IT1401A - DATABASE MANAGEMENT SYSTEM

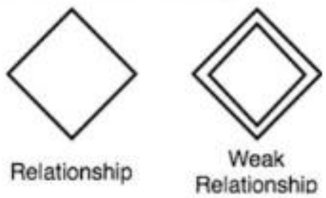
UNIT II – DATABASE DESIGN

PART-A

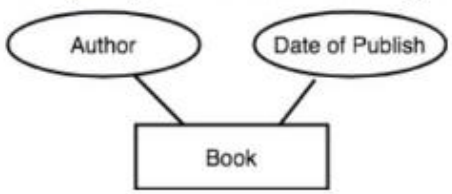
1. List the different symbols used in ER diagram with examples.

Entity
Simple rectangular box represents an Entity.

Relationships between Entities - Weak and Strong
Rhombus is used to setup relationships between two or more entities.



Attributes for any Entity
Ellipse is used to represent attributes of any entity. It is connected to the entity.

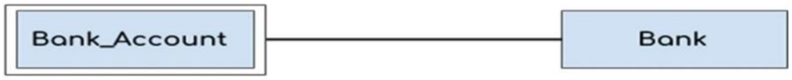


Weak Entity
A weak Entity is represented using double rectangular boxes. It is generally connected to another entity.

Key Attribute for any Entity
To represent a Key attribute, the attribute name inside the Ellipse is underlined.

2. What is a weak entity? Give an example.

An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle. For example – a bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.



3. **What are the desirable properties of decomposition?**

- **Lossless:** All the decomposition that we perform in Database management system should be lossless. All the information should not be lost while performing the join on the sub-relation to get back the original relation. It helps to remove the redundant data from the database.

- **Dependency Preservation:** Dependency Preservation is an important technique in database management system. It ensures that the functional dependency between the entities is maintained while performing decomposition. It helps to improve the database efficiency, maintain consistency and integrity.

- **Lack of Data Redundancy:** Data Redundancy is generally termed as duplicate data or repeated data. This property states that the decomposition performed should not suffer redundant data. It will help us to get rid of unwanted data and focus only on the useful data or information.

4. **What is meant by non-loss decomposition?**

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.

5. **Define the two types of functional dependency with an example**.

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.

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.

6. **Define full functional dependency.**

Fully functional dependencies mean that every attribute in a set uniquely determines the value of another attribute, and no proper subset of the set has this property. This ensures minimal redundancy and dependency ambiguity within the relation.

7. **Consider the following relation: R (A, B, C, D, E). The primary key of the relation is AB. The following functional dependencies hold: A->C, B->D, AB->E. Is the above relation in second normal form?**

   Yes, the relation is in Second Normal Form (2NF) because all non-prime attributes (C, D, and E) are fully functionally dependent on the entire primary key (AB).

8. **Define BCNF.**

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.

For a table to be in BCNF, following conditions must be satisfied:

- R must be in 3rd Normal Form
- and, for each functional dependency (X → Y), X should be a super Key.

9. **Compare 3NF with BCNF with an example**

3 NF	BCNF
1.It concentrates on the primary key	it concentrates on all candidate keys
2.Redundancy is high compared to BCNF	Redundancy is low compared to 3 NF
3.It may preserve all dependencies	It may not preserve all F.D's
If there is a dependency x→ y is allowed in 3NF if x is a super key or Y is the	If there is a dependency x→y. It is allowed in BCNF if X is super key.

EXAMPLE: 3NF

Retail_Outlet_ID	Street	City	Pincode
R1001	King Street	Hyderabad	530001

Retail_Outlet_ID	Street	Pincode
R1001	King Street	530001

Pincode	City
53001	Hyderabad

Example:

Table 1

user_id	first_name	Last_name	Gender
1	Sam	M	M
2	Sejal	B	F
3	John	M	M
4	Reena	P	F

Table 2

user_id	contact_no
1	8498584849
1	9994848484
2	9383839927
2	7373782882
3	8796958589
4	9585884474

Table 3

Gender	Salutation
M	Mr.
F	Mrs.

