## 1. What is Normalization? Discuss different types of Normal Forms.

Ans-

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows.

# Types of Normal Forms:

| Normal Form | Description |
|---|---|
| 1NF | A relation is in 1NF if it contains an atomic value. A relation is in first normal form if every attribute in that relation is singled valued attribute. |
| 2NF | A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key. |
| 3NF | A relation will be in 3NF if it is in 2NF and no transition dependency exists. A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form. |
| BCNF | A relation R is in BCNF if R is in Third Normal Form and for every FD, LHS is super key. A relation is in BCNF iff in every non-trivial functional dependency X –> Y, X is a super key. |
| 4NF | A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency. |
| 5NF | A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless. |

# Advantages of Normalization

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Enforces the concept of relational integrity.

# Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

## 2. ***Define the terms Full Functional Dependency and Partial Functional Dependency?***

**Ans –**

| Serial no. | Full Functional Dependency | Partial Functional Dependency |
|---|---|---|
| 1. | A functional dependency X->Y is a fully functional dependency if Y is functionally dependent on X and Y is not functionally dependent on any proper subset of X. | A functional dependency X->Y is a partial dependency if Y is functionally dependent on X and Y can be determined by any proper subset of X. |
| 2. | In full functional dependency, the non-prime attribute is functionally dependent on the candidate key. | In partial functional dependency, the non-prime attribute is functionally dependent on part of a candidate key. |
| 3. | In fully functional dependency, if we remove any attribute of X, then the dependency will not exist anymore. | In partial functional dependency, if we remove any attribute of X, then the dependency will still exist. |
| 4. | Full Functional Dependency equates to the normalization standard of Second Normal Form. | Partial Functional Dependency does not equate to the normalization standard of Second Normal |

| Serial no. | Full Functional Dependency | Partial Functional Dependency |
|---|---|---|
| | | Form. Rather, 2NF eliminates the Partial Dependency. |
| 5. | An attribute A is fully functional dependent on another attribute B if it is functionally dependent on that attribute, and not on any part (subset) of it. | An attribute A is partially functional dependent on other attribute B if it is functionally dependent on any part (subset) of that attribute. |
| 6. | Functional dependency enhances the quality of the data in our database. | Partial dependency does not enhance the data quality. It must be eliminated in order to normalize in the second normal form. |

27. ***Explain Multi Valued Dependency and Join Dependency with suitable examples***.

# Multivalued Dependency

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

**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.

# Join Dependency

Whenever we can recreate a table by simply joining various tables where each of these tables consists of a subset of the table's attribute, then this table is known as a Join Dependency. Thus, it is like a generalization of MVD. The concept of Join Dependency is directly based on the concept of 5NF, or Fifth Normal Form. Similar to functional or multivalued dependency, join dependency is a constraint. It is satisfied only if and only if the relation concerned is the joining of a set of projections.

**Example of Join Dependency**

Suppose we have the following table R:

| E_Name | Company | Product |
|---|---|---|
| Rohan | Comp1 | Jeans |
| Harpreet | Comp2 | Jacket |
| Anant | Comp3 | TShirt |

- We can break, or decompose the above table into three tables, this would mean that the table **is not in 5NF!**
- The **three decomposed tables** would be:

**1. R1: The table with columns E_Name and Company.**

| E_Name | Company |
|---|---|
| Rohan | Comp1 |
| Harpreet | Comp2 |
| Anant | Comp3 |

**2. R2: The table with columns E_Name and Product.**

| E_Name | Product |
|---|---|
| Rohan | Jeans |
| Harpreet | Jacket |
| Anant | TShirt |

**3. R3: The table with columns Company and Product.**

| Company | Product |
|---|---|
| Comp1 | Jeans |
| Comp2 | Jacket |
| Comp3 | TShirt |

**Note:** If the natural join of all three tables yields the relation table R, the relation will be said to have join dependency.

## ***28. Explain Loss Join Decomposition, Non-Additive Join and Dependency Preserving***.

**Lossless-join decomposition** is a process in which a relation is decomposed into two or more relations. This property guarantees that the extra or less tuple generation problem does not occur and no information is lost from the original relation during the decomposition. It is also known as non-additive join decomposition.
When the sub relations combine again then the new relation must be the same as the original relation was before decomposition.
In Lossless Decomposition, we select the common attribute and the criteria for selecting a common attribute is that the common attribute must be a candidate key or super key in either relation R1, R2, or both.

