Answer Key									
Internal Assessment Test II – AUGUST 2022									
Sub:	**DATABASE MANAGEMENT SYSTEMS**							**Sub Code:**	**20MCA21**
Date:	**29-08-2022**	Duration:	90 mins	Max Marks:	50	**Sem:**	**II**	**Branch:**	**MCA**

CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test II – August 2022

Sub:	**Database Management System**							**Sub Code:**	**20MCA21**
Date:	**29-08-2022**	**Duration:**	**90 min's**	**Max Marks:**	**50**	**Sem:**	**II**	**Branch:**	**MCA**

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

	PART I	MARKS	OBE	
			CO	RBT
1	Discuss the different relational algebra operations. **OR**	[10]	CO2	L1
2	What are the different notations used in ER diagram. Explain with example of company database.	[10]	CO2	L2
	PART II			
3	Explain any four different Keys used in Relational Database with example. **OR**	[10]	CO2	L1
4	Explain the main phases of database design with a proper diagram.	[10]	CO2	L1
	PART III			
5	Explain the following i)Unary Relationship ii) Recursive Relationship iii) binary Relationship iv)Ternary Relationship OR	[10]	CO2	L1
6	Define an entity and an attribute. Explain the different types of attributes that occur in the ER model, with an example.	[10]	CO2	L2
	PART IV			
7	Explain functional dependency with different IR rules. OR	[10]	CO2	L2
8	Define the following. I)Null values ii)Composite Attribute iii) Multivalued Attribute, iv) Weak Entity	[10]	CO2	L1

9	**PART V** Define Normalization. Explain 1NF, 2NF and 3NF with examples. OR	[10]	CO5	L2
10	Given Relation R(P, Q, R, S, T) and FD set ={PQ → R,S → T }determine whether the given R is in 2NF? If not convert it into 2 NF.	[10]	CO5	L3

1. Discuss the different relational algebra operations.

Unary Relational Operations

- SELECT (symbol: σ)
- PROJECT (symbol: π)
- RENAME (symbol: ρ)

Relational Algebra Operations From Set Theory

- UNION (υ)
- INTERSECTION ( ),
- DIFFERENCE (-)
- CARTESIAN PRODUCT (x)

Binary Relational Operations

- JOIN
- DIVISION

SELECT (σ)

The SELECT operation is used for selecting a subset of the tuples according to a given selection condition. Sigma(σ)Symbol denotes it. It is used as an expression to choose tuples which meet the selection condition. Select operator selects tuples that satisfy a given predicate.Projection(π)

The projection eliminates all attributes of the input relation but those mentioned in the projection list. The projection method defines a relation that contains a vertical subset of Relation.

This helps to extract the values of specified attributes to eliminates duplicate values. (pi) symbol is used to choose attributes from a relation. This operator helps you to keep specific columns from a relation and discards the other columns.Rename (ρ)

Rename is a unary operation used for renaming attributes of a relation.

ρ (a/b)R will rename the attribute ‘b’ of relation by ‘a’.

Operation(Symbols)	Purpose
Select(σ)	The SELECT operation is used for selecting a subset of the tuples according to a given selection condition
Projection(π)	The projection eliminates all attributes of the input relation but those mentioned in the projection list.
Union Operation(∪)	UNION is symbolized by symbol. It includes all tuples that are in tables A or in B.
Set Difference(-)	– Symbol denotes it. The result of A – B, is a relation which includes all tuples that are in A but not in B.
Intersection(∩)	Intersection defines a relation consisting of a set of all tuple that are in both A and B.
Cartesian Product(X)	Cartesian operation is helpful to merge columns from two relations.
Inner Join	Inner join, includes only those tuples that satisfy the matching criteria.
Theta Join(θ)	The general case of JOIN operation is called a Theta join. It is denoted by symbol θ.
EQUI Join	When a theta join uses only equivalence condition, it becomes a equi join.
Natural Join(⋈)	Natural join can only be performed if there is a common attribute (column) between the relations.
Outer Join	In an outer join, along with tuples that satisfy the matching criteria.
Left Outer Join(⟕)	In the left outer join, operation allows keeping all tuple in the left relation.
Right Outer join(⟖)	In the right outer join, operation allows keeping all tuple in the right relation.
Full Outer Join(⟗)	In a full outer join, all tuples from both relations are included in the result irrespective of the matching condition.