EXAMPLE: BCNF

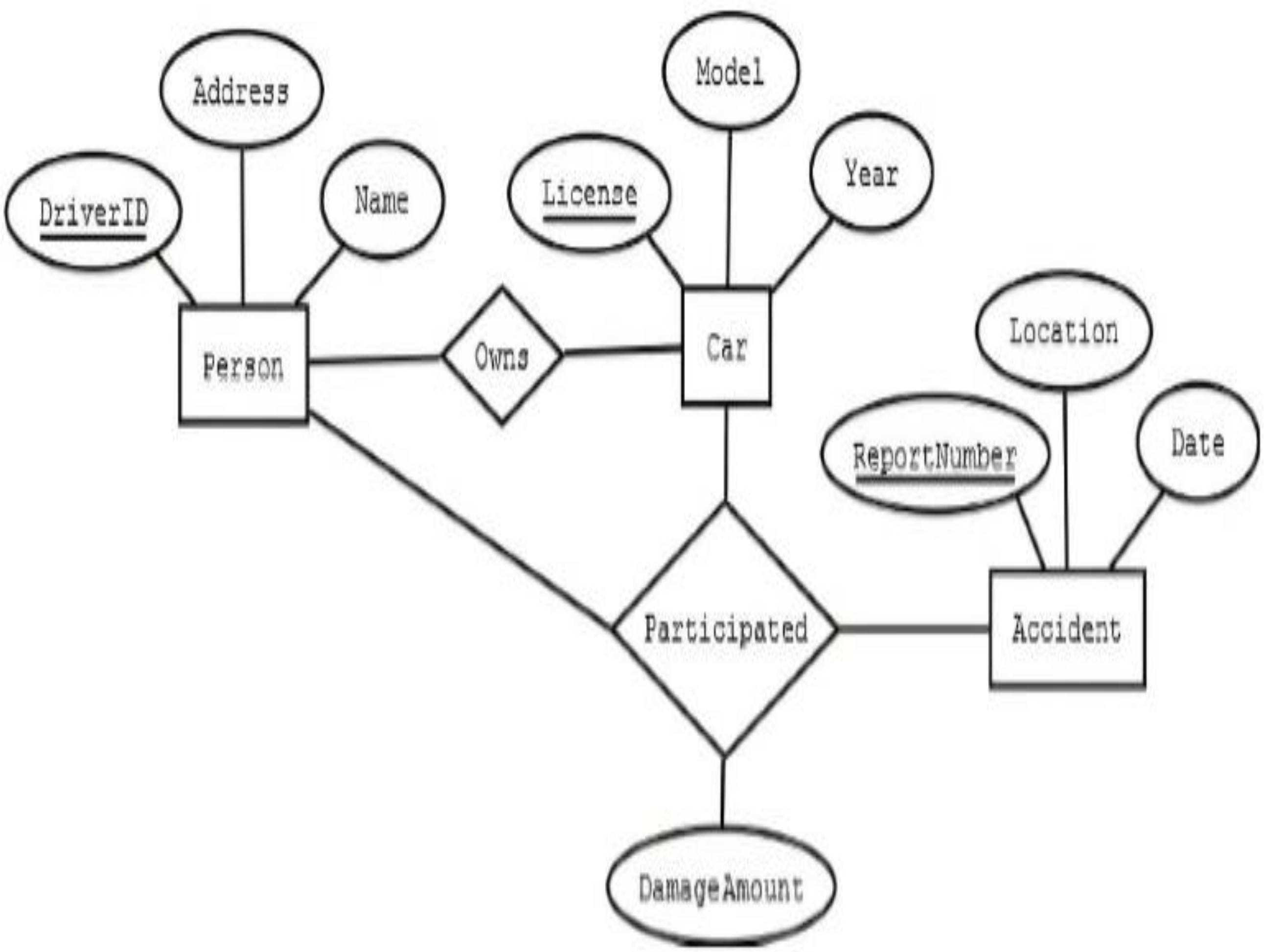
10.Define Fourth Normal Form.

Fourth Normal Form (4NF) ensures that relations are free from certain types of multi-valued dependencies, promoting data integrity by eliminating redundancy and anomalies in the database schema.

