1.Explain various commands on TCL and dcl ?

ANS: Data Control Language:

- DCL commands are used to grant and take back authority from any database user.
- Here are some commands that come under DCL:
- Grant
- Revoke

a. Grant: It is used to give user access privileges to a database.

Syntax: grant <permissions>on <tablename to <user list>;

GRANT SELECT, UPDATE ON std TO SOME_USER, ANOTHER_USER;

b. Revoke: It is used to take back permissions from the user.

Syntax: Revoke <permissions>on <tablename from <user list>;

Example

REVOKE SELECT, UPDATE ON std FROM USER1, USER2;

Transaction Control Language:

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- COMMIT
- ROLLBACK
- SAVEPOINT

a. Commit: Commit command is used to save all the transactions to the database.

Syntax:

- COMMIT;

Example:

- DELETE FROM CUSTOMERS
- WHERE AGE = 25;
- COMMIT;

b. Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

Syntax:

ROLLBACK;

Example:

DELETE FROM CUSTOMERS

WHERE AGE = 25;

ROLLBACK;

c. SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntax:

- SAVEPOINT SAVEPOINT_NAME;

2.what is functional dependency. explain various functional dependencies and write about Armstrong axiom rules?

ANS:

Functional Dependency

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

1. X → Y

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

For example:

Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.

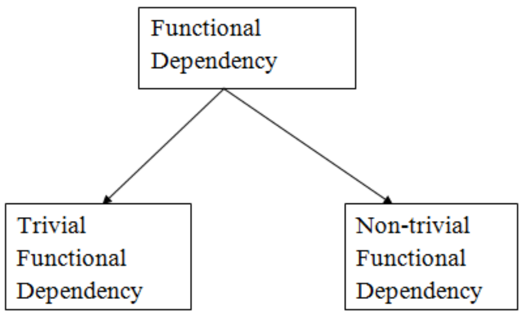
Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

1. Emp_Id → Emp_Name

We can say that Emp_Name is functionally dependent on Emp_Id.

Types of Functional dependency


Functional Dependency
Trivial Functional Dependency
Non-trivial Functional Dependency

1. Trivial functional dependency

- A → B has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: A → A, B → B

Example:

1. Consider a table with two columns Employee_Id and Employee_Name.
2. {Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as
3. Employee_Id is a subset of {Employee_Id, Employee_Name}.
4. Also, Employee_Id → Employee_Id and Employee_Name → Employee_Name are trivial d ependencies too.

2. Non-trivial functional dependency

- A → B has a non-trivial functional dependency if B is not a subset of A.
- When A intersection B is NULL, then A → B is called as complete non-trivial.

Example:

1. ID → Name,
2. Name → DOB

3.what is schema refinement and explain problems caused by redundancy

ANS: 1. Schema Refinement:

The Schema Refinement refers to refine the schema by using some technique. The best technique of schema refinement is decomposition.

Normalisation or Schema Refinement is a technique of organizing the data in the database.

It is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies.

Redundancy refers to repetition of same data or duplicate copies of same data stored in different locations.

Anomalies: Anomalies refers to the problems occurred after poorly planned and normalised databases where all the data is stored in one table which is sometimes called a flat file database.

Anomalies or problems facing without normalization(problems due to redundancy) :

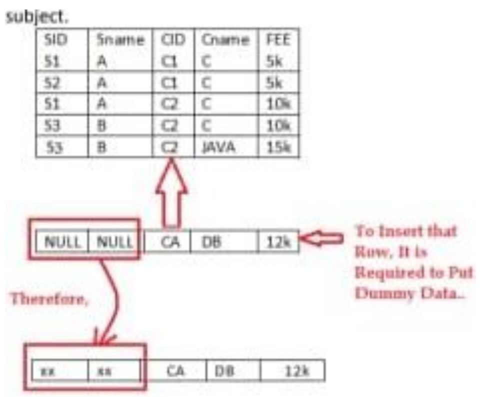
Anomalies refers to the problems occurred after poorly planned and unnormalised databases where all the data is stored in one table which is sometimes called a flat file database. Let us consider such type of schema –

Here all the data is stored in a single table which causes redundancy of data or say anomalies as SID and Sname are repeated once for same CID . Let us discuss anomalies one by one.

Due to redundancy of data we may get the following problems, those are-

1.insertion anomalies : It may not be possible to store some information unless some other

information is stored as well.



subject.

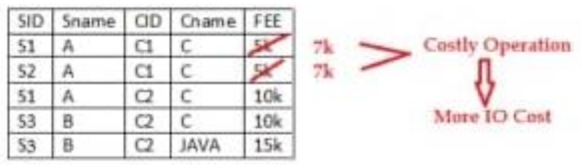
SID	Sname	CID	Cname	FEE
S1	A	C1	C	5k
S2	A	C1	C	5k
S1	A	C2	C	10k
S3	B	C2	C	10k
S3	B	C2	JAVA	15k

NULL	NULL	CA	DB	12k

xx	xx	CA	DB	12k

2.update anomalies: If one copy of redundant data is updated, then inconsistency is created

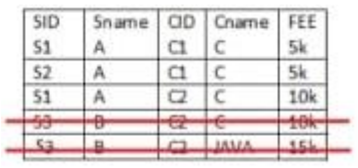
unless all redundant copies of data are updated.



SID	Sname	CID	Cname	FEE
S1	A	C1	C	~~5k~~ 7k
S2	A	C1	C	~~5k~~ 7k
S1	A	C2	C	10k
S3	B	C2	C	10k
S3	B	C2	JAVA	15k

3.deletion anomalies: It may not be possible to delete some information without losing some

other information as well.



SID	Sname	CID	Cname	FEE
S1	A	C1	C	5k
S2	A	C1	C	5k
S1	A	C2	C	10k
~~S3~~	~~B~~	~~C2~~	~~C~~	~~10k~~
~~S3~~	~~B~~	~~C2~~	~~JAVA~~	~~15k~~

Problem in updation / updation anomaly – If there is updation in the fee from 5000 to 7000,

then we have to update FEE column in all the rows, else data will become inconsistent.