2. What are the different notations used in ER diagram. Explain with example of company database.

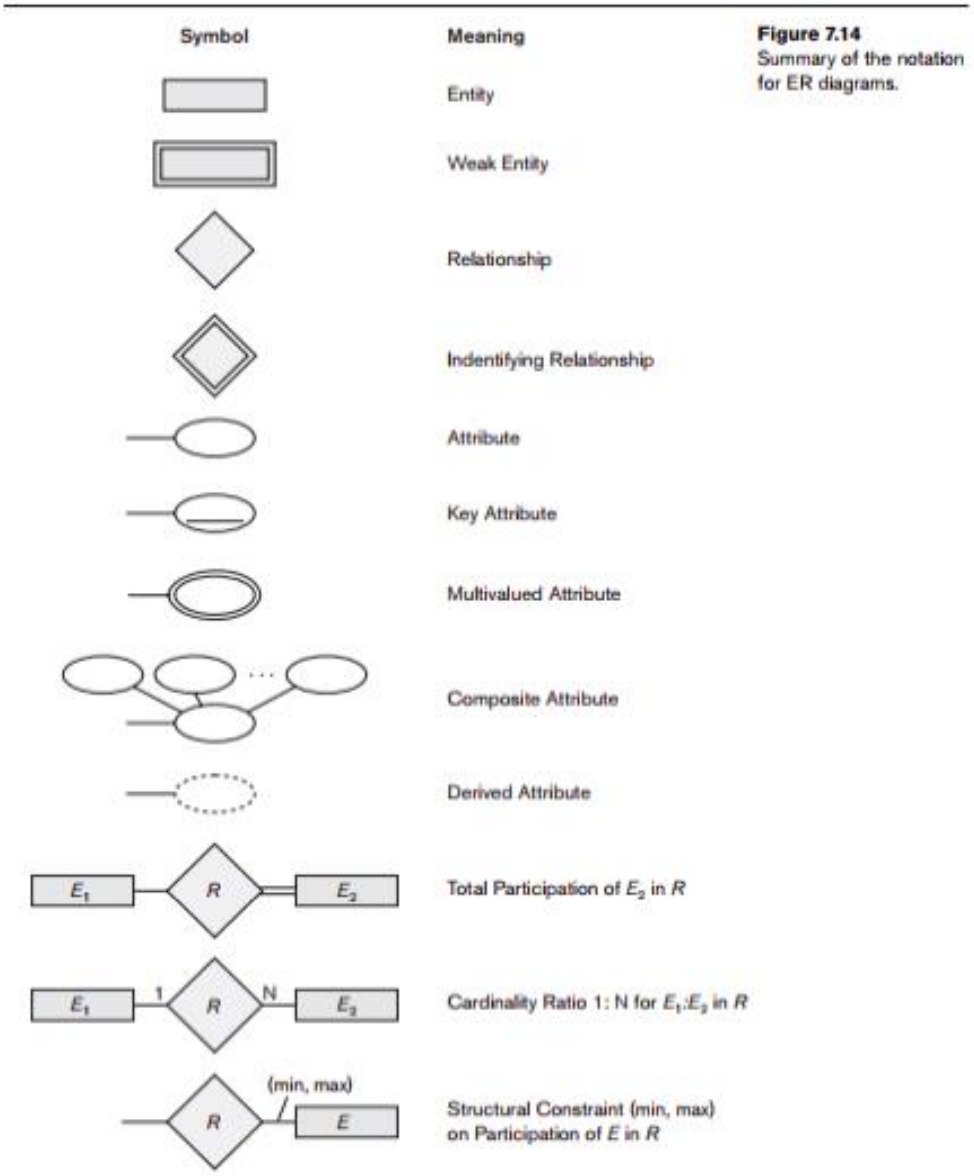


Figure 7.14
Summary of the notation for ER diagrams.