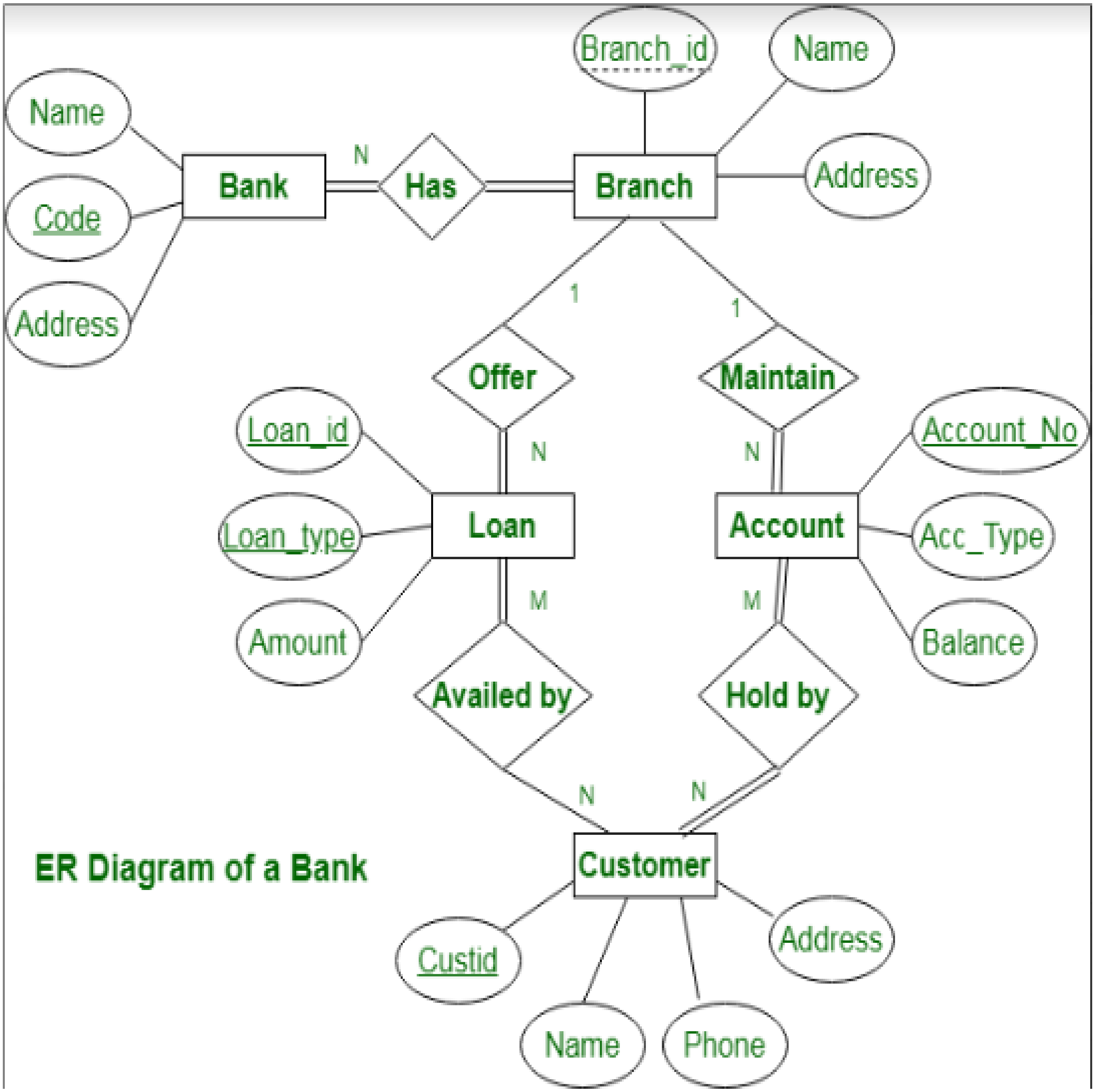
PART-B

1. **(i) Construct an ER diagram for a car insurance company whose customers own**
   one or more cars each. Each car has associated with zero to any number of recorded accidents. Each insurance policy covers one or more cars and has one or more premium payments associated with it. Each payment is for a particular period of time, and has an associated due date and the date when the payment was received.
   (ii) Construct an ER diagram for a Banking System.

(i)



(ii)



<u>Explanation</u>:

Entities: Customer: Represents a customer of the bank, with attributes like CustomerID (primary key), Name, Address, Phone, etc.