Insertion Anomaly and Deletion Anomaly- These anomalies exist only due to redundancy,

otherwise they do not exist.

4.what is lock based concurrency control with suitable examples

ANS:

Concurrency:- Accessing same data by number of users at the same time.Transaction May success/May not be success.
To prevent this problems Locking mechanisms are used.

Locking Methods:A procedure used to control concurrent access to data when one transaction is accessing the database.
Locking mechanisms:----

Binary Lock:-
it has 2 states/values associated with each data item. These values are:
1.Locked-1
2.Unlocked-0

if data is locked then it can't be accessed by other transactions.
These locks are applied and removing using Lock() & Unlock() operation respectively.

- In binary locks,at a particular point in time,only one transaction can hold a lock on the data item.
- No other transaction will be able to access the same data concurrently.
- Hence, binary locks are very simple to apply but are not used pratically.....

Shared lock:-
If a transaction has shared lock on a data item, it can read the item, but can't update/add the data in it.
It is represented with "S"(lock-s)
It is also known as "Read Only Lock"

Exclusive Lock:-
If a transaction has Exclusive lock on a data item can be performed both read as well as write operations.
It is represented with "X"(lock-X)
In this multiple transactions don't modify the same data simultaneously.

Two Phase Locking (2PL):-
it is a concurrency control technique.
*Both locks & Unlocks
2PL Divides into 2 phases
1.Growing phase
2.Shrinking phase

Growing Phase:
locks are obtained but Not released.
Shrinking Phase:
locks are obtained but no new locks are acquired.

holding lock un-necessarily
*locking too early
*penalty to other transactions
These reduce the concurrency in 2PL

5.what is normalization explain 1nf,2nf,3nf to reduce redundancy in database

ANS:

Normalization

* The process of minimizing "Redundancy"
* It removes → Insertion Anomaly
  → updation Anomaly
  → Deletion Anomaly

Def :- "Normatization" divides the larger table into smaller tables and links them using relation ships.

Example:-

id	Name	age	Branch	HOD
1	Vaishu	18	AIML	TK
2	Vinod	18	AIML	TK
3	chandu	19	AIML	TK

id	Name	age	branch-id
1	Vaishu	18	101
2	Vinod	18	101
3	chandu	19	101

branchid	branch	HOD
101	AIML	TK

1) Insertion Anomaly :-
   1) Repetition of data
   2) If the branch is not alloted it has to be put as "NULL".
2) updation Anomaly :-
   1) If HOD is changed we need to change in all HOD coloumns.
   2) After decomposition we need to change in one place.
3) Deletion Anomaly :-
   1) If 2 diffrent informations are kept together eg:- Student records like student id, Branch if we delete student-id the Branch will also delete.

Types of Normal forms :-

Normal forms

- 1NF (First Normal form)
- 2NF (Second Normal form)
- 3NF (Third Normal form)
- BCNF (Boyce codd normal form)
- 4NF (Fourth Normal form
- 5NF (Fifth Normal form)

First Normal Form (1NF) :-

Rules for 1NF :-

① Single valued attribute (Atomic values). Every cell having only one value.

② Attribute domain should not change

③ Unique name for attributes (or) columns.

④ Order doesnot matter.

id	Name	age	subject
1	Sanju	18	DBMS, DSP
2	chaitra	18	DBMS
3	yash	19	FAM

⇩

id	Name	age	subject
1	Sanju	18	DBMS
1	Sanju	18	DSP
2	chaitra	18	DBMS
3	Yash	19	FAM

Second Normal Form (2NF) :-

Rules for 2NF :-

① The table should be in 1-NF

② No partial dependency

Dependency = Functional dependency = All columns are dependent on one "primary key"

Partial dependency :- Partial dependency happens if there are two or more primary keys in one table. That keys are called as "candidate key" partial dependendency will occur any one part of a candidate key.

Functional dependency :-

Any attribute depends only on a part of a "candidate key".

Example :- Consider '3' tables as student, subject score.

student

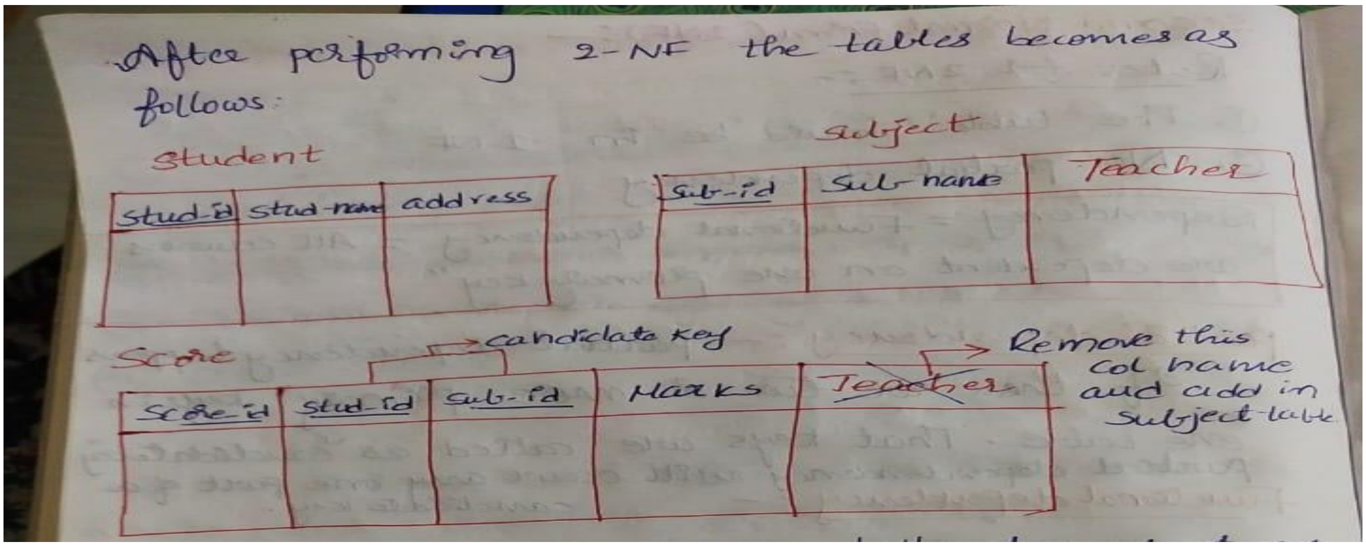
stud-id	studname	address

Subject

Sub-id	Subname

score

score id	stud-id	sub-id	Marks	Teacher


After performing 2-NF the tables becomes as follows:

Student

Stud-id	Stud-name	address

Subject

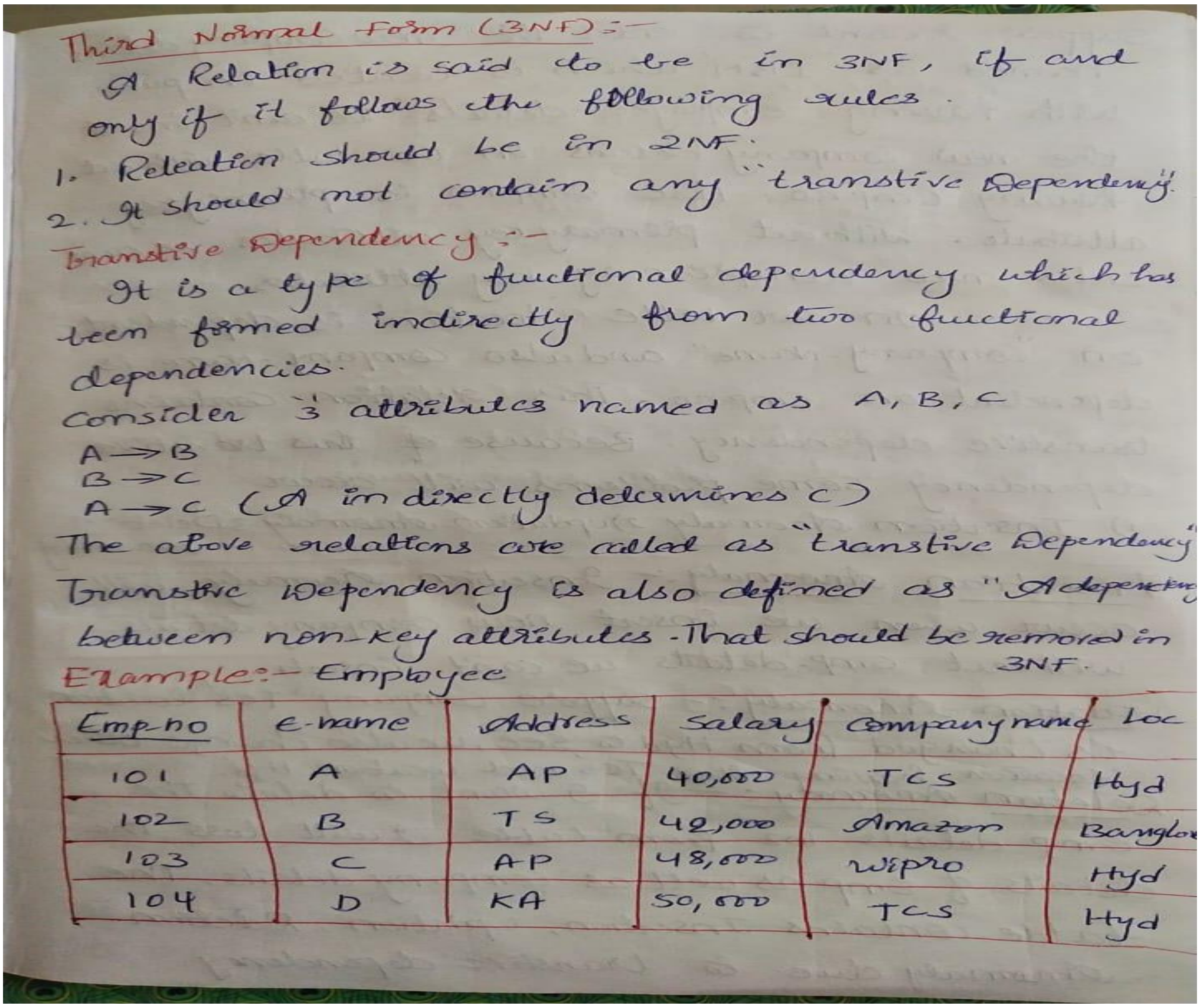
Sub-id	Sub-name	Teacher

Score

candidate key

Remove this col name and add in subject table

Score_id	Stud-id	Sub-id	Marks	~~Teacher~~


Third Normal Form (3NF):-

A Relation is said to be in 3NF, if and only if it follows the following rules.

1. Releation should be in 2NF.
2. It should not contain any "transtive Dependency".

Transtive Dependency :-

It is a type of fuctional dependency which has been formed indirectly from two fuctional dependencies.