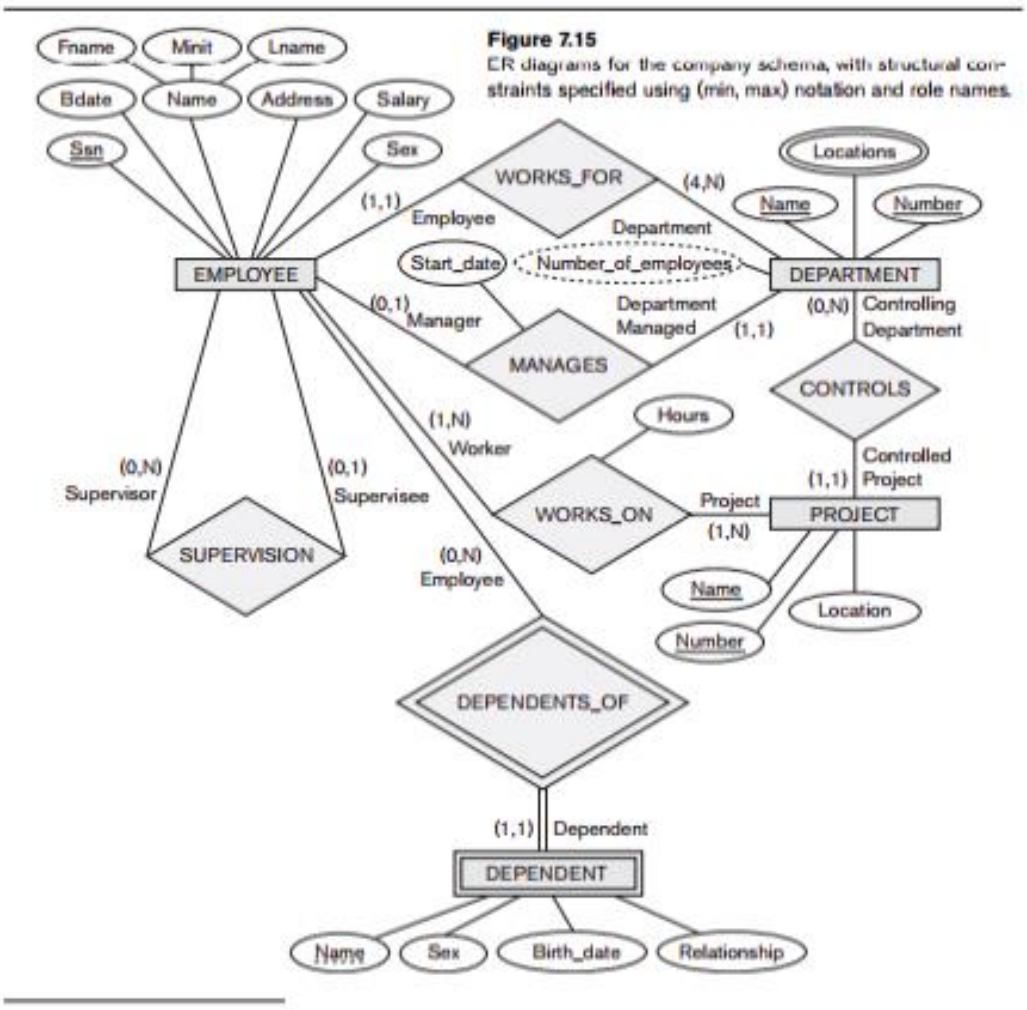


Figure 7.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

3. Explain any four different Keys used in Relational Database with example.

Keys

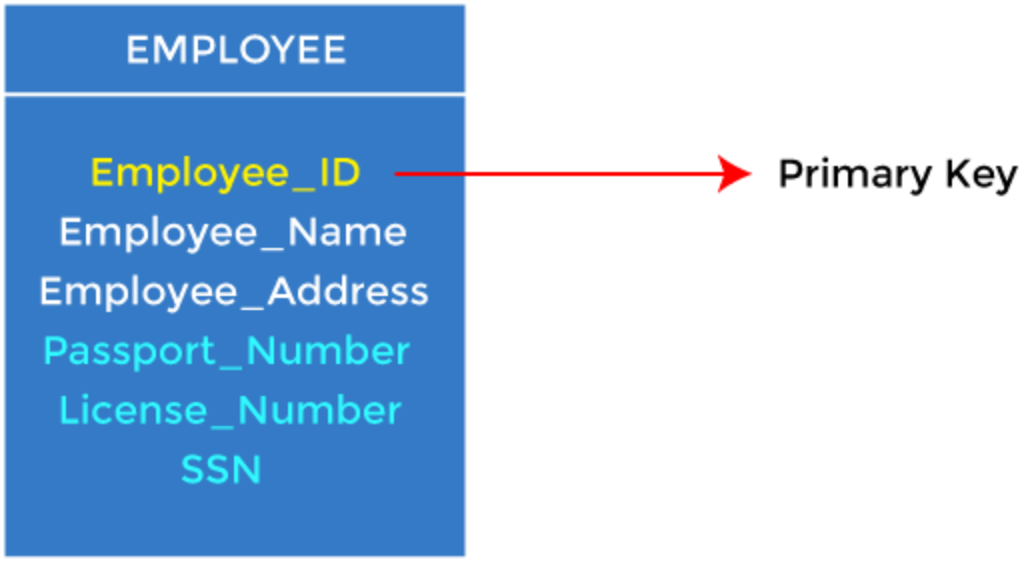
- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

The different types of keys in DBMS are −

- **Candidate Key -** The candidate keys in a table are defined as the set of keys that is minimal and can uniquely identify any data row in the table.
- **Primary Key -** The primary key is selected from one of the candidate keys and becomes the identifying key of a table. It can uniquely identify any data row of the table.
- **Super Key -** Super Key is the superset of primary key. The super key contains a set of attributes, including the primary key, which can uniquely identify any data row in the table.
- **Composite Key -** If any single attribute of a table is not capable of being the key i.e it cannot identify a row uniquely, then we combine two or more attributes to form a key. This is known as a composite key.
- **Secondary Key -** Only one of the candidate keys is selected as the primary key. The rest of them are known as secondary keys.
- **Foreign Key -** A foreign key is an attribute value in a table that acts as the primary key in another table. Hence, the foreign key is useful in linking together two tables. Data should be entered in the foreign key column with great care, as wrongly entered data can invalidate the relationship between the two tables.

Primary key

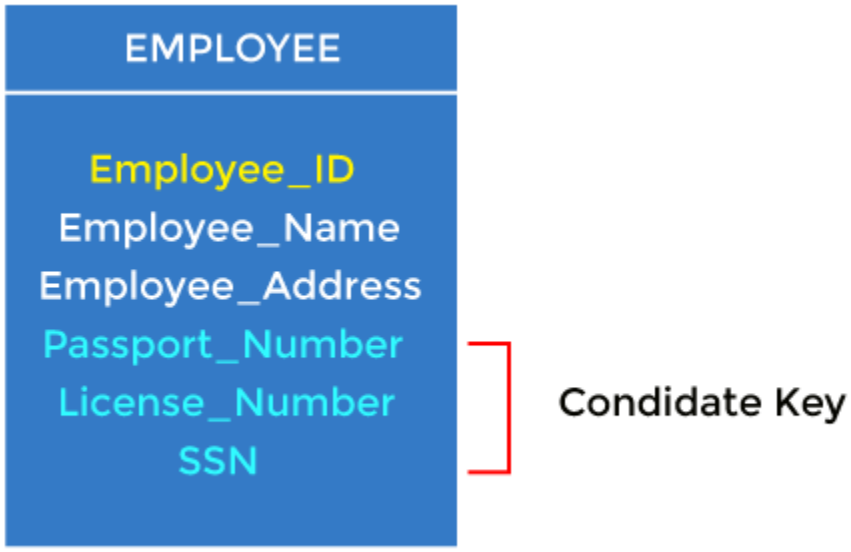
- It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.
- In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.
- For each entity, the primary key selection is based on requirements and developers.



2. Candidate key

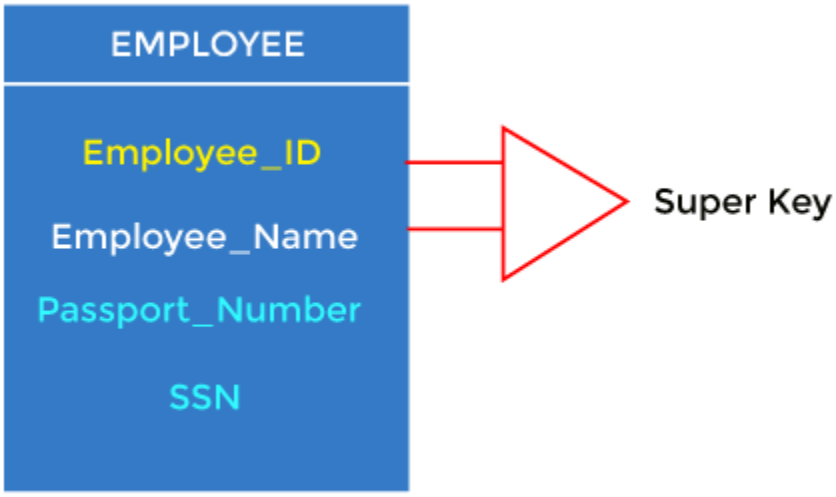
- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



3. Super Key

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.