Account: Represents a bank account, with attributes like AccountNumber (primary key), AccountType (e.g., Savings, Checking), Balance, etc.

Transaction: Represents a financial transaction, with attributes like TransactionID (primary key), AccountNumber (foreign key referencing Account), TransactionType (e.g., Deposit, Withdrawal), Amount, Date, etc.

Loan: Represents a loan granted to a customer, with attributes like LoanID (primary key), CustomerID (foreign key referencing Customer), LoanAmount, InterestRate, Term, etc.

Relationships:

Customer has a one-to-many relationship with Account: One customer can have multiple accounts.

Account has a one-to-many relationship with Transaction: One account can have multiple transactions.

Customer has a one-to-many relationship with Loan: One customer can have multiple loans.

2. **What is data normalization? Apply the understanding of first normal form, second normal form, third normal form by providing examples that illustrate the key principles and transformations involved in achieving each level of normalization**.

Normalization:

- The normalization process is first proposed by Codd (1972)
- The **normalization process** takes a relation schema through a series of tests to certify whether it satisfies a certain **normal form**. The process, which proceeds in a top-down fashion by evaluating each relation against the criteria for normal forms and decomposing relations as necessary, can thus be considered as relational design by analysis.
- Initially, Codd proposed three normal forms, which he called first, second, and third normal form. A stronger definition of 3NF—called Boyce-Codd normal form (BCNF)—was proposed later by Boyce and Codd. All these normal forms are based on a single analytical tool: the functional dependencies among the attributes of a relation. Later, a fourth normal form (4NF) and a fifth normal form (5NF) were proposed, based on the concepts of multivalued dependencies and join dependencies, respectively.

Normalization of data can be considered a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of

- minimizing redundancy
- minimizing the insertion, deletion, and update anomalies

Example: Suppose a manufacturing company stores the employee details in a table named employee that has four attributes: emp_id for storing employee's id, emp_name for storing employee's name, emp_address for storing employee's address and emp_dept for storing the department details in which the employee works. At some point of time the table looks like this:

emp_id	emp_name	emp_address	emp_dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890
166	Glenn	Chennai	D900
166	Glenn	Chennai	D004

First Normal Form:

For a table to be in the First Normal Form, it should follow the following 4 rules:

1. It should only have single(atomic) valued attributes/columns.
2. Values stored in a column should be of the same domain
3. All the columns in a table should have unique names.
4. And the order in which data is stored, does not matter.

Rules for First Normal Form

The first normal form expects you to follow a few simple rules while designing your database, and they are:

Rule 1: Single Valued Attributes Each column of your table should be single valued which means they should not contain multiple values. We will explain this with help of an example later, let's see the other rules for now.

Rule 2: Attribute Domain should not change This is more of a "Common Sense" rule. In each column the values stored must be of the same kind or type. For example: If you have a column dob to save date of births of a set of people, then you cannot or you must not save 'names' of some of them in that column along with 'date of birth' of others in that column. It should hold only 'date of birth' for all the records/rows.

Rule 3: Unique name for Attributes/Columns This rule expects that each column in a table should have a unique name. This is to avoid confusion at the time of retrieving data or performing any other operation on the stored data. If one or more columns have same name, then the DBMS system will be left confused.

Rule 4: Order doesn't matters This rule says that the order in which you store the data in your table doesn't matter. EXAMPLE Create a table to store student data which will have student's roll no., their name and the name of subjects they have opted for. Here is the table, with some sample data added to it.

roll_no	name	subject
101	Akon	OS, CN
103	Ckon	Java
102	Bkon	C, C++

The table already satisfies 3 rules out of the 4 rules, as all our column names are unique, we have stored data in the order we wanted to and we have not inter-mixed different type of data in columns.

