Q.1 What is Normalization? List The different types of them.

“Normalization is the process to convert the complex database into normal form”

Types of them
- 1NF (Normal Form)
- 2NF
- 3NF
- BCNF (Boyce Code Normal Form)
- More NF

Q.2 Explain 1NF with example.

- In 1NF there are no any composite attribute or all attribute have scalar value.
- Example For Student Record
- Student(Enr_no, Name, Address, Branch, Mobile)

<table>
<thead>
<tr>
<th>Enr_no</th>
<th>Name</th>
<th>Address</th>
<th>Branch</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>166550311001</td>
<td>Ronika Patel</td>
<td>20 Shiv Nagar, Ta Xyz State MP</td>
<td>EC</td>
<td>8986885588</td>
</tr>
<tr>
<td>166550307002</td>
<td>Naavi Patel</td>
<td>Vankal Mokha, Ta Chikhli State Gujarat</td>
<td>Computer</td>
<td>7359539584</td>
</tr>
</tbody>
</table>
In above example there are name have part first name and surname and address have part address, taluka, dist, state so convert them in to 1NF by below way.

Student(Enr_no, First_name, Surname, Address, City, State, Branch, Mobile)

1NF of Student Record

<table>
<thead>
<tr>
<th>Enr_no</th>
<th>First_name</th>
<th>Surname</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>branch</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>166550311001</td>
<td>Ronika</td>
<td>Patel</td>
<td>20 Shiv Nagar,</td>
<td>Xyz</td>
<td>MP</td>
<td>EC</td>
<td>8986885588</td>
</tr>
<tr>
<td>166550307002</td>
<td>Naavi</td>
<td>Patel</td>
<td>Vankal Mokha</td>
<td>Chikhli</td>
<td>Gujara t</td>
<td>Computer</td>
<td>7359539584</td>
</tr>
</tbody>
</table>

Q.3 Explain 2NF with example.

- It must be in 1NF.
- There are all none prime attribute depend on prime attribute.
- In the 2NF there are other field depend or find by Primary Key.
- So all Composite table must have primary key.
- Example For Student Record

Student(Enr_no, First_name, Surname, Address, City, State, Branch, Mobile)

<table>
<thead>
<tr>
<th>Enr_no</th>
<th>First_name</th>
<th>Surname</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>branch</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>166550311001</td>
<td>Ronika</td>
<td>Patel</td>
<td>20 Shiv Nagar,</td>
<td>Xyz</td>
<td>MP</td>
<td>EC</td>
<td>8986885588</td>
</tr>
<tr>
<td>166550307002</td>
<td>Naavi</td>
<td>Patel</td>
<td>Vankal Mokha</td>
<td>Chikhli</td>
<td>Gujara t</td>
<td>Computer</td>
<td>7359539584</td>
</tr>
</tbody>
</table>

In above example the student record divied in table where Enr_no is primary key.

Student info

Enr_no <- First_name, Surname, Address, City, State, Mobile

Student admission

Enr_no <- Branch
2NF of Student Record

Student_Info

<table>
<thead>
<tr>
<th>Enr_no</th>
<th>First_name</th>
<th>Surname</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>16655031100</td>
<td>Ronika</td>
<td>Patel</td>
<td>20 Shiv Nagar,</td>
<td>Xyz</td>
<td>MP</td>
<td>898688558</td>
</tr>
<tr>
<td>16655030700</td>
<td>Naavi</td>
<td>Patel</td>
<td>Vankal Mokha</td>
<td>Chikhli</td>
<td>Gujara</td>
<td>735953958</td>
</tr>
</tbody>
</table>

Student_Admission

<table>
<thead>
<tr>
<th>Enr_no</th>
<th>Branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>16655031100</td>
<td>EC</td>
</tr>
<tr>
<td>16655030700</td>
<td>Computer</td>
</tr>
</tbody>
</table>

Q.4 Explain 3NF with example.