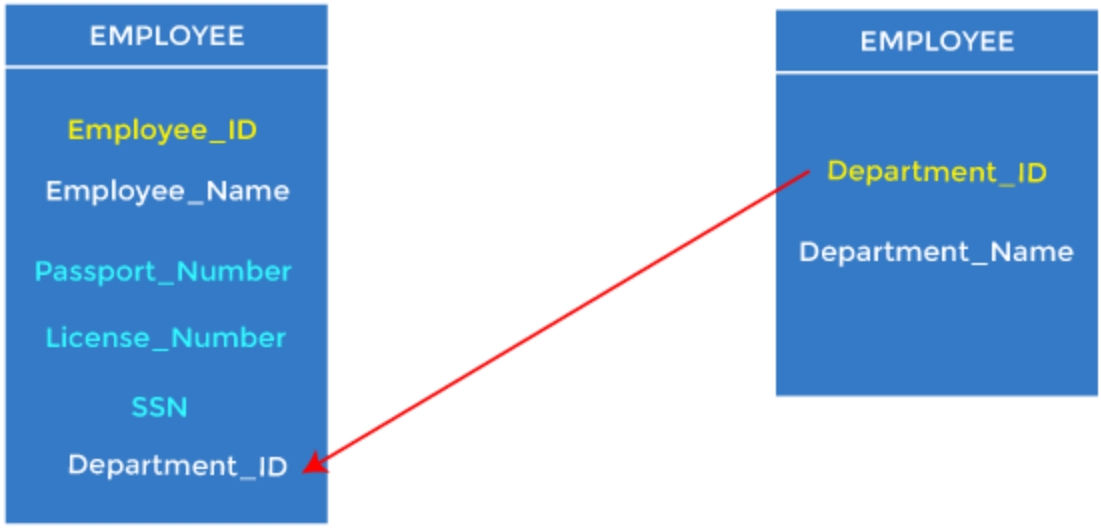


For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

4. Foreign key

- Foreign keys are the column of the table used to point to the primary key of another table.
- Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.
- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.

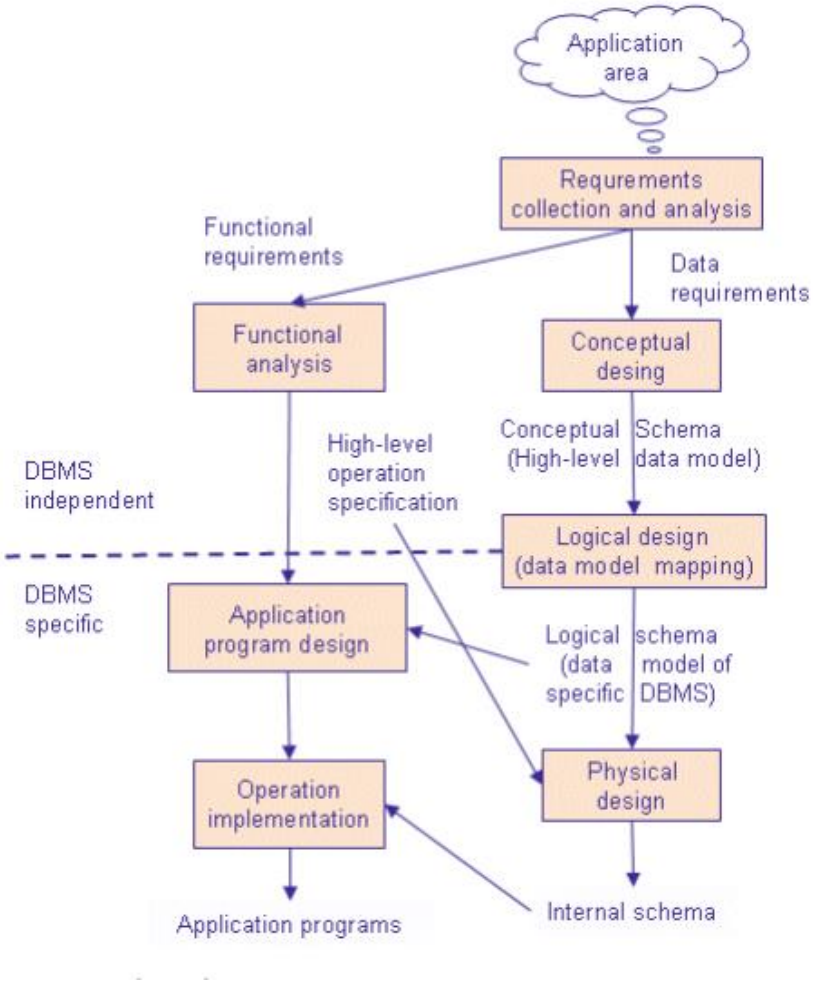


Difference Between Primary key & Foreign key

Following is the main difference between primary key and foreign key:

Primary Key	Foreign Key
Helps you to uniquely identify a record in the table.	It is a field in the table that is the primary key of another table.
Primary Key never accept null values.	A foreign key may accept multiple null values.
Primary key is a clustered index and data in the DBMS table are physically organized in the sequence of the clustered index.	A foreign key cannot automatically create an index, clustered or non-clustered. However, you can manually create an index on the foreign key.
You can have the single Primary key in a table.	You can have multiple foreign keys in a table.

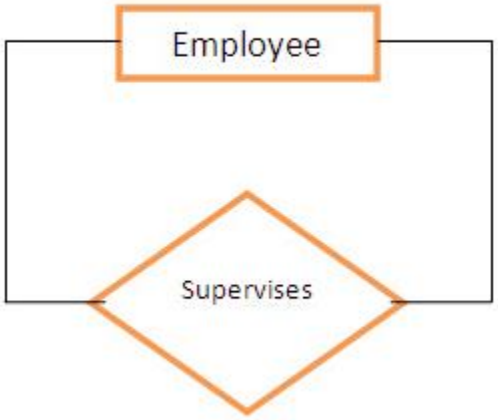
4. Explain the main phases of database design with a proper diagram.



5. Explain the following i) Unary Relationship ii) Recursive Relationship iii) binary Relationship iv)Ternary Relationship

i) Unary Relationship

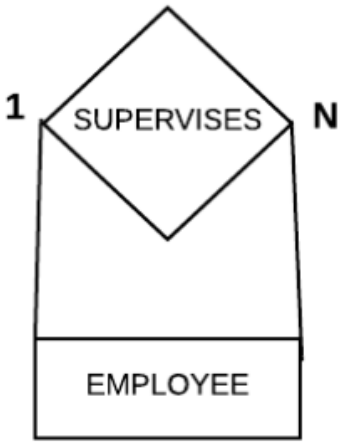
A unary relationship, also called recursive, is one in which a relationship exists between occurrences of the same entity set.



ii) Recursive Relationship

When there is a relationship between two entities of the same type, it is known as a recursive relationship. This means that the relationship is between different instances of the same entity type.