Consider 3 attributes named as A, B, C

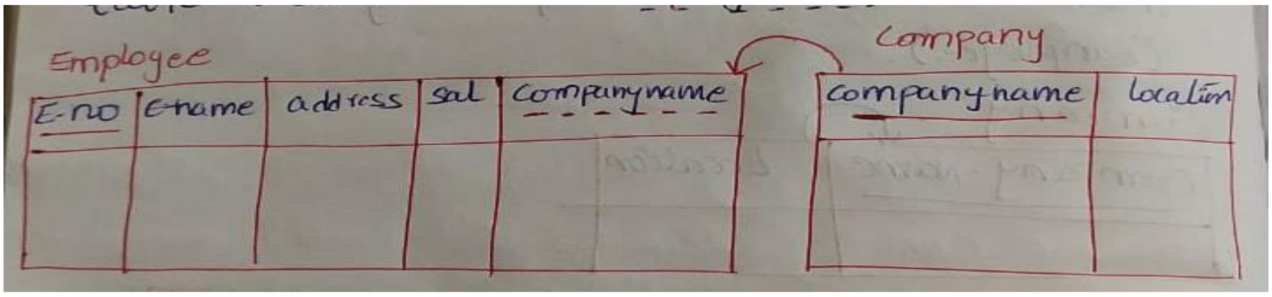
A → B
B → C
A → C (A indirectly determines C)

The above relations are called as "transtive Dependency"

Transtive Dependency is also defined as "A dependency between non-key attributes. That should be removed in 3NF.

Example:- Employee

Emp-no	E-name	Address	Salary	Company name	Loc
101	A	AP	40,000	TCS	Hyd
102	B	TS	42,000	Amazon	Banglo
103	C	AP	48,000	Wipro	Hyd
104	D	KA	50,000	TCS	Hyd



6.Explain concurrency control with timestamp-based locking protocol

ANS:

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.

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 T_i is denoted as $TS(T_i)$.
- 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.

T1	T2
10	20
R(A)	
	W(A)

RTS = 10	RTS>TS(Ti) -> 10>20 -.> FALSE
WTS= 20	WTS > TS(Ti) -> 20>20 -> FALSE
TS(Ti)=20	THE OPERATION EXECUTED

7.Explain join and types of joins with suitable examples

ANS: **SQL JOIN:**

The name shows, JOIN means *to combine something*. In case of SQL, JOIN means "to combine two or more tables".

The SQL JOIN clause takes records from two or more tables in a database and combines it together.

ANSI standard SQL defines four types of JOIN :

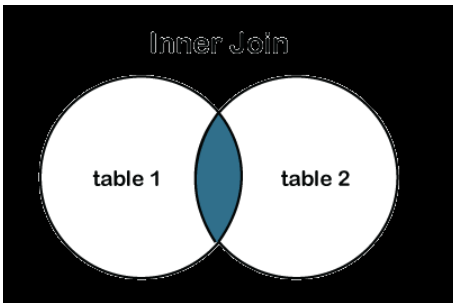
- inner join,
- left outer join,
- right outer join,
- full outer join.

Inner Join

The inner join is used to select all matching rows or columns in both tables or as long as the defined condition is valid in SQL.

Syntax:

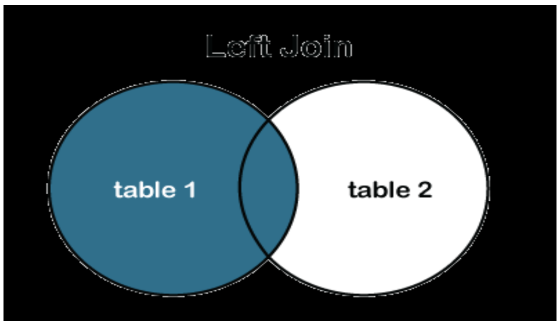
Select column_1, column_2, column_3 FROM table_1 INNER JOIN table_2 ON table_1.column = table_2.column;



LEFT JOIN

The LEFT JOIN is used to retrieve all records from the left table (table1) and the matched rows or columns from the right table (table2). If both tables do not contain any matched rows or columns, it returns the NULL.

Select column_1, column_2, column(s) FROM table_1 LEFT JOIN table_2 ON table_1.column_name = table_2.column _name;

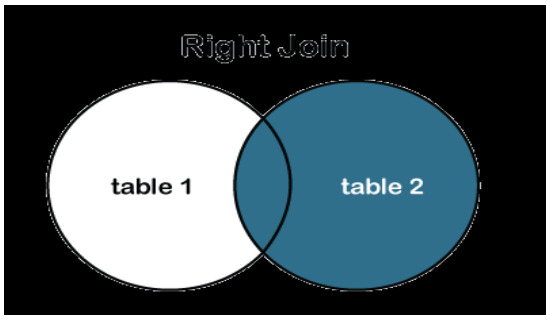


RIGHT JOIN or RIGHT Outer JOIN:

The RIGHT JOIN is used to retrieve all records from the right table (table2) and the matched rows or columns from the left table (table1). If both tables do not contain any matched rows or columns, it returns the NULL.

Syntax:

Select column_1, column_2, column(s) FROM table_1 RIGHT JOIN table_2 ON table_1.column_name = table_2.colu mn_name;

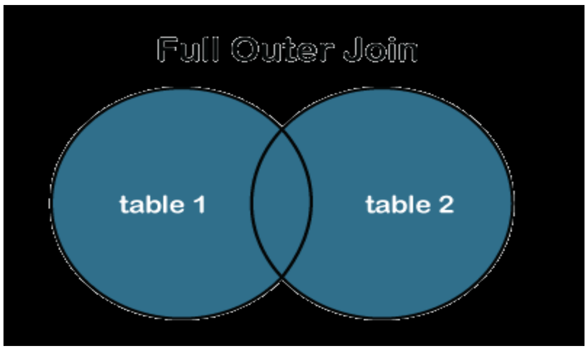


FULL JOIN or FULL Outer JOIN:

It is a combination result set of both LEFT JOIN and RIGHT JOIN. The joined tables return all records from both the tables and if no matches are found in the table, it places NULL. It is also called a FULL OUTER JOIN.

Syntax:

Select column_1, column_2, column(s) FROM table_1 FULL JOIN table_2 ON table_1.column_name = table_2.column _name;



8.what is decomposition and explain types of decomposition with examples

ANS:

Decomposition

A functional **decomposition** is the process of breaking down the functions of an organization into progressively greater (finer and finer) levels of detail.

In decomposition, one function is described in greater detail by a set of other supporting functions.

The decomposition of a relation scheme R consists of replacing the relation schema by two or more relation schemas that each contain a subset of the attributes of R and together include all attributes in R.

