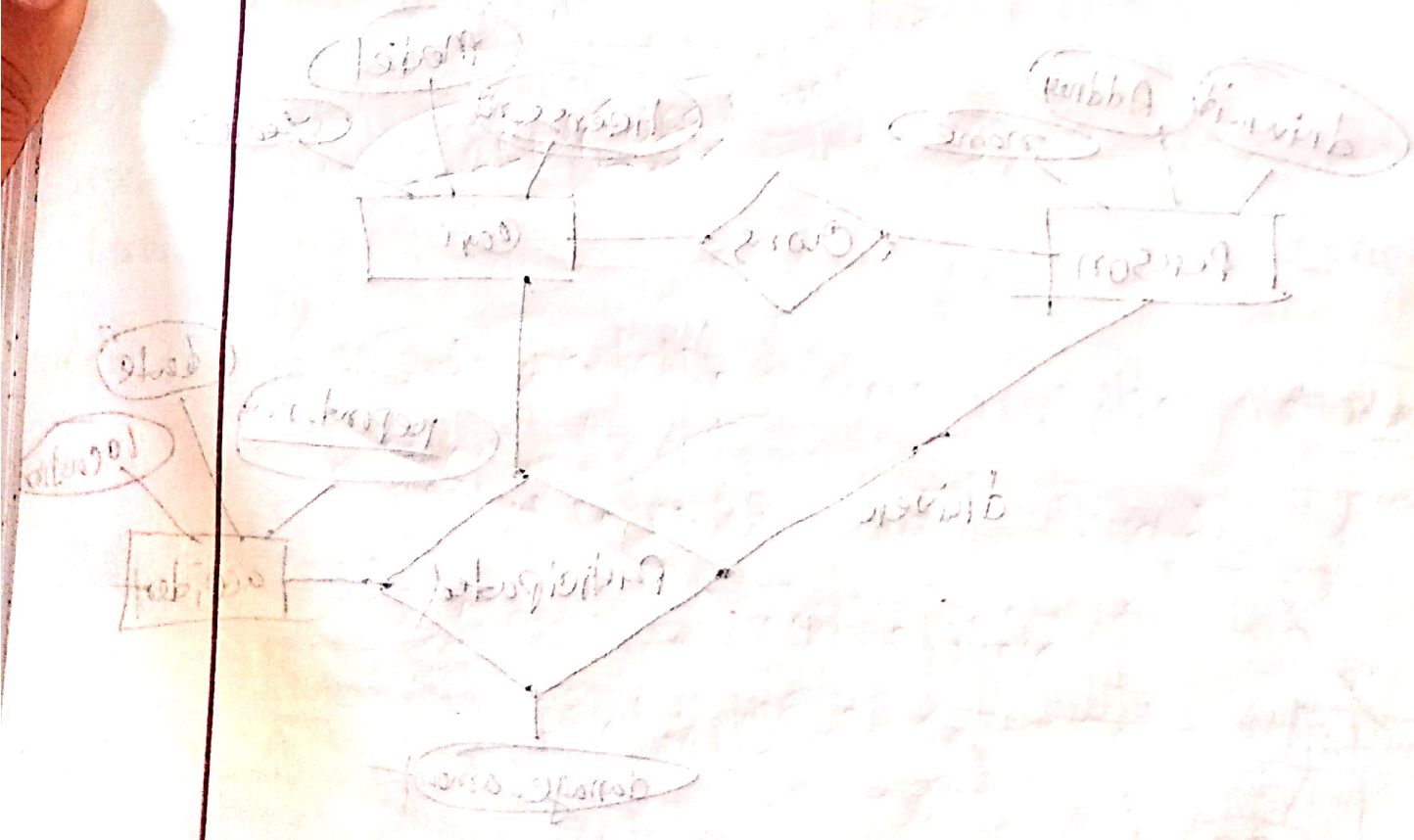
Participated (driverid, license_no, reportnum damage amount)



3. a) 19-20 (2-a)