- It must be in 1NF and 2NF.
- There are all none prime attribute functionally depend on candidate key.
- Some time we can not find the uniq record using primary key at that time we have to take one more field for find the uniq record this fiels is known as a candidate key.
- Example For Student Recortd
  Student(Enr_no, First_name, Surname, Address, City, State, Branch, Mobile)
<table>
<thead>
<tr>
<th>Enr_no</th>
<th>First_name</th>
<th>Surname</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>branch</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>16655031100</td>
<td>Ronika</td>
<td>Patel</td>
<td>20 Shiv Nagar,</td>
<td>Xyz</td>
<td>Gujarat</td>
<td>EC</td>
<td>8986885588</td>
</tr>
<tr>
<td>16655030700</td>
<td>Naavi</td>
<td>Patel</td>
<td>Vankal Mokha</td>
<td>Chikhli</td>
<td>Gujarat</td>
<td>Computer</td>
<td>7359539584</td>
</tr>
<tr>
<td>16655030700</td>
<td>Monika</td>
<td>Patel</td>
<td>Xyz</td>
<td>Pune</td>
<td>Maha</td>
<td>Computer</td>
<td>7858858855</td>
</tr>
</tbody>
</table>

In above example the student record there are chances of same enr_no in diriment state. So we have to take state with them to find uniq record Student (Enr_no,State) <- First_name,Surname, Address,City, Branch, Mobile

State Table

<table>
<thead>
<tr>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP</td>
</tr>
<tr>
<td>Gujarat</td>
</tr>
</tbody>
</table>

Select state first then enter the enr_no

Q.5 Explain Functional Dependency

- Functional dependency (FD) is a set of constraints between two attributes in a relation.
- Functional dependency says that if two tuples have same values for attributes A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.
- Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y.
- The left-hand side attributes determine the values of attributes on the right-hand side.
Q.6 Short on Armstrong's Axioms.

OR

Explain three rules of Armstrong's Axioms

- Reflexive rule – If alpha is a set of attributes and beta is subset of alpha, then alpha holds beta.
- Augmentation rule – If \( a \rightarrow b \) holds \( y \) is attribute set, then \( ay \rightarrow by \) also holds.
- Transitivity rule – If \( a \rightarrow b \) holds and \( b \rightarrow c \) holds, then \( a \rightarrow c \) also holds.

Q.7 Explain Trivial Functional Dependency and Non-trivial Functional Dependency.

- Trivial – If a functional dependency (FD) \( X \rightarrow Y \) holds, where \( Y \) is a subset of \( X \), then it is called a trivial FD. Trivial FDs always hold.
- Non-trivial – If an FD \( X \rightarrow Y \) holds, where \( Y \) is not a subset of \( X \), then it is called a non-trivial FD.
- Completely non-trivial – If an FD \( X \rightarrow Y \) holds, where \( x \) intersect \( Y = \Phi \), it is said to be a completely non-trivial FD.

Q.8 Explain Generalization.

- A number of entities with same characteristics are together into one generalized entity is called generalization.
  - For example sparrow, crow and dove can all be generalized as Birds.

Q.5 Explain Specialization.

- Specialization is the opposite of generalization.
o In specialization, a group of entities is divided into sub-groups based on their characteristics.

o Take a group ‘Person’ for example. A person has name, date of birth, gender, etc. if he/she is teacher or student.

Q.9 Explain abstraction.

o In ER-Model create classes of objects in object-oriented programming. The details of entities are generally hidden from the user this process known as abstraction.

Q.10 Explain Inheritance.

• Inheritance is an important feature of Generalization and Specialization.

• It allows lower-level entities to inherit the attributes of higher-level entities.

• For example, the attributes of a Person class such as name, age, and gender can be inherited by lower-level entities such as Student or Teacher.
Q.11 What is decomposition?

- Decomposition is the process of breaking down in parts or elements.
- It replaces a relation with a collection of smaller relations.
- It breaks the table into multiple tables in a database.
- Properties of Decomposition
  - Lossless Decomposition
  - Dependency Preservation
  - Lack of Data Redundancy
Q.12 Explain Schedule and Serial Schedule.

- **Schedule**
  - A execution sequence of a transaction is called a schedule.
  - A schedule can have many transactions each have a number of instructions/tasks.

- **Serial Schedule**
  - It is a schedule in which transactions enter next transaction if first is completed. Transactions are ordered one after the other. This type of schedule is called a serial schedule.
  - When the first transaction completes its cycle, then the next transaction is executed.

Q.13 DCL command with example.

- Data Control Language (DCL) is used to control privilege in Database.
- It perform any operation in the database, such as for creating tables, sequences or views.
- Privileges are of two types,
  - System : creating session, table etc.
  - Object : any command or query.
- DCL defines two commands,
  - Grant : Gives user access to database.
  - Revoke : Take back permissions from user.
- Example:
  1. To Allow a User to create Table
     - SQL:>grant create table to username;
  2. To Grant all privilege to a User
     - SQL:> grant sysdba to username
3. To Grant permission to Create any Table
   - SQL:>grant create any table to username

4. To Grant permission to Drop any Table
   - SQL:>grant drop any table to username

5. To take back Permissions
   - SQL:>revoke create table from username

Q.14 TCL command with example.