Decomposition helps in eliminating some of the problems of bad design such as redundancy, inconsistencies and anomalies.

There are two types of decomposition :

1. Lossy Decomposition
2. Lossless Join Decomposition

Lossy Decomposition :

"The decompositio of relation R into R1 and R2 is **lossy** when the join of R1 and R2 does not yield the same relation as in R."

One of the disadvantages of decomposition into two or more relational schemes (or tables) is that some information is lost during retrieval of original relation or table.

Consider that we have table STUDENT with three attribute roll_no , sname and department.

STUDENT:

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

This relation is decomposed into two relation no_name and name_dept :

No_name:

Roll_no	Sname
111	parimal
222	parimal

Name_dept :

Sname	Dept
parimal	COMPUTER
parimal	ELECTRICAL

In lossy decomposition ,spurious tuples are generated when a natural join is applied to the relations in the decomposition.

stu_joined :

Roll_no	Sname	Dept
111	parimal	COMPUTER
111	parimal	ELECTRICAL
222	parimal	COMPUTER
222	parimal	ELECTRICAL

The above decomposition is a bad decomposition or Lossy decomposition.

Lossless Join Decomposition :

"The decompositio of relation R into R1 and R2 is **lossless** when the join of R1 and R2 yield the same relation as in R."

A relational table is decomposed (or factored) into two or more smaller tables, in such a way that the designer can capture the precise content of the original table by joining the decomposed parts. This is called lossless-join (or non-additive join) decomposition.

This is also refferd as non-additive decomposition.

The lossless-join decomposition is always defined with respect to a specific set F of dependencies.

Consider that we have table STUDENT with three attribute roll_no , sname and department.

STUDENT :

Roll_no	Sname	Dept

111	parimal	COMPUTER
222	parimal	ELECTRICAL

This relation is decomposed into two relation Stu_name and Stu_dept :

Stu_name: **Stu_dept :**

Roll_no	Sname
111	parimal
222	parimal

Roll_no	Dept
111	COMPUTER
222	ELECTRICAL

Now ,when these two relations are joined on the comman column 'roll_no' ,the resultant relation will look like stu_joined.

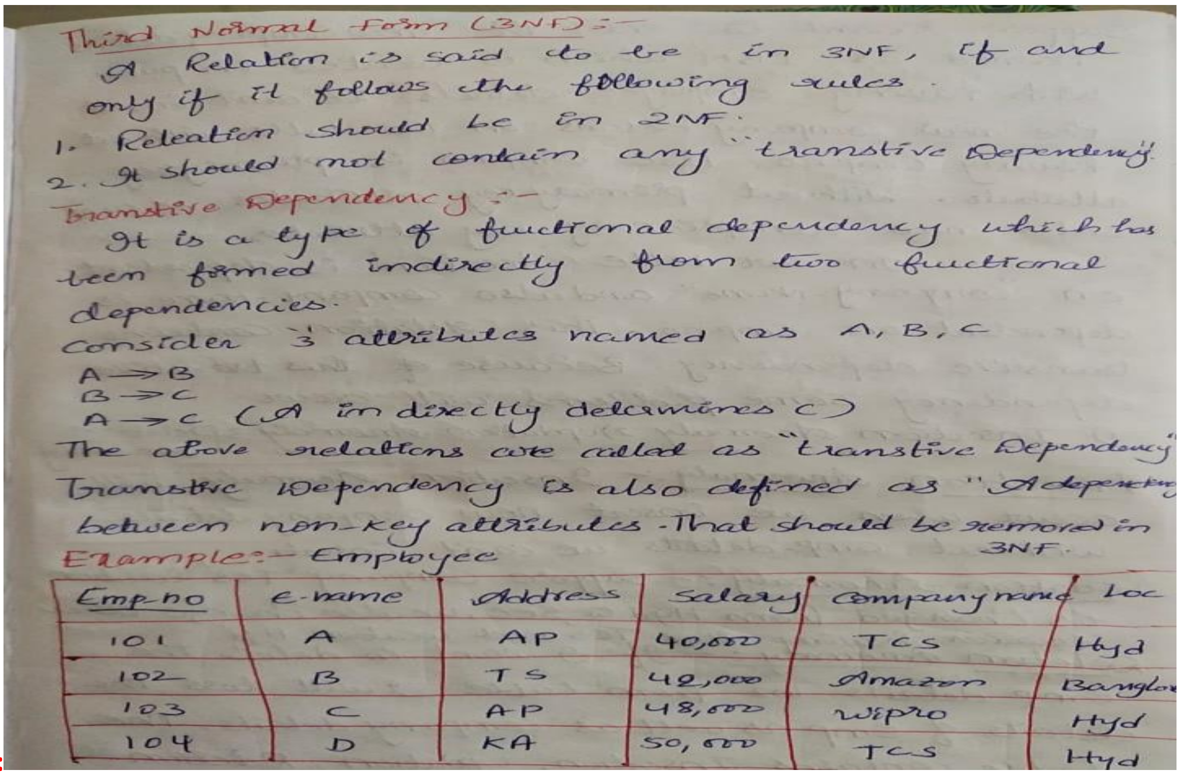
stu_joined :

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

In lossless decomposition, no any spurious tuples are generated when a natural joined is applied to the relations in the decomposition.

9.what is 3nf and bcnf. Explain difference between 3nf and bcnf

ANS:



Third Normal Form (3NF):-

A Relation is said to be in 3NF, if and only if it follows the following rules.