The decomposition is lossless when it satisfies the following statement −

- If we union the sub Relation R1 and R2 then it must contain all the attributes that are available in the original relation R before decomposition.
- Intersections of R1 and R2 cannot be Null. The sub relation must contain a common attribute. The common attribute must contain unique data.

## Dependency Preserving

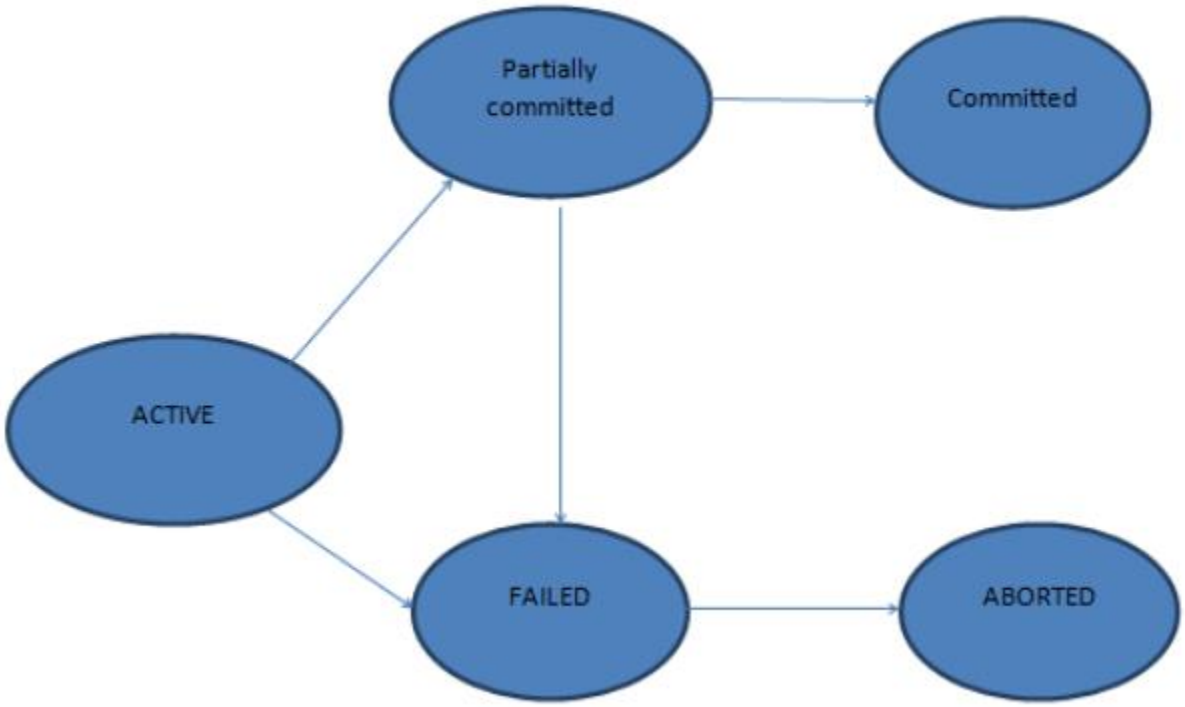
- It is an important constraint of the database.
- In the dependency preservation, at least one decomposed table must satisfy every dependency.
- If a relation R is decomposed into relation R1 and R2, then the dependencies of R either must be a part of R1 or R2 or must be derivable from the combination of functional dependencies of R1 and R2.
- For example, suppose there is a relation R (A, B, C, D) with functional dependency set (A->BC). The relational R is decomposed into R1(ABC) and R2(AD) which is dependency preserving because FD A->BC is a part of relation R1(ABC).

➔ Let R is decomposed into {R1, R2,....,Rn} with projected FD set {F1,F2,.....Fn}. This decomposition is dependency preserving if $F^+ = \{F1 \cup F2 \cup .....Fn\}^+$.