Examples of recursive relationship can be shown as follows −

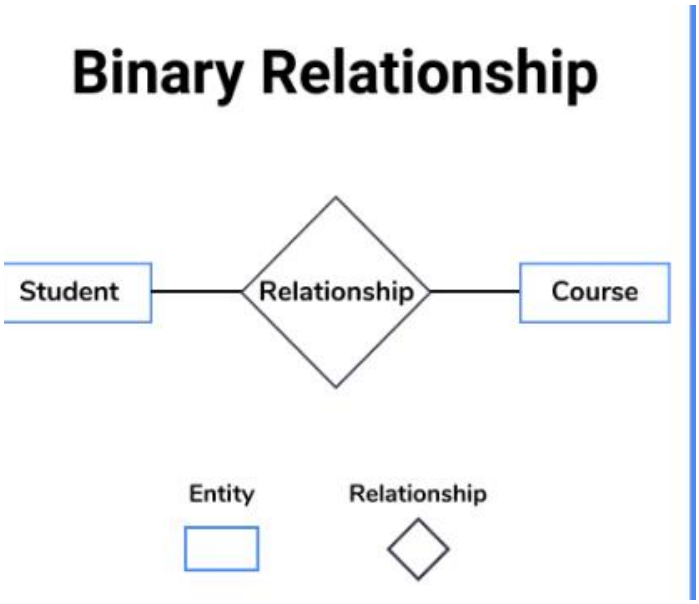


An employee can supervise multiple employees. Hence, this is a recursive relationship of entity employee with itself. This is a 1 to many recursive relationship as one employee supervises many employees.

iii)

Binary Relationship

When there are exactly two entity sets participating in a relationship then such type of relationship is called binary relationship.

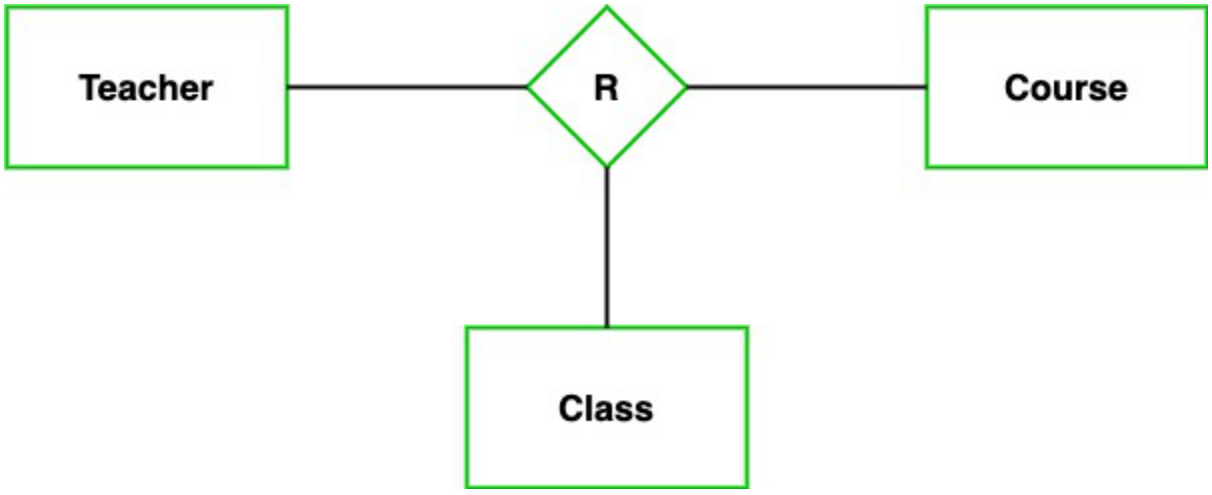


iv)Ternary Relationship

Ternary (degree 3)

In the Ternary relationship, there are three types of entity associates. So, we can say that a Ternary relationship exists when there are three types of entity and we call them a degree of relationship is 3. Since the number of entities increases due to this, it becomes very complex to turn E-R into a relational table. Now let's understand with the examples.

Example: We have three entity types 'Teacher', 'Course', and 'Class'. The relationship between these entities is defined as the teacher teaching a particular course, also the teacher teaches a particular class. So, here three entity types are associating we can say it is a ternary relationship.



6. Define an entity and an attribute. Explain the different types of attributes that occur in the ER model, with an example.

Entity and Attribute: Entity and Attributes are two essential terms of a database management system (DBMS). The main difference between the Entity and attribute is that an entity is a real-world object, and attributes describe the properties of an Entity.

An attribute is a property or characteristic of an entity. An entity may contain any number of attributes. One of the attributes is considered as the primary key. In an Entity-Relation model, attributes are represented in an elliptical shape.

***Example*:** Student has attributes like name, age, roll number, and many more. To uniquely identify the student, we use the primary key as a roll number as it is not repeated. Attributes can also be subdivided into another set of attributes.
There are five such types of attributes: Simple, Composite, Single-valued, Multi-valued, and Derived attribute. One more attribute is their, i.e. Complex Attribute, this is the rarely used attribute.

Simple attribute :
An attribute that cannot be further subdivided into components is a simple attribute.
Example: The roll number of a student, the id number of an employee.
Composite attribute :
An attribute that can be split into components is a composite attribute.

Example: The address can be further split into house number, street number, city, state, country, and pin code, the name can also be split into first name middle name, and last name.
Single-valued attribute :
The attribute which takes up only a single value for each entity instance is a single-valued attribute.

Example: The age of a student.
Multi-valued attribute :
The attribute which takes up more than a single value for each entity instance is a multi-valued attribute.