1. Releation should be in 2NF.
2. It should not contain any "transtive Dependency".

Transtive Dependency:-

It is a type of functional dependency which has been formed indirectly from two functional dependencies.

consider 3 attributes named as A, B, C

$A \rightarrow B$

$B \rightarrow C$

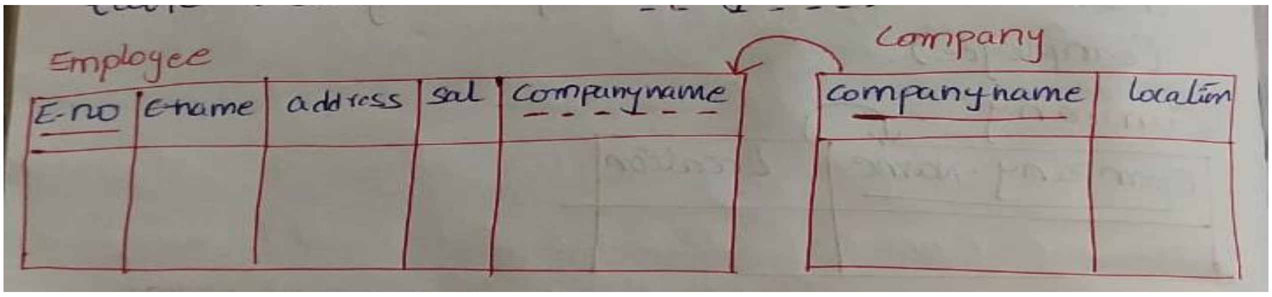
$A \rightarrow C$ (A indirectly determines C)

The above relations are called as "transtive Dependency"

Transtive Dependency is also defined as "A dependency between non-key attributes. That should be removed in 3NF.

Example:- Employee

Emp-no	E-name	Address	Salary	Company name	Loc
101	A	AP	40,000	TCS	Hyd
102	B	TS	42,000	Amazon	Banglo
103	C	AP	48,000	Wipro	Hyd
104	D	KA	50,000	TCS	Hyd



Employee

E-no	E-name	address	sal	Companyname

Company

companyname	location

Boyce Codd normal form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key(SK-is a combination of col's that uniquely identifies any row in a table.
- E.F CODD and Raymond F-Boyce developed BCNF

Example: Let's assume there is a company where employees work in more than one department.

EMPLOYEE table:

In the above table Functional dependencies are as follows:

1. EMP_ID → EMP_COUNTRY

2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP-ID, EMP-DEPT}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

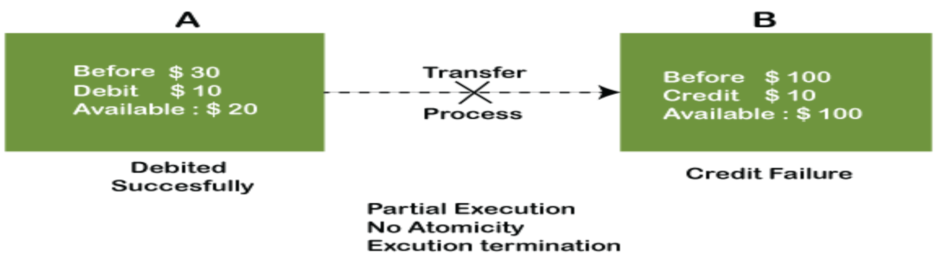
EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

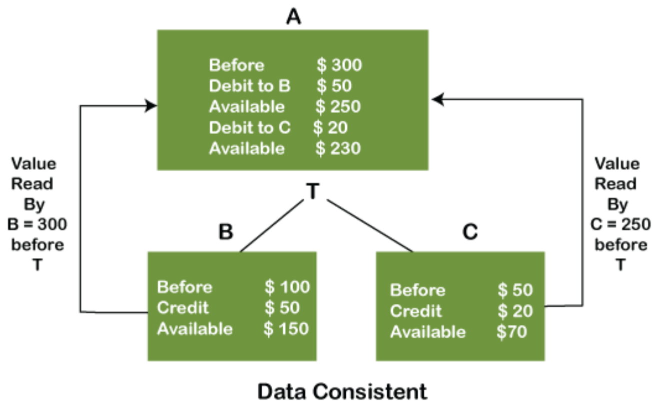
10.what is acid properties. Explain transaction states with a neat diagram

ANS: the data base maintain the following properties called ACID properties.

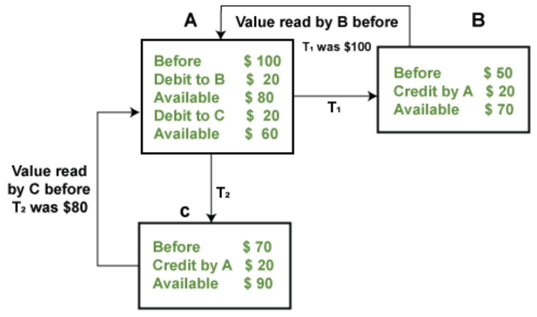
1.Atomicity: either it should be performed or executed completely or should not be executed at all. It further means that the operation should not break in between or execute partially. In the case of executing operations on the transaction, the operation should be completely executed and not partially.



2.Consistency: means that the value should remain preserved always.



3.Isolation: the operation on one database should begin when the operation on the first database gets complete.



Isolation - Independent execution of T_1 & T_2 by A

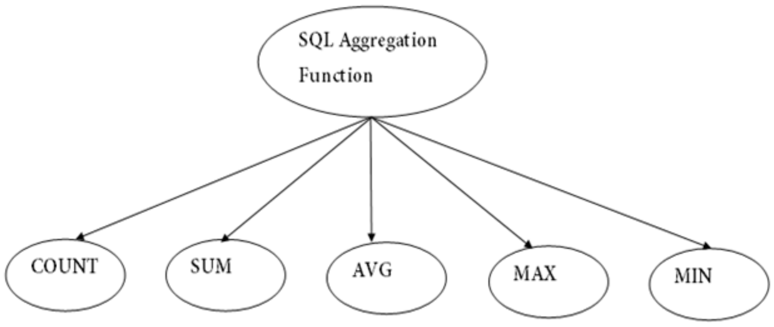
4.Durability: ensures that the data after the successful execution of the operation becomes permanent in the database. In DBMS, the term durability ensures that the data after the successful execution of the operation becomes permanent in the database. The durability of the data should be so perfect that even if the system fails or leads to a crash, the database still survives. However, if gets lost, it becomes the responsibility of the recovery manager for ensuring the durability of the database. For committing the values, the COMMIT command must be used every time we make changes.