- Transaction Control Language(TCL) commands are used to manage transactions in database.
- These are used to manage the changes made by DML statements.
- It also allows statements to be grouped together into logical transactions.

Example

- Commit command
  
  Commit command is used to permanently save any transaction into database.
  
  SQL:> commit;

- Rollback command

  This command restores the database to last committed state.
  
  SQL:> rollback to savepoint-name;

- Savepoint command

  Savepoint command is used to temporarily save a transaction.
  
  SQL:> savepoint to savepoint-name;
Q.15 Explain View.

- A view is a “virtual” table that is derived from other tables
- Allows for limited update operations
- Since the table may not physically be stored
- Allows full query operations

Syntax for creating a View,

```sql
SQL:> CREATE view view_name AS
    SELECT column_name(s)
    FROM table_name
    WHERE condition
```

- Syntax for creating a View,

```sql
SQL:>CREATE view XYZ AS
    SELECT Name
    FROM Student
    WHERE rollno=5;
SQL:>Select *from XYZ;
```

Q.16 Explain View types. OR Different Between Simple view and Complex View.

There are two types of view,

- Simple View
- Complex View
### Q.17 Explain INDEX.

- Indexes are pointers that map the physical address of data.
- Using indexes data manipulation becomes faster.

### Q.18 Explain Transaction.

- A transaction is a unit of program execution that access and updates various data items.
- When the transaction is committed, the database must be consistent.
- Two main issues to deal with:
  - Failures of various kinds, such as hardware failures and system crashes
  - Concurrent execution of multiple transactions
- Transaction to transfer $50 from account A to account B:
  1. read(A)
  2. A := A – 50
  3. write(A)
  4. read(B)
5. \( B := B + 50 \)
6. \( \text{write}(B) \)

Q.19 Explain ACID Properties of transaction.

- **Atomicity.**
  - Either all operations of the transaction are properly reflected in the database or none are.
- **Consistency**
  - Execution of a transaction in isolation.
- **Isolation.**
  - Multiple transactions may execute concurrently
  - Each transaction must be unaware of other concurrently executing transactions.
  - Intermediate transaction results must be hidden from other concurrently executed transactions.
- **Durability.**
  - After a transaction completes successfully, there must be the changes in the database even if there are system failures.
Q.20 Explain Transaction State.

- **Active**, 
  - It initial state
  - The transaction stays in this state while it is executing
- **Partially committed**
  - Before the final statement it executes.
  - There are two possibility
    - Committed
    - Fail
- **Failed**
  - Transaction not success.
- **Aborted**
  - After the transaction has been rolled back database restored using this state.
Two options after it has been aborted:

- restart the transaction
- kill the transaction

Committed

After successful completion.

Q.21 Explain Lock with example.

A lock is a variable associated with a data item that determines whether read/write operations can be performed on that data item.

Locking-based concurrency control systems can use either one-phase or two-phase locking protocols.

One-phase Locking Protocol

- In this method, each transaction locks an item before use and releases the lock when finished.
- This locking method provides for maximum concurrency but does not always enforce serializability.

Two-phase Locking Protocol

- In this method, the transaction have two phases.
  - In the first phase, a transaction only acquires all the locks it needs and do not release any lock. It is also the growing phase.
  - In the second phase, the transaction releases the locks and cannot request any new locks. This is called the shrinking phase.
Q.22 Explain Timestamp Concurrency Control Algorithms.

- Timestamp-based concurrency control algorithms use a transaction’s timestamp to coordinate concurrent access to a data item to ensure serializability.

- A timestamp is a unique identifier given by DBMS to a transaction that represents the transaction’s start time.

- It have follow three rules
  
  o Access Rule
    
    ▪ When two transactions try to access the same data item, priority is given to the older transaction.
    
    ▪ This causes the younger transaction to wait for the older transaction to commit first.

  o Late Transaction Rule
    
    ▪ If a younger transaction has written a data item, then an older transaction is not allowed to read or write that data item.

  o Younger Transaction Rule –
    
    ▪ A younger transaction can read or write a data item that has already been written by an older transaction.

Q.23 Explain Optimistic Concurrency Control Algorithm

In this approach, a transaction’s life cycle is divided into the following three phases –

- Execution Phase
  
  ▪ A transaction fetches data items to memory and performs operations upon them.

- Validation Phase
  
  ▪ A transaction performs checks to ensure that committing
• Commit Phase

  o A transaction writes modified data item in memory to the disk.