Example

- Let the relation R{A,B,C,D,E} F:{AB->C, C->D, AB->D} R is decomposed to R1(A,B,C), R2(D,E). Prove decomposition is dependency preserving.
- Solution
- F1={AB->C}
- F2={C->D}
- => (F1 u F2) = {AB->C, C->D}
- AB+ under (F1 U F2) = {A,B,C,D} => AB->D is under (F1 U F2)
- F+ = $(F1 \cup F2)^+$
- => Decomposition is dependency preserving.

## ***29. Explain State Transition Diagram for Transaction Execution***.




The following are the states of the above 'State Transition Diagram' :
**ACTIVE**: The initial state; the state remains constant while the transaction executes.
**PARTIALLY COMMITTED:** After the final statement has been executed.
**FAILED**: After the discovery that normal execution can no longer proceed.

**ABORTED:** The state after transaction roll-back and data restoration to the start of the transaction
**COMMITTED**: After successful completion.

## 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 the 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

## ***30. Explain the terms Cascadeless Schedules, Recoverable Schedules, Serializable Schedules, Concurrent Schedules, Equivalent Schedules, Blind Writes.***

**Ans-** Transactions are a set of instructions that perform operations on databases. When multiple transactions are running concurrently, then a sequence is needed in which the operations are to be performed because at a time, only one operation can be performed on the database. This sequence of operations is known as Schedule, and this process is known as Scheduling.

### Cascade less Schedule

If in a schedule, a transaction is not allowed to read a data item until and unless the last transaction that has been written is committed/aborted, then such a schedule is called a Cascadeless Schedule. It avoids cascading rollback and thus saves CPU time. To prevent cascading rollbacks, it disallows a transaction from reading uncommitted changes from another transaction in the same Schedule.

### Recoverable Schedule

A schedule is recoverable if each transaction commits only after all the transactions from which it has read have committed. In other words, if some transaction Ty reads a value that has been updated/written by some other transaction Tx, then the commit of Ty must occur after the commit of Tx.

## Serializable schedule

- 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.

### Concurrent Schedules:

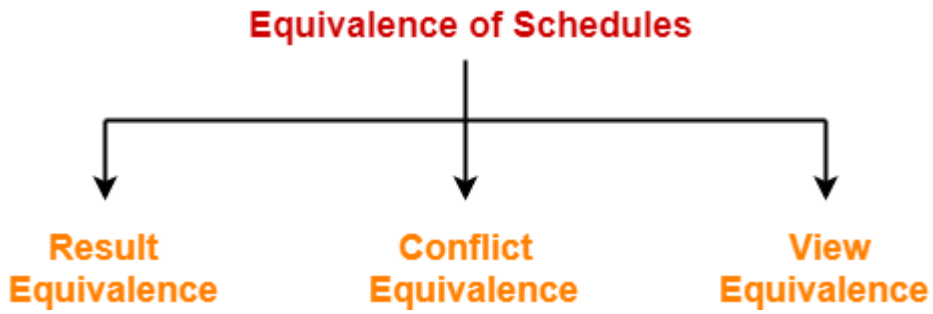
A schedule is said to be concurrent in case the instructions of the transactions get executed preemptively.

When the database system executes several transactions concurrently, the corresponding schedule no longer needs to be serial.

If two transactions are running concurrently, the operating system may execute one transaction for a little while, then perform a context switch, execute the second transaction for some time, and then switch back to the first transaction for some time, and so on. With multiple transactions, the CPU time is shared among all the transactions.

## Equivalence of Schedules-

In DBMS, schedules may have the following three different kinds of equivalence relations among them-




1. Result Equivalence
2. Conflict Equivalence
3. View Equivalence

## 1. Result Equivalent Schedules-

- If any two schedules generate the same result after their execution, then they are called as result equivalent schedules.
- This equivalence relation is considered of least significance.
- This is because some schedules might produce same results for some set of values and different results for some other set of values.

## 2. Conflict Equivalent Schedules-

If any two schedules satisfy the following two conditions, then they are called as conflict equivalent schedules-

1. The set of transactions present in both the schedules is same.
2. The order of pairs of conflicting operations of both the schedules is same.

## 3. View Equivalent Schedules-

Consider two schedules S1 and S2 each consisting of two transactions T1 and T2.