11.Explain aggregate functions with example

ANS: SQL Aggregate Functions

- SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.
- It is also used to summarize the data.

Types of SQL Aggregation Function



PRODUCT_MAST

PRODUCT	COMPANY	QTY	RATE	COST
Item1	Com1	2	10	20
Item2	Com2	3	25	75
Item3	Com1	2	30	60
Item4	Com3	5	10	50
Item5	Com2	2	20	40
Item6	Cpm1	3	25	75
Item7	Com1	5	30	150
Item8	Com1	3	10	30
Item9	Com2	2	25	50
Item10	Com3	4	30	120

1. COUNT FUNCTION

- COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.
- COUNT function uses the COUNT(*) that returns the count of all the rows in a specified table. COUNT(*) considers duplicate and Null.

Syntax COUNT(*) or COUNT([ALL|DISTINCT] expression)

Ex: SELECT COUNT(*) FROM PRODUCT_MAST;
Output:
```
10
```

Ex: SELECT COUNT(DISTINCT COMPANY) FROM PRODUCT_MAST;
Output:
```
3
```

2. SUM Function
Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.
Syntax SUM() or SUM([ALL|DISTINCT] expression)
Example: SELECT SUM(COST) FROM PRODUCT_MAST;
Output: 670

3. AVG function

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

Syntax AVG() or AVG([ALL|DISTINCT] expression)

Example: SELECT AVG(COST) FROM PRODUCT_MAST;

Output:

```
67.00
```

4. MAX Function

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

Syntax MAX() or MAX([ALL|DISTINCT] expression)

Example: SELECT MAX(RATE) FROM PRODUCT_MAST;

```
30
```

5. MIN Function

MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

Syntax MIN() or MIN([ALL|DISTINCT] expression)

Example: SELECT MIN(RATE) FROM PRODUCT_MAST;

Output:

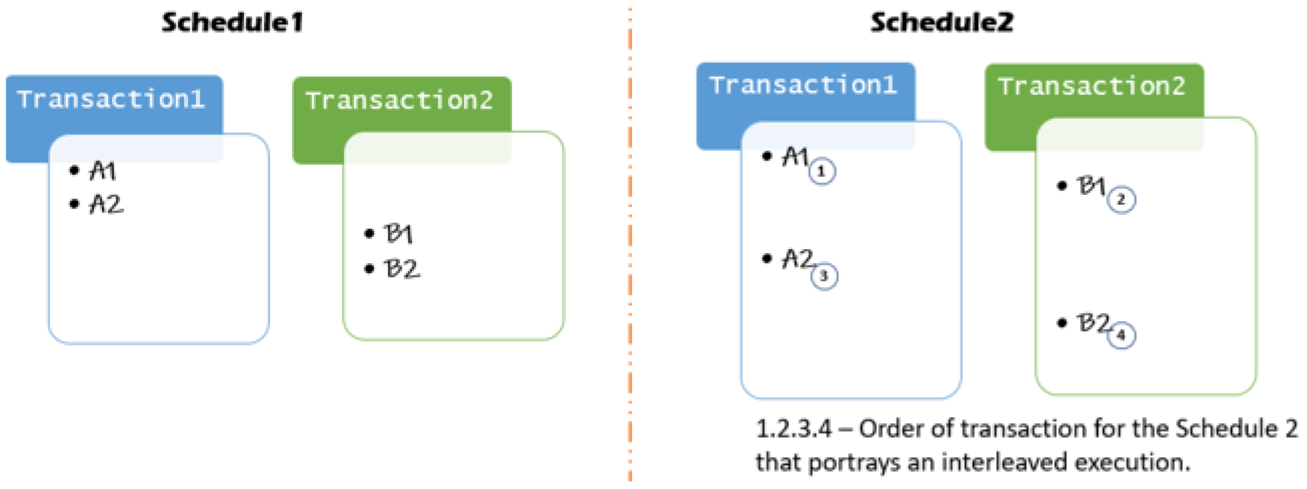
```
10
```

12.what is serializability. Explain conflict serializability and view serializability with an example

ANS: Serial schedule both by definition and execution means that the transactions bestowed upon it will take place serially, that is, one after the other. This leaves no place for inconsistency within the database. But, when a set of transactions are scheduled non-serially, they are interleaved leading to the problem of concurrency within the database. Non-serial schedules do not wait for one transaction to complete for the other one to begin. Serializability in DBMS decides if an interleaved non-serial schedule is serializable or not.

Example of Serializability

Consider 2 schedules, Schedule1 and Schedule2:



1.2.3.4 – Order of transaction for the Schedule 2 that portrays an interleaved execution.

- Schedule1 is a serial schedule consisting of Transaction1 and Transaction2 wherein the operations on data item A (A1 and A2) are performed first and later the operations on data item B (B1 and B2) are carried out serially.
- Schedule2 is a non-serial schedule consisting of Transaction1 and Transaction2 wherein the operations are interleaved.