**Q.24 Explain Function And PROCEDURES.**

- A procedure is a module performing one or more actions, it does not need to return any values.
- A stored procedure contains a sequence of SQL commands stored in the database catalog so that it can be invoked later by a program.
- Functions are a type of stored code and are very similar to procedures but function is a PL/SQL block that returns a single value.

**Q.25 Difference between Stored Procedure and Function**

- Function must return a value but in Stored Procedure it is optional (Procedure can return zero or n values).
- Functions can have only input parameters for it whereas Procedures can have input/output parameters.
- Functions can be called from Procedure whereas Procedures cannot be called from Function.

**Q.26 Explain Store PROCEDURES with example.**

- A procedure is a module performing one or more actions, it does not need to return any values.
- A stored procedure contains a sequence of SQL commands stored in the database catalog so that it can be invoked later by a program.
- Stored procedures are declared using the following syntax:

  ```sql
  Create Procedure <proc-name>
  (param_spec1, param_spec2, ..., param_specn )
  begin
  -- execution code
  end;
  ```
Q.27 Explain Trigger with example.

➢ Triggers are stored programs, which are automatically executed when some events occur.

➢ Triggers have following events –

   A database manipulation (DML) statement (DELETE, INSERT, or UPDATE)

   A database definition (DDL) statement (CREATE, ALTER, or DROP).

   A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

➢ Benefits of Triggers

   • Generating some derived column values automatically

   • Event logging and storing information on table access

   • Auditing

   • Synchronous read of tables

   • security and authorizations

   • Preventing invalid transactions

Creating Triggers

The syntax for creating a trigger is –

CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF}
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;

CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
  sal_diff number;
BEGIN
  sal_diff := :NEW.salary - :OLD.salary;
  dbms_output.put_line('Old salary: ' || :OLD.salary);
  dbms_output.put_line('New salary: ' || :NEW.salary);
  dbms_output.put_line('Salary difference: ' || sal_diff);
END;
Q.28 Explain Cursor with example.

- Oracle creates a memory area, known as the context area. Cursor is pointer to control this context area.
- There are two types of cursors –
  - Implicit cursors
  - Explicit cursors (Parameter Cursor)

- Implicit Cursors
  - Implicit cursors are automatically created by Oracle whenever an SQL statement is executed.
  - Programmers cannot control the implicit cursors and the information in it.
  - Insert, Update and Delete statement used this cursor.
- Explicit Cursors
  - Explicit cursors are programmer-defined cursors for gaining more control over the context area.
  - An explicit cursor should be defined in the declaration section of the PL/SQL Block.
  - It is created on a SELECT Statement which returns more than one row.

Q.29 PL/SQL data Type.

- CHARACTER Data type
- NUMBER Data type
- BOOLEAN Data type
- DATE Data type
- LOB Data type
CHARACTER Data type:
This data type basically stores alphanumerical characters in string format.

- CHAR Data type (fixed string size)
- VARCHAR2 Data type (variable string size)
- VARCHAR Data type
- NCHAR (native fixed string size)
- NVARCHAR2 (native variable string size)
- LONG and LONG RAW

NUMBER Data type:
This data type stores fixed or floating point numbers up to 38 digit.

Syntax for declaration:

```
A  NUMBER (8,2);
B  NUMBER (8)
C  NUMBER;
```

BOOLEAN Data type:
This data type stores the logical values. It represents either TRUE or FALSE

Syntax for declaration:

```
Var1  BOOLEAN;
```

DATE Data type:
This data type stores the values in date format, as date, month, and year.

Syntax for declaration:

```
newyear  DATE := '01-JAN-2015';
current_date  DATE := SYSDATE;
```
LOB Data type:
This data type is mainly used to store and manipulate large blocks of unstructured data's like images, multimedia files, etc.

- BLOB
- CLOB and NCLOB
- BFILE

Q.30 Explain Concurrency control

- Concurrency control (CC) is a process to ensure that data is updated correctly when multiple transactions are concurrently executed in DBMS.

Q.31 Explain Dead Lock.

- A deadlock is a condition where in two or more tasks are waiting for each other in order to be finished but none of the task is give up the resources that other task needs.

Conditions for deadlocks

- Mutual exclusion. No resource can be shared by more than one process at a time.
- Hold and wait. There must exist a process that is holding at least one resource and is waiting to resources that are currently being held by other processes.
- No preemption. A resource cannot be preempted.
- Circular wait. There is a cycle in the wait-for graph.