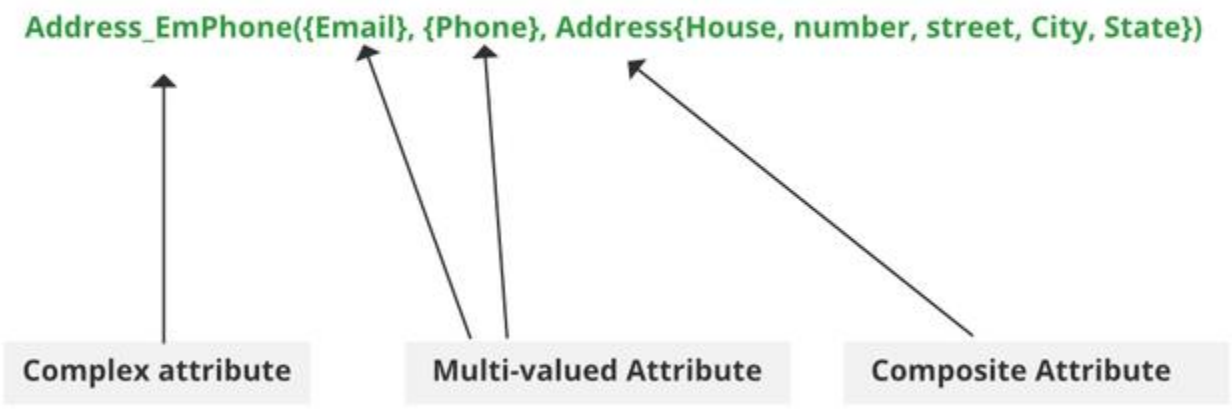
Example: Phone number of a student: Landline and mobile.
Derived attribute :
An attribute that can be derived from other attributes is derived attributes.

Example: Total and average marks of a student.
Complex attribute :
Those attributes, which can be formed by the nesting of composite and multi-valued attributes, are called "*Complex Attributes*". These attributes are rarely used in DBMS(DataBase Management System). That's why they are not so popular.
Representation:
Complex attributes are the nesting of two or more composite and multi-valued attributes. Therefore, these multi-valued and composite attributes are called 'Components' of complex attributes.

These components are grouped between parentheses '( )' and multi-valued attributes between curly braces '{ }', Components are separated by commas ', '.

For example**:** let us consider a person having multiple phone numbers, emails, and an address.

Here, phone number and email are examples of multi-valued attributes and address is an example of the composite attribute, because it can be divided into house number, street, city, and state.



Complex attributes

Components:
Email, Phone number, Address(All are separated by commas and multi-valued components are represented between curly braces).

Complex Attribute: Address_EmPhone(You can choose any name).

7. Explain functional dependency with different IR rules.

Functional Dependencies:

FD are used to specify formal measures of the "goodness" of relational designs and Keys are used to define normal forms for relations. FD are constrains that are derived from the meaning and interrelationship of the data attributes.

A set of attributes X functionally determines a set of attributes Y if the value of X determines a unique value of Y.

X→Y holds if whenever two tuples have the same value for X, they must have the same value of Y.

For any two tuples t1 and t2 in any relation instance r[R]: if t1[X]=t2[X], then t1[Y]=t2[Y]

X→Y in R specifies a constraint on all realtion instances r(R), Written as X →Y; can be displayed graphically on a relation schema . FDs are derived from the real-world constraints on the attributes.

Example:

Note that given the state of the TEACH relation, we can say that the FD: Text → Course may exist. However, the FDs Teacher → Course, Teacher → Text and Couse → Text are ruled out.

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

``````````````````

## Rules of functional dependencies

There are six inference rules which are as follows −

- Reflexivity: X->X // An attribute(s) determines itself
- Augmentation: if X->Y then XZ->YZ
- Transitivity: if X->Y & Y->Z then X->Z
- Additivity or Union : if X->Y & X->Z then X->YZ
- Projectivity or Decomposition: If X->YZ then X-> Y & X->Z
- Pseudo-Transitivity: If X->Y, YZ->W then XZ->W
``````````````````

8. Define the following. i)Null values ii)Composite Attribute iii) Multivalued Attribute, iv) Weak Entity

Null values:

A null value in a relational database is used when the value in a column is unknown or missing. A null is neither an empty string (for character or datetime data types) nor a zero value (for numeric data types).

Composite attribute :
An attribute that can be split into components is a composite attribute.

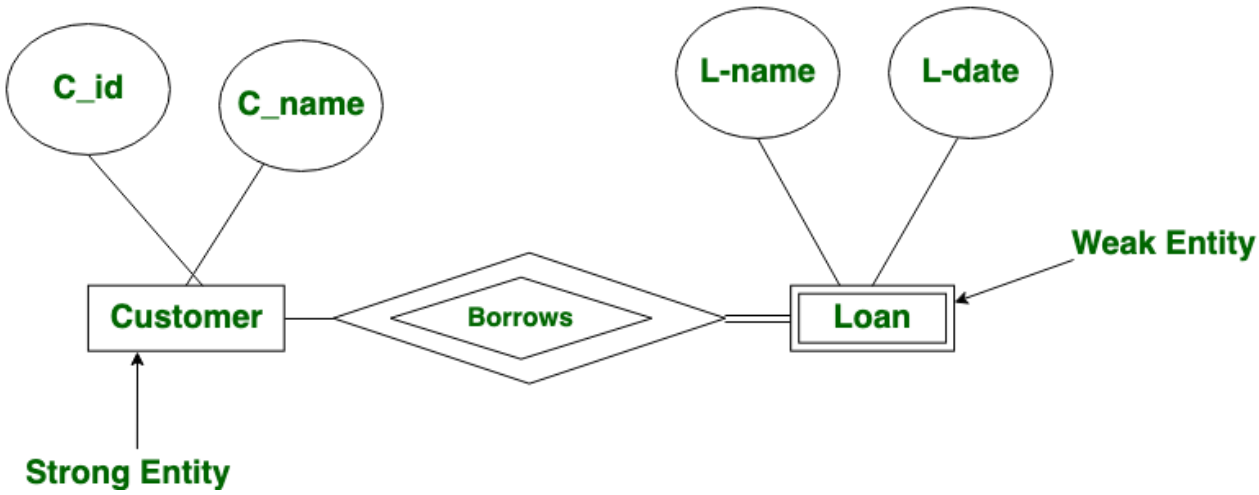
Example: The address can be further split into house number, street number, city, state, country, and pin code, the name can also be split into first name middle name, and last name.
Single-valued attribute :
The attribute which takes up only a single value for each entity instance is a single-valued attribute.