Explanation: In the given scenario, schedule2 is serializable if the output obtained from both Schedule2 and Schedule1 are equivalent to one another In a nutshell, a transaction within a given non-serial schedule is serializable if its outcome is equivalent to the outcome of the same transaction when executed serially.

types of Serializability

. Conflict Equivalent Schedule

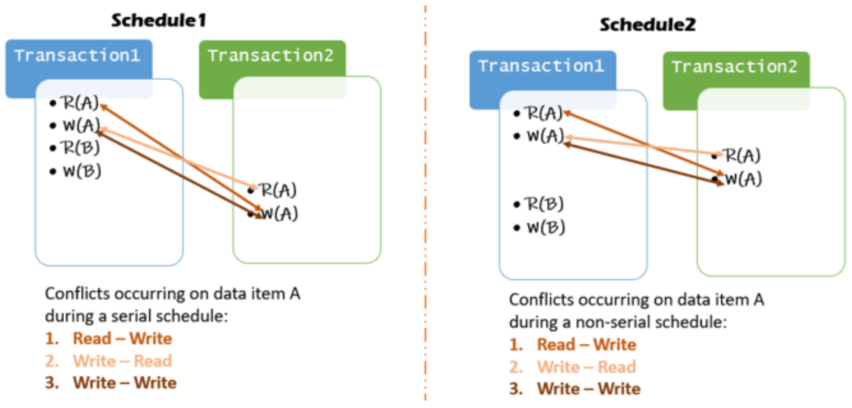
When either of a conflict operation such as Read-Write or Write-Read or Write-Write is implemented on the same data item at the same time within different transactions then the schedule holding such transactions is said to be a conflict schedule. The prerequisites for such conflict schedule are:

1. The conflict operations are to be implemented on the same data item.
2. The conflict operations (RW, WR, WW) must take place within different transactions.
3. At least one of the conflict operations must be the write operation.
4. Two Read operations will not create any conflict.

Two schedules (one being serial schedule and another being non-serial) are said to be conflict serializable if the conflict operations in both the schedules are executed in the same order.

Example:

Consider 2 schedules, Schedule1 and Schedule2,

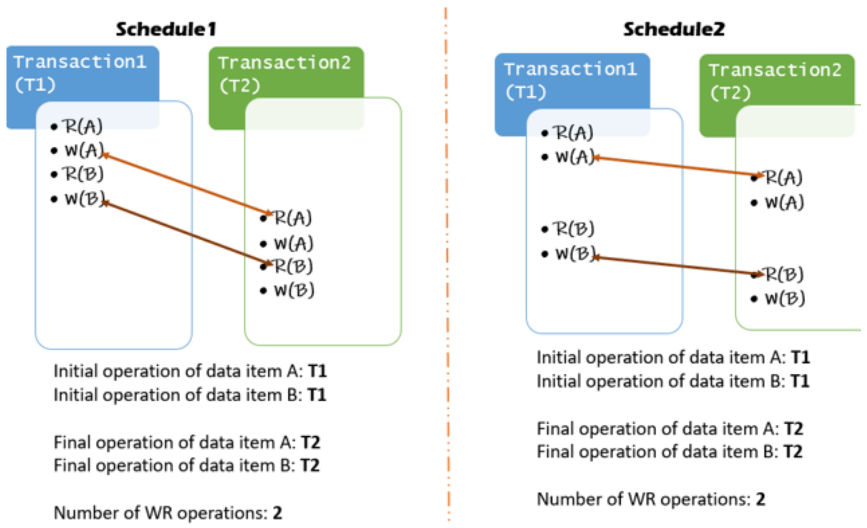


Schedule2 (a non-serial schedule) is considered to be conflict serializable when its conflict operations are the same as that of Shedule1 (a serial schedule).

View Equivalent Schedule

Two schedules (one being serial schedule and another being non-serial) are said to be view serializable if they satisfy the rules for being view equivalent to one another.

The rules to be upheld are:



Initial operation of data item A: **T1**
Initial operation of data item B: **T1**

Final operation of data item A: **T2**
Final operation of data item B: **T2**

Number of WR operations: **2**

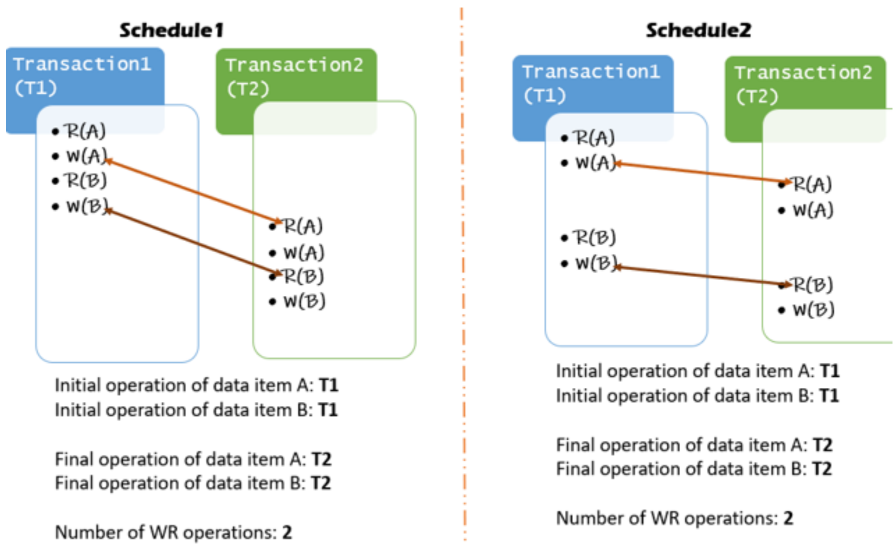
Initial operation of data item A: **T1**
Initial operation of data item B: **T1**

Final operation of data item A: **T2**
Final operation of data item B: **T2**

Number of WR operations: **2**

1. Initial values of the data items involved within a schedule must be the same.
2. Final values of the data items involved within a schedule must be the same.
3. The number of WR operations performed must be equivalent for the schedules involved.

Example:



Initial operation of data item A: **T1**
Initial operation of data item B: **T1**

Final operation of data item A: **T2**
Final operation of data item B: **T2**

Number of WR operations: **2**

Initial operation of data item A: **T1**
Initial operation of data item B: **T1**

Final operation of data item A: **T2**
Final operation of data item B: **T2**

Number of WR operations: **2**

The (non-serial) Schedule2 is considered as a view equivalent of the (serial) Schedule1, when the 3 rules of view serializability are satisfied. For the example shown above,

- The Initial transaction of read operation on the data items A and B both begin at T1
- The Final transaction of write operations on the data items A and B both end at T2
- The number of updates from write-read operations are 2 in both the cases