Two types of deadlocks

- Resource deadlock: uses AND condition.
  
  AND condition: a process that requires resources for execution can proceed when it has acquired all those resources.

- Communication deadlock: uses OR condition.
OR condition: a process that requires resources for execution can proceed when it has acquired at least one of those resources

**Deadlock prevention (Control)**

- Removing mutual exclusion:
- Removing hold and wait condition:
- Preemption of resources:
- Avoid circular wait condition:

Q.32 Explain Lossy Decomposition with example.

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

- One of the disadvantages is that there are two or more relational schemes (or tables) so lost the information of original relation or table.

- Example

<table>
<thead>
<tr>
<th>Roll_no</th>
<th>Sname</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>parimal</td>
<td>COMPUTER</td>
</tr>
<tr>
<td>222</td>
<td>parimal</td>
<td>ELECTRICAL</td>
</tr>
</tbody>
</table>

This relation is decomposed into two relation stu_name and name_dept:

stu_name:
### Roll_no | Sname
--- | ---
111 | parimal
222 | parimal

### Name_dept :

<table>
<thead>
<tr>
<th>Sname</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>parimal</td>
<td>COMPUTER</td>
</tr>
<tr>
<td>parimal</td>
<td>ELECTRICAL</td>
</tr>
</tbody>
</table>

### stu_joined :

<table>
<thead>
<tr>
<th>Roll_no</th>
<th>Sname</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>parimal</td>
<td>COMPUTER</td>
</tr>
<tr>
<td>111</td>
<td>parimal</td>
<td>ELECTRICAL</td>
</tr>
<tr>
<td>222</td>
<td>parimal</td>
<td>COMPUTER</td>
</tr>
<tr>
<td>222</td>
<td>parimal</td>
<td>ELECTRICAL</td>
</tr>
</tbody>
</table>
Q.33 Explain Lossless Join Decomposition with example.

- "The decomposition of relation R into R1 and R2 is lossless when the join of R1 and R2 in 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 original table by joining the decomposed parts. This is called lossless-join (or non-additive join) decomposition.
- This is also referred as non-additive decomposition.

**STUDENT:**

<table>
<thead>
<tr>
<th>Roll_no</th>
<th>Sname</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>parimal</td>
<td>COMPUTER</td>
</tr>
<tr>
<td>222</td>
<td>parimal</td>
<td>ELECTRICAL</td>
</tr>
</tbody>
</table>

This relation is decomposed into two relation Stu_name and Stu_dept:

**Stu_name:**

<table>
<thead>
<tr>
<th>Roll_no</th>
<th>Sname</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>parimal</td>
</tr>
<tr>
<td>222</td>
<td>parimal</td>
</tr>
</tbody>
</table>
Q.34 Explain Dependency-Preserving Decomposition with example.

- The dependency preservation decomposition is another property of decomposed relational database schema.
- In them each functional dependency $X \rightarrow Y$ specified By $F$
- If $F$ is either appeared directly in one of the relation schemas $R_i$ in the decomposed $D$ or could be indirect appear in some $R_i$. 

<table>
<thead>
<tr>
<th>Roll_no</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>COMPUTER</td>
</tr>
<tr>
<td>222</td>
<td>ELECTRICAL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Roll_no</th>
<th>Sname</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>parimal</td>
<td>COMPUTER</td>
</tr>
<tr>
<td>222</td>
<td>parimal</td>
<td>ELECTRICAL</td>
</tr>
</tbody>
</table>
Decomposition $D = \{ R_1, R_2, R_3, \ldots, R_m \}$ of $R$ is said to be dependency-preserving with respect to $F$ if the union of the projections of $F$ on each $R_i$, in $D$ is equivalent to $F$.

If decomposition is not dependency-preserving, some dependency is lost in the decomposition.

Example:

Let a relation $R(A,B,C,D)$ and set a FDs $F = \{ A \rightarrow B \, , \, A \rightarrow C \, , \, C \rightarrow D \}$ are given.

A relation $R$ is decomposed into:

$R_1 = (A, B, C)$ with FDs $F_1 = \{A \rightarrow B, A \rightarrow C\}$, and

$R_2 = (C, D)$ with FDs $F_2 = \{C \rightarrow D\}$.

$F' = F_1 \cup F_2 = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$

so, $F' = F$.

And so, $F'^+ = F^+$.

Q.35 Explain Explicit Lock and Implicit Lock.

Q.36 Explain Sequence with example.

Q.37 Store procedure Example.

Q.38 Explain GOTO.