Example: The age of a student.
Multi-valued attribute :
The attribute which takes up more than a single value for each entity instance is a multi-valued attribute.

Example: Phone number of a student: Landline and mobile.

Weak Entity:

The entity sets which do not have sufficient attributes to form a primary key are known as **weak entity sets** and the entity sets which have a primary key are known as strong entity sets.
As the weak entities do not have any primary key, they cannot be identified on their own, so they depend on some other entity (known as owner entity). The weak entities have total participation constraint (existence dependency) in its identifying relationship with owner identity. Weak entity types have partial keys. Partial Keys are set of attributes with the help of which the tuples of the weak entities can be distinguished and identified.
Note – Weak entity always has total participation but Strong entity may not have total participation.
Weak entity is **depending on strong entity** to ensure the existence of weak entity. Like strong entity, weak entity does not have any primary key, It has partial discriminator key. Weak entity is represented by double rectangle. The relation between one strong and one weak entity is represented by double diamond.



Weak entities are represented with **double rectangular** box in the ER Diagram and the identifying relationships are represented with double diamond. Partial Key attributes are represented with dotted lines.

9. Define Normalization. Explain 1NF, 2NF and 3NF with examples.

- 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.

Why do we need Normalization?

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. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

Data modification anomalies can be categorized into three types:

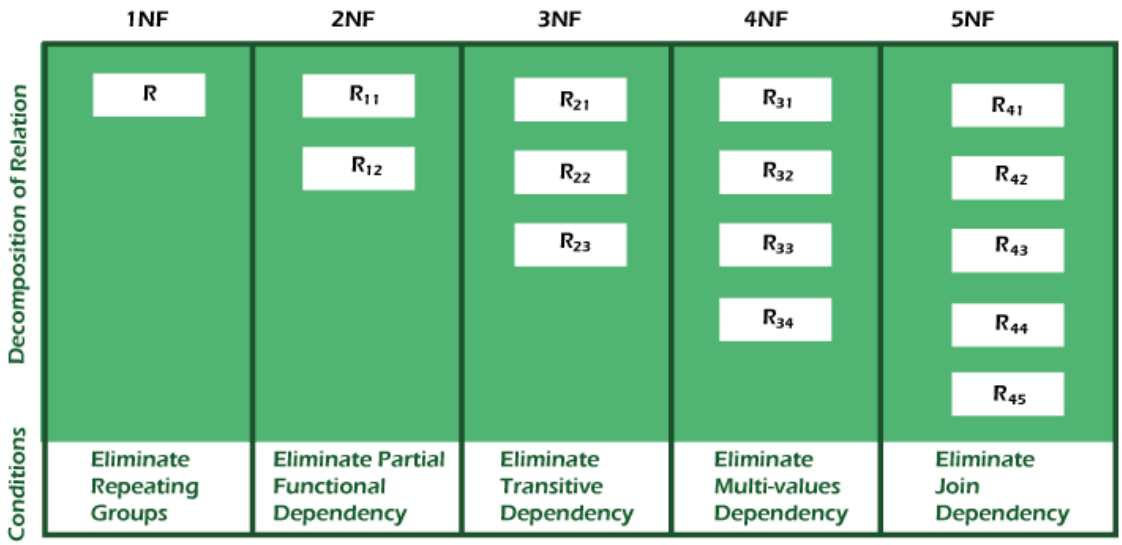
- **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.

- **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
- **Updatation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

Types of Normal Forms:

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

Following are the various types of Normal forms:



- **First Normal Form (1NF)**

  Definition: The relation is said to be in 1NF

  if all the attributes in a relation must have atomic domains

- the domain of an attribute must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute
- The only attribute values permitted by 1NF are single atomic (or indivisible) values

  The main techniques for a relation to be qualified in first normal form:

Remove the attribute that violates 1NF and place it in a separate relation along with the primary key.

This decomposes the non-1NF relation into two 1NF relations

Expand the key so that there will be a separate tuple in the original relation for each instance.

- In this case, the primary key becomes the combination of keys.
- This solution has the disadvantage of introducing redundancy in the relation.

If a maximum number of values is known for the attribute, replace the attribute by atomic attributes.

- This solution has the disadvantage of introducing NULL values.

PRODUCT		
Product_ID	Colour	Price
1	Black, Red	210
2	Green	150
3	Red	110
4	Green, Blue	260
5	Black	100

PRODUCT		
Product_ID	Colour	Price
1	Black	210
1	Red	210
2	Green	150
3	Red	110
4	Green	260
4	Blue	260
5	Black	100

Second Normal Form (2NF)

It is based on the concept of full functional dependency

- A functional dependency X → Y is a fully functional dependency if removal of any attribute A from X means that the dependency does not hold any more
- Example: {Ssn, Pnumber} → Hours ➔ neither Ssn → Hours nor Pnumber → Hours would become not functionally dependent

D**efinition**: A relation schema R is in 2NF

if every non-prime attribute A in R is

fully functionally dependent on the primary key of R

- The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key.
- If the primary key contains a single attribute, the test need not be applied at all