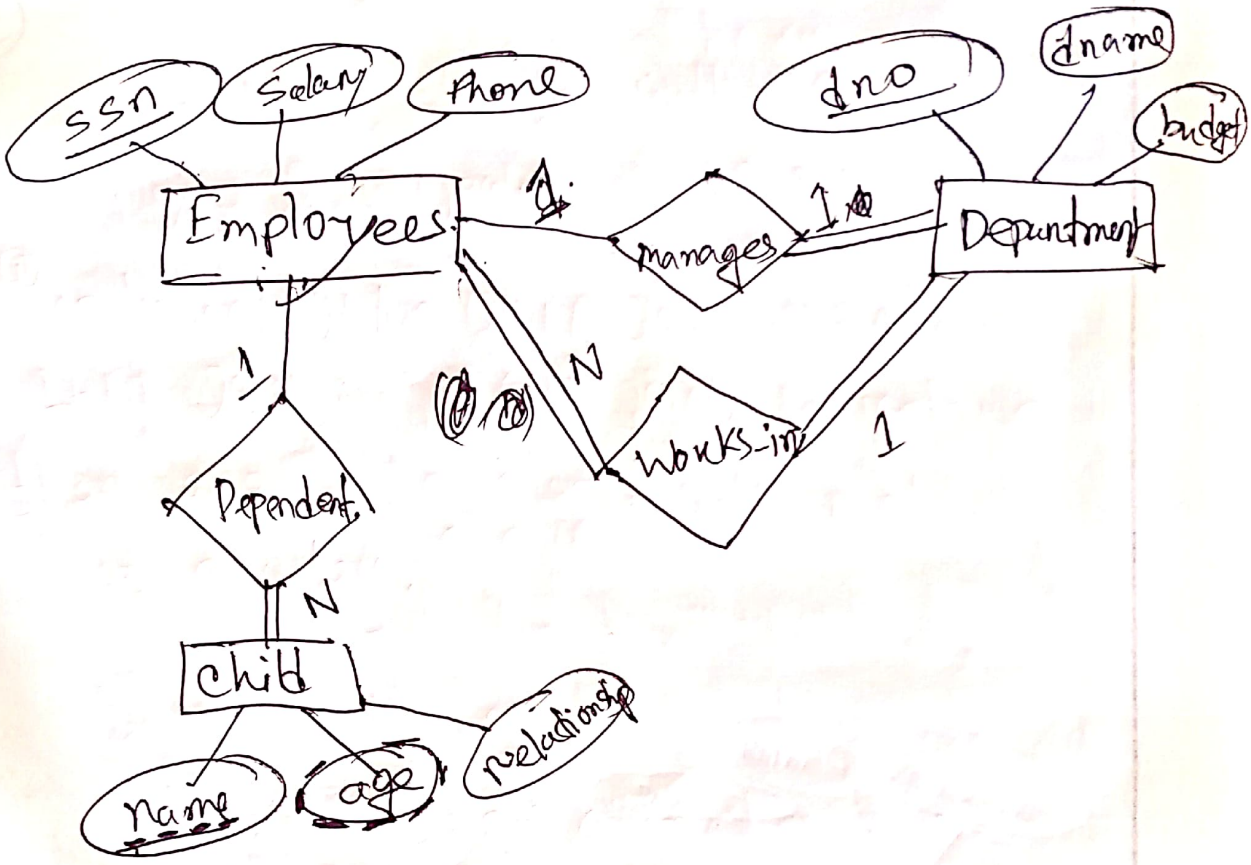
b) 19-20 (2-d)

c)



4. c) 19-20 (4-a)

b)




6. a) What are the various anomalies we may have to face in retriving data through joining relations? How can we can resolve those through various Joining operations?

Anomaly means inconsistency in the pattern from the normal form. When retrieving data through joining relations in a relational database, various anomalies may arise.

i. Update anomalies;

- Definition:— Update anomalies occurs when updating data in one relation leads to inconsistencies or unintended changes in other related relation.
- Ex - Let you have a database of employees & projects. If an employee changes their department
- Resolution - Use INNER JOIN or OUTER JOIN operations based on the specific requirement of your query.

ii. Insert Anomalies:

- Definition - Insert anomalies occur when adding new data to a table is not possible without adding data to another related table, leading to incomplete or inaccurate information.

5. a) SQL PDF

b) 2019-20 (5-b)

6. a) Eg – In a database with seprate tables for employees & projects

Resolution: Utilize OUTER JOIN operation. Such as LEFT JOIN or RIGHT JOIN, to retrive data from one table along with matching data from another.

iii. Delete Anomalies:

Definition: Delete anomalies occur when deleting data from one relation unintentionally removes related data in other tables, leading to the loss of information.

Ex: If you delete an employee record and that employees is associated with only one project in another table,

iii) Resolution: Use INNER and OUTER JOIN operations to avoid unnecessary data loss during deletion

b) Ans:)
The given relational schema for the "Department" table has problems that is each department can have multiple locations, which violates the 1st INF requirement. In INF each attribute in a relations must be atomic (can't have multiple values).

Department:

| dept number | dept name | dept manager no | dept location |
|---|---|---|---|
| 101 | CSE | 001 | Barishal |
| 102 | MATH | 002 | Dhaka |
| 103 | ENG | 003 | Raj, Khulna, Dh |
| | | | Khu |

To achive INF for such a relation, need to restructure the schema by creating a separate table for dept. locations.