Schedules S1 and S2 are called view equivalent if the following three conditions hold true for them-

### Condition-01:

For each data item X, if transaction $T_i$ reads X from the database initially in schedule S1, then in schedule S2 also, $T_i$ must perform the initial read of X from the database.

Thumb Rule

"Initial readers must be same for all the data items".

## Condition-02:

If transaction $T_i$ reads a data item that has been updated by the transaction $T_j$ in schedule S1, then in schedule S2 also, transaction $T_i$ must read the same data item that has been updated by the transaction $T_j$.

Thumb Rule

"Write-read sequence must be same.".

## Condition-03:

For each data item X, if X has been updated at last by transaction $T_i$ in schedule S1, then in schedule S2 also, X must be updated at last by transaction $T_i$.

Thumb Rule

"Final writers must be same for all the data items".

### Blind write

Performing the Writing operation (updating), without reading operation, such write operation is known as a blind write. Blind write is simply when a transaction writes without reading. i.e a transaction have WRITE(Q), but no READ(Q) before it. So, the transaction is writing to the database "blindly" without reading previous value.

### *31. Explain View Serializability, Conflict Serializability.*

# View Serializability

- A schedule will view serializable if it is view equivalent to a serial schedule.
- If a schedule is conflict serializable, then it will be view serializable.
- The view serializable which does not conflict serializable contains blind writes.

View serializability is a concept that is used to compute whether schedules are View-Serializable or not. A schedule is said to be View-Serializable if it is view equivalent to a Serial Schedule (where no interleaving of transactions is possible).

# 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.

Conflict Serializability checks if a non-serial schedule is conflict serializable or not. A non-serial schedule is conflict serializable if it can convert into a serial schedule by swapping its non-conflicting operations.

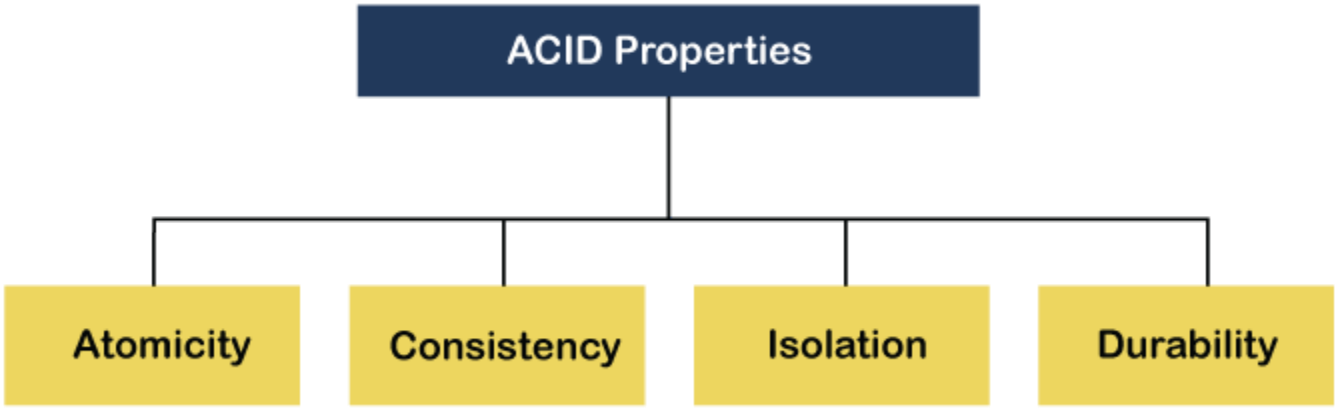
Conflict Serializable: A schedule is called conflict serializable if it can be transformed into a serial schedule by swapping non-conflicting operations.

Conflicting operations: Two operations are said to be conflicting if all conditions satisfy:

1. They belong to different transactions
2. They operate on the same data item
3. At Least one of them is a write operation

## ***(32). Explain ACID properties of a Transaction***.

The expansion of the term ACID defines for:




**1) Atomicity:** The term atomicity defines that the data remains atomic. It means if any operation is performed on the data, 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:** The word consistency means that the value should remain preserved always. In DBMS, the integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always. In the case of transactions, the integrity of the data is very essential so that the database remains consistent before and after the transaction. The data should always be correct.