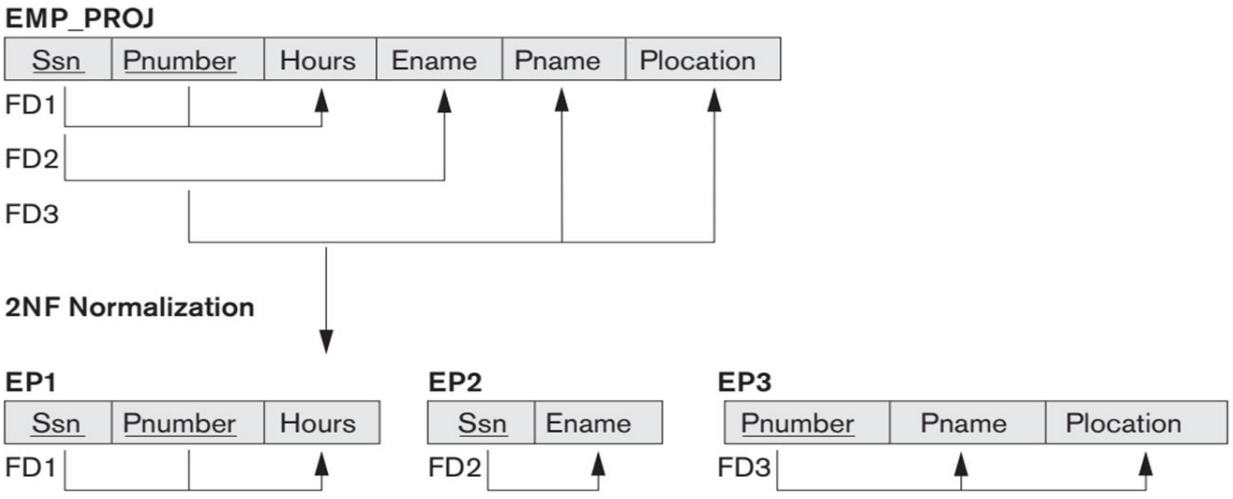
PURCHASE		
Customer_ID	Store_ID	Location
1	1	Bangalore
1	3	Hyderabad
2	1	Bangalore
3	2	Chennai
4	3	Hyderabad

-

-

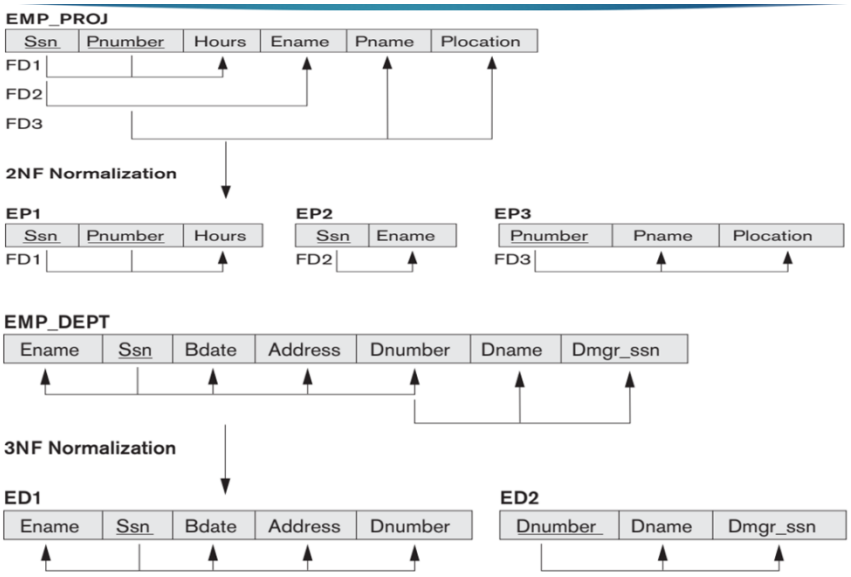
PURCHASE	
Customer_ID	Store_ID
1	1
1	3
2	1
3	2
4	3

STORE	
Store_ID	Location
1	Bangalore
3	Hyderabad
1	Bangalore
2	Chennai
3	Hyderabad



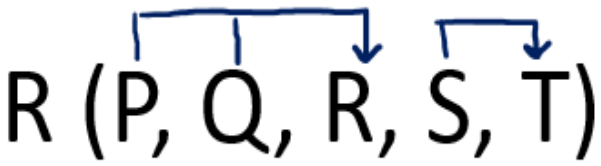
Third Normal Form (3NF)

- It is based on the concept of transitive dependency
- A functional dependency X → Y in a relation schema R is a transitive dependency if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and both X → Z and Z → Y hold
- Definition: a relation schema R is in 3NF
- if it satisfies 2NF and no non-prime attribute of R is
- transitively dependent on the primary key



10. Given Relation R(P, Q, R, S, T) and FD set ={PQ → R,S → T }determine whether the given R is in 2NF? If not convert it into 2 NF.

Solution: Let us construct an arrow diagram on R using FD to calculate the candidate key.



From above arrow diagram on R, we can see that an attributes PQS is not determined by any of the given FD, hence PQS will be the integral part of the Candidate key, i.e., no matter what will

be the candidate key, and how many will be the candidate key, but all will have PQS compulsory attribute.

Let us calculate the closure of PQS

PQS + = PQSRT (from the method we studied earlier)

Since the closure of PQS contains all the attributes of R, hence **PQS is Candidate Key**

From the definition of Candidate Key (**Candidate Key is a Super Key whose no proper subset is a Super key**)

Since all key will have PQS as an integral part, and we have proved that PQS is Candidate Key. Therefore, any superset of PQS will be Super Key but not Candidate key.

Hence there will be only **one candidate key PQS**

Definition of 2NF: No non-prime attribute should be partially dependent on Candidate Key.

Since R has 5 attributes: - P, Q, R, S, T and Candidate Key is PQS, Therefore, prime attributes (part of candidate key) are P, Q, and S while a non-prime attribute is R and T

a) FD: **PQ → R** does not satisfy the definition of 2NF, that non-prime attribute(R) is partially dependent on part of candidate key PQS.

b) FD: **S → T** does not satisfy the definition of 2NF, as a non-prime attribute(T) is partially dependent on candidate key PQS (i.e., key should not be broken at any cost).

Hence, FD PQ → R and S → T, the above table R(P, Q, R, S, T) is not in 2NF

Convert the table R(P, Q, R, S, T) in 2NF:

Since due to FD: PQ → R and S → T, our table was not in 2NF, let's decompose the table

R1(P, Q, R) (Now in table R1 FD: PQ → R is Full F D, hence R1 is in 2NF)

R2(S, T) (Now in table R2 FD: S → T is Full F D, hence R2 is in 2NF)

And create one table for the key, since the key is PQS.

R3(P, Q, S)

Finally, the decomposed tables which are in 2NF are:

a) R1(P, Q, R) b) R2(S, T) c) R3(P, Q, S)