Department:

| dept number | dept name | dept manager no |
|---|---|---|
| 101 | CSE | 001 |
| 102 | MATH | 002 |
| 103 | ENG | 003 |

Department_location:

| dept_number | dept location |
|---|---|
| 101 | Barishal |
| 101 | Khulna |
| 102 | Dhaka |
| 102 | Khulna |
| 102 | Rajshahi |
| 103 | Khulna |

In this modified schema,

i. The "Department" table now contains only attributes that are atomic

ii. The "Dept_location" table is introduced to handle the multiple locations associated with each dept. Composite PK — dept number & dept location. for this table

(c) FDS hold in the relation R?

| A | B | C | D | E |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 1 | 4 | 3 | 4 | 5 |
| 1 | 2 | 4 | 4 | 1 |

i. $AB \rightarrow C$

ii. $B \rightarrow D$

iii. $DE \rightarrow A$.

Consider R(A, B) with the following instance of R. on this instance $A \to B$ holds $B \to A$ does not holds.

(i) $AB \to C$

suppose $AB = A$ & $C = B$

$t_1[A] = 1$ 2 $t_2[A] = 1$ 4 $t_3[A] = 1$ 2

$t_1[B] = 3$ $t_2[B] = 3$ $t_3[B] = 4$

for tuple 3 $B \to A$ does not holds

so, $AB \to C$ is not possible

(ii) $B \to D$

$t_1[A] = 2$ $t_2[A] = 4$ $t_3[A] = 2$

$t_1[B] = 4$ $t_2[B] = 4$ $t_3[B] = 4$

$B \to A$ is hold but $A \to B$ does not hold

So, $B \to D$ is possible on holds in R

(iii) $DE \rightarrow A$

$t_1[A] = 9,5$ $t_2[A] = 9,5$ $t_3[A] = 9,1$

$t_1[B] = 1$ $t_2[B] = 1$ $t_3[B] = 1$

So, $DE \rightarrow A$ holds in R.

d) FDS: $\{AB \rightarrow CD, BC \rightarrow D\}$. Find minimal Cover or Irreducible set.

Soln:

Break down the following FDs:

* $AB \rightarrow C$

  $AB \rightarrow D$

  $BC \rightarrow D$

Using these dependencies:

i. $AB^+ = \{A, B, C, D\}$

$AB^+ = \{A, B, D\}$ [without $AB \rightarrow C$]

So, the closure is changed

So, $AB \rightarrow C$ is redundant.

ii. $AB^+ \Rightarrow \{A, B, D\}$

$AB^+ = \{A, B, C\}$ [without $AB \rightarrow D$]

So, $AB \rightarrow D$ is essential

iii. $BC^+ = \{B, C, D\}$

$BC^+ = \{B, C\}$ [without $BC \rightarrow D$]

So, $BC \rightarrow D$ is essential

So, Canonical Cover

$AB \rightarrow C$

$A \rightarrow D$ — Union

$BC \rightarrow D$

Transitivity rule

$AB \rightarrow CD$

7. a) Why should NULL in a relation be avoided as possible? Explain

Ans. Avoiding the use of NULL values in a relation is a good practice in DB design for several reasons:

i. Semantic Ambiguity: Null values can introduce semantic ambiguity, as they may represent different things in different contexts. This lack of clarity can lead to misunderstanding and errors in data interpretation.

ii. Complexity in Quering: Handling NULL values in queries additional logic, such as using the 'IS NULL' or 'IS NOT NULL' conditions. This complexity makes queries more difficult to read, write & maintain.

iii. Aggregation challenges: NULL values can cause challenges when aggregating data. Operands like SUM, AVG, COUNT etc. may not provide accurate results when NULL values involved.

iv. Joining & Comparison issues: Null values ~~can cause challenges~~ Complicate join

operations & Comparisons.

v. Indexing & Searching: Indexing on columns with NULL values may not be as efficient as indexing on columns with non-NULL values.

vi. Data integrity: Null values can make it challenging to enforce data integrity constraints

vii. Compatibility Issues: Different DB system may handle NULL values differently.

viii. Three-Valued logic: SQL uses a three valued logic system that includes - True, False & Unknown (representing NULL).

b) i) lets analyze the given transactions:

$T_1$: R(A); R(B); if A=0 then B:=B+1; W(B)

$T_2$: R(B); R(A); if B=0, then A:=A+1; W(A);

The consistency requirements is A=0 ∨ B=0 with the initial values A=0 & B=0.