**4) Isolation:** The term 'isolation' means separation. In DBMS, Isolation is the property of a database where no data should affect the other one and may occur concurrently. In short, the operation on one database should begin when the operation on the first database gets complete. It means if two operations are being performed on two different databases, they may not affect the value of one another.

**4) Durability:** Durability ensures the permanency of something. 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.

## ***(33). Explain VIEWS and their types.***

- Views in SQL are considered as a virtual table. A view also contains rows and columns.
- To create the view, we can select the fields from one or more tables present in the database.

- A view can either have specific rows based on certain condition or all the rows of a table.

# Types of Views

1. System Defined Views

System-defined Views are predefined Views that already exist in the Master database of SQL Server. These are also used as template Views for all newly created databases.

a. Information Schema View--In SQL Server we have twenty different schema views. These are used to display information in a database, like as tables and columns.
b. Catalog Views - were introduced with SQL Server 2005. These are used to show database self-describing information.
c. Dynamic Management Views - were introduced in SQL Server 2005. These Views give the administrator information about the database about the current state of the SQL Server machine.

2. User Defined Views
These types of views are defined by users. We have two types of user-defined views.
a. Simple View - When we create a view on a single table, it is called a simple view.
b. Complex View - When we create a view on more than one table, it is called a complex view.

## ***34. Explain Cursors and their types.***

## Cursor in SQL Server

A cursor in SQL Server is a d**atabase object that allows us to retrieve each row at a time and manipulate its data**. A cursor is nothing more than a pointer to a row. It's always used in conjunction with a SELECT statement.

The SQL Server cursor's purpose is to update the data row by row, change it, or perform calculations that are not possible when we retrieve all records at once. It's also useful for performing administrative tasks like SQL Server database backups in sequential order.

# Types of Cursors in SQL Server

The following are the different types of cursors in SQL Server listed below:

- **Static Cursors** - The result set shown by the static cursor is always the same as when the cursor was first opened. We can use the static cursor to move both forward and backward. In contrast to other cursors, it is slower and consumes more memory. As a result, we can use it only when scrolling is necessary, and other cursors aren't suitable.

- **Dynamic Cursors -** The dynamic cursors are opposite to the static cursors that allow us to perform the data updation, deletion, and insertion operations while the cursor is open. It is scrollable by default. It can detect all changes made to the rows, order, and values in the result set, whether the changes occur inside the cursor or outside the cursor.
- **Forward-Only Cursors** - It is the default and fastest cursor type among all cursors. It is called a forward-only cursor because it moves only forward through the result set. This cursor doesn't support scrolling. It can only retrieve rows from the beginning to the end of the result set. It allows us to perform insert, update, and delete operations.
- **Keyset Cursors -** This cursor functionality lies between a static and a dynamic cursor regarding its ability to detect changes. It can detect changes in the result set's rows values as like a dynamic cursor. It can only move from the first to last and last to the first row.

***(35). Explain the concept of Triggers, Procedures, Packages.***

# Triggers

A trigger is a set of SQL statements that reside in system memory with unique names. It is a specialized category of stored procedure that is called automatically when a database server event occurs. Each trigger is always associated with a table.

A trigger is called a special procedure because it cannot be called directly like a stored procedure. The key distinction between the trigger and procedure is that a trigger is called automatically when a data

Triggers will be helpful when we need to execute some events automatically on certain desirable scenarios.

# Procedure

The PL/SQL stored procedure or simply a procedure is a PL/SQL block which performs one or more specific tasks. It is just like procedures in other programming languages.

The procedure contains a header and a body.

- **Header:** The header contains the name of the procedure and the parameters or variables passed to the procedure.
- **Body:** The body contains a declaration section, execution section and exception section similar to a general PL/SQL block.

## Packages

Packages are schema objects that groups logically related PL/SQL types, variables, and subprograms.

A package will have two mandatory parts −

- Package specification - The specification is the interface to the package. It just DECLARES the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package. In other words, it contains all information about the content of the package, but excludes the code for the subprograms.
- Package body or definition - has the codes for various methods declared in the package specification and other private declarations, which are hidden from the code outside the package