7. a) There are two possible executions:

$T_1 T_2$ & $T_2 T_1$

Case 1:

| | A | B |
|---|---|---|
| initially | 0 | 6 |
| after $T_1$ | 0 | 1 |
| after $T_2$ | 0 | 1 |

Consistency met: $A=0 \vee B=0 \equiv T \vee F \equiv T$

Case 2:

| | A | B |
|---|---|---|
| initially | 0 | 0 |
| after $T_2$ | 1 | 0 |
| after $T_1$ | 1 | 0 |

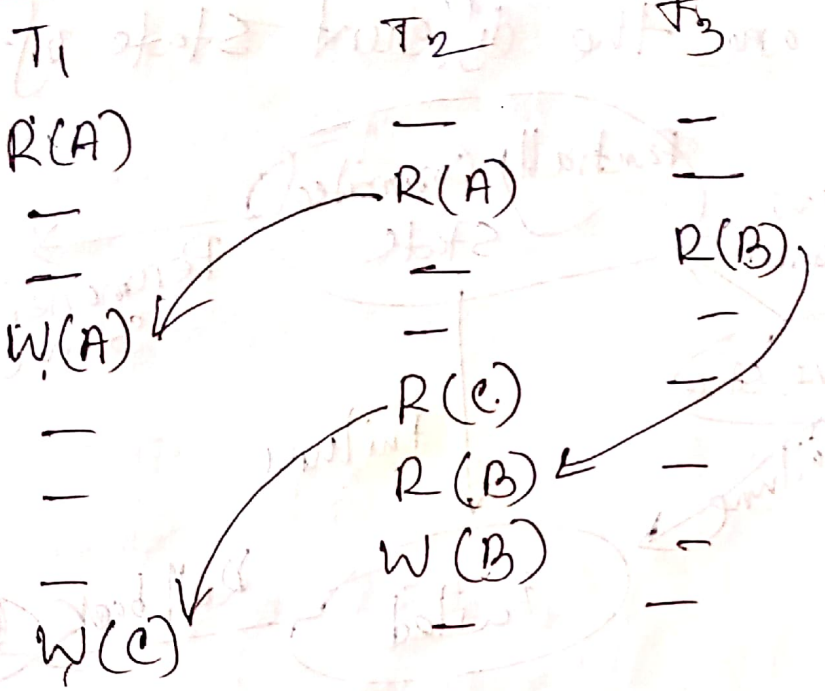
Consistency met:

$A=0 \vee B=0 \equiv F \vee T \equiv T$.

c) Check whether the schedule is conflict Serializable or not?

S: R1(A); R2(A); R3(B); W1(A); R2(C); R2(B; W2(B); W1(C).

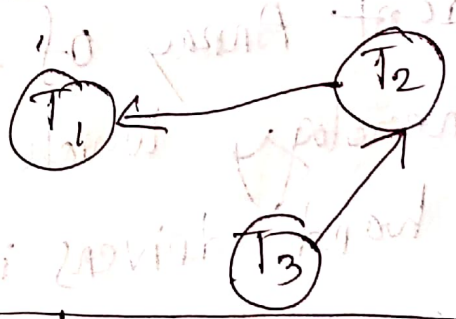
Schedule table:




$T_2$ conflict with $T_1$

$T_3$ " $T_2$

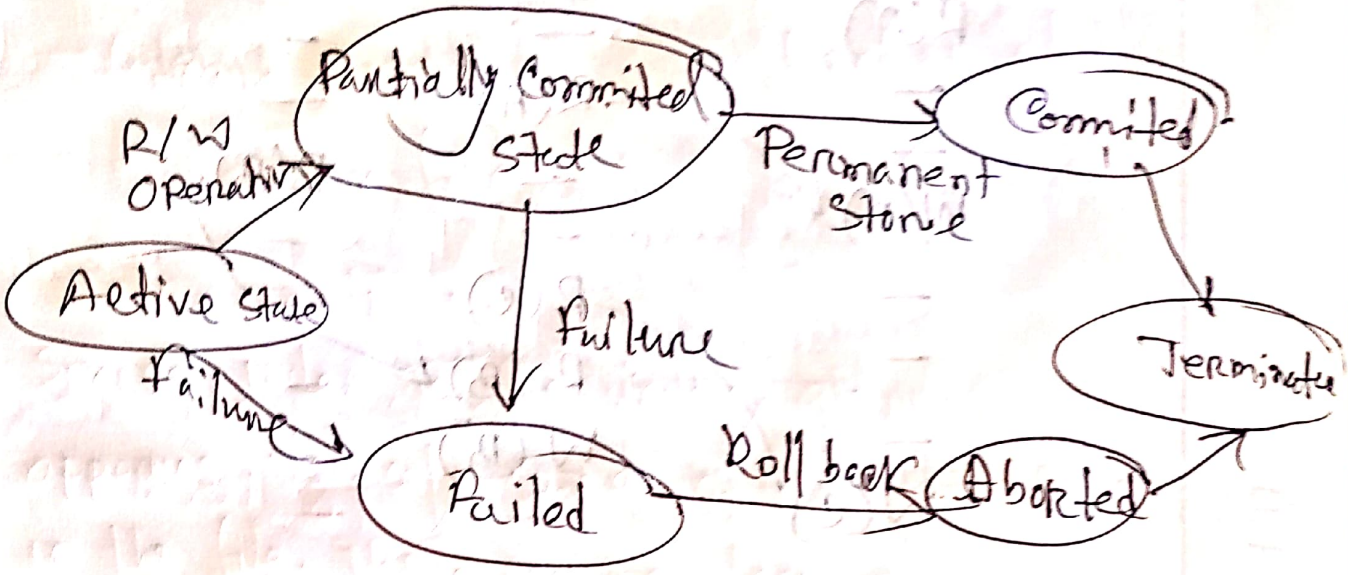
$T_2$ " $T_1$




Precedence graph

No cycle has created so it is ~~confl~~ Serializable schedule. And order of Serializability: ~~T1 T2 T3~~ $T_3 \rightarrow T_2 \rightarrow T_1$

d) What are the different state of transaction




8. a) Discuss how Raid can be used to improve reliability via redundancy.

RAID (Redundant Array of Independent Disks) is a technology which makes use of two or more hard drivers in order to improve performance, reliability or create large data volumes.

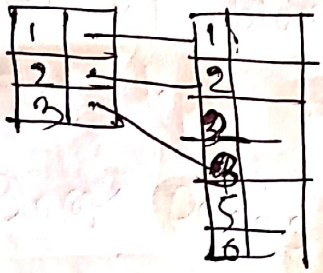
i. The chance that at least one disk out of a set of N disks will fail is much higher than the chance that a specific disk

will fail

ii. Redundancy stores extra information that is not needed normally, but that can be used in the event of failure of disk to rebuild the lost information.

b) What is sparse indexing? How does multilevel indexing improve the efficiency of searching an index file?

Sparse index is the index where an index record appears for only some of the search-key values in the file. They require less space & impose less maintance overhead for insertions & deletions. Index entry for each page of data file.



multilevel index: An index with two or more level is called a multi. Searching of records with a Multilevel Index requires significantly fewer I/O operations than does searching for records by binary search.

Advantages of / how does multilevel indexing improve the efficiency of searching an index file:

i. faster access to specific data
ii. Reduced search time
iii. Improved Data Management
iv. Scalability:

- A multilevel index reduces the number of blocks accessed when searching for a re cord, given its indexing field value
- To retain the benefits of using multilevel indexing while reducing index insertion & deletion problems.
- Designers adopted a multilevel indexing that leaves some space in each of its blocks, for inserting new entries.

c) Assumption:

- Each index level is stored in one disk block
- There is no overflow block containing additional index entries
- The desired entry is found at the last level of the index structure (leaf level)

Best Scenario:

fanout 'f': $\boxed{\text{Depth} = \log_f(1000)}$

The number of block reads in the best case is the depth of the index structure:

$$\text{Block Reads (Best Case)} = \text{Depth} = \log_f(1000)$$

Worst:

$$\text{Block Reads (Worst Case)} = \text{Depth} \times (F-1) = \log_f(1000) \times (F-1)$$

(F-1) represents the num of blocks

Instance of Relations:

Course Relation —

| Course_name | Room | Instruction |
|---|---|---|
| CSE001 | 6001 | XYZ |
| MATH002 | | |

Enrollment Relation —

| Course name | Stu_name | grade |
|---|---|---|
| CSE001 | X | A |
| CSE001 | Y | A+ |
| CSE001 | Z | A+ |
| MATH002 | | |
| MATH002 | | |

File Structure with Clustering:

Block-1 —

| Course_name | Student-name | grade |
|---|---|---|
| CSE101 | X | A |
| CSE101 | Y | A+ |
| CSE101 | Z | A+ |

Block2-

| Course-name | Stu name | grade |
|---|---|---|
| MATH002 | X | B |
| MATH002 | Y | A |
| MATH002 | Z | B+ |

In file structure, records are physically ordered based on the "Search-key"

records for the same course are clustered together, making it more efficient to retrieve information related to a specific course.