But out of the 3 different students in our table, 2 have opted for more than 1 subject. And we have stored the subject names in a single column. But as per the 1st Normal form each column must contain atomic value. It's very simple, because all we have to do is break the values into atomic values. Here is our updated table and it now satisfies the First Normal Form.

roll_no	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	Java
102	Bkon	C
102	Bkon	C++

By doing so, although a few values are getting repeated but values for the subject column are now atomic for each record/row. Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

Second Normal Form (2NF)

For a table to be in the Second Normal Form,

1. It should be in the First Normal form.
2. And, it should not have Partial Dependency.
3. Let's take an example of a **Student** table with columns student_id, name, reg_no(registration number), branch and address(student's home address).

student_id	name	reg_no	branch	address

In this table, student_id is the primary key and will be unique for every row, hence we can use student_id to fetch any row of data from this table

Even for a case, where student names are same, if we know the student_id we can easily fetch the correct record.

student_id	name	reg_no	branch	address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat

Hence we can say a **Primary Key** for a table is the column or a group of columns(composite key) which can uniquely identify each record in the table.

I can ask from branch name of student with student_id **10**, and I can get it. Similarly, if I ask for name of student with student_id **10** or **11**, I will get it. So all I need is student_id and every other column **depends** on it, or can be fetched using it.

This is **Dependency** and we also call it **Functional Dependency**.

Third Normal Form (3NF)

A table is said to be in the Third Normal Form when,

1. It is in the Second Normal form.
2. And, it doesn't have Transitive Dependency.

Requirements for Third Normal Form

For a table to be in the third normal form,

1. It should be in the Second Normal form.
2. And it should not have Transitive Dependency.

   With exam_name and total_marks added to our Score table, it saves more data now. Primary key for our Score table is a composite key, which means it's made up of two attributes or columns → **student_id + subject_id**.

   Our new column exam_name depends on both student and subject. For example, a mechanical engineering student will have Workshop exam but a computer science student won't. And for some subjects you have Practical exams and for some you don't. So we can say that exam_name is dependent on both student_id and subject_id.

   And what about our second new column total_marks? Does it depend on our Score table's primary key?

   Well, the column total_marks depends on exam_name as with exam type the total score changes. For example, practicals are of less marks while theory exams are of more marks.

   But, exam_name is just another column in the score table. It is not a primary key or even a part of the primary key, and total_marks depends on it.

   This is **Transitive Dependency**. When a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key.

How to remove Transitive Dependency?

Again the solution is very simple. Take out the columns exam_name and total_marks from Score table and put them in an **Exam** table and use the exam_id wherever required.

score_id	student_id	subject_id	marks	exam_id

exam_id	exam_name	total_marks
1	Workshop	200
2	Mains	70
3	Practicals	30

<u>Advantage of removing Transitive Dependency</u>

The advantage of removing transitive dependency is,

- Amount of data duplication is reduced.
- Data integrity achieved.

3.Normalize the following table into 1NF, 2NF and 3NF.EMPLOYEE [EMPNO, EMPNAME, WORKDEPT, DEPTNAME, SKILLID1, SKILLID2, SKILLID3]

Unnormalized table

EMP_NO	EMP_NAME	WORKDEPT	DEPT NAME	SKILL1	SKILL2	SKILL3
030	KWAN	C01	Marketing	141		
250	SMITH	D21	purchasing	002	011	067
270	PEREZ	E11	Personnel	415	447	
300	SMITH	D21	purchasing	011	032.	

Normalized to 1NF

EMP TABLE

EMP_SKILL TABLE
- EMP-ID
- SKILL-ID
- SKILL_NAME

EMPLOYEE Table

EMP_NO	EMP_NAME	WORKDEPT	DEPT_NAME
030	KWAN	C01	Marketing
250	SMITH	D21	purchasing
270	PEREZ	E11	personnel
300	SMITH	D21	purchasing

Transitivity problem exists
so ~~we have~~ it has to normalized
to 3NF

Composite Primary Key EMP_SKILL Table

EMP_NO	SKILL_ID	SKILL NAME
030	141	Research
250	002	Bid preparation
250	011	Negotiation
250	067	Product Specification
270	415	Benefit Analysis
270	447	Testing
300	011	Negotiation
300	032	Inventory Control

partial dependency exists.
So it has to be normalized to 2NF.

Second Normal Form (2NF)

The objective of 2NF is to eliminate partial dependency.

Partial dependency means column(s) partly depend on the primary key.

EMP_SKILL
(Normalized to 2NF)

EMP_SKILL
(EMPNO, SKILL_ID)

SKILL Table
(SKILL_ID, SKILL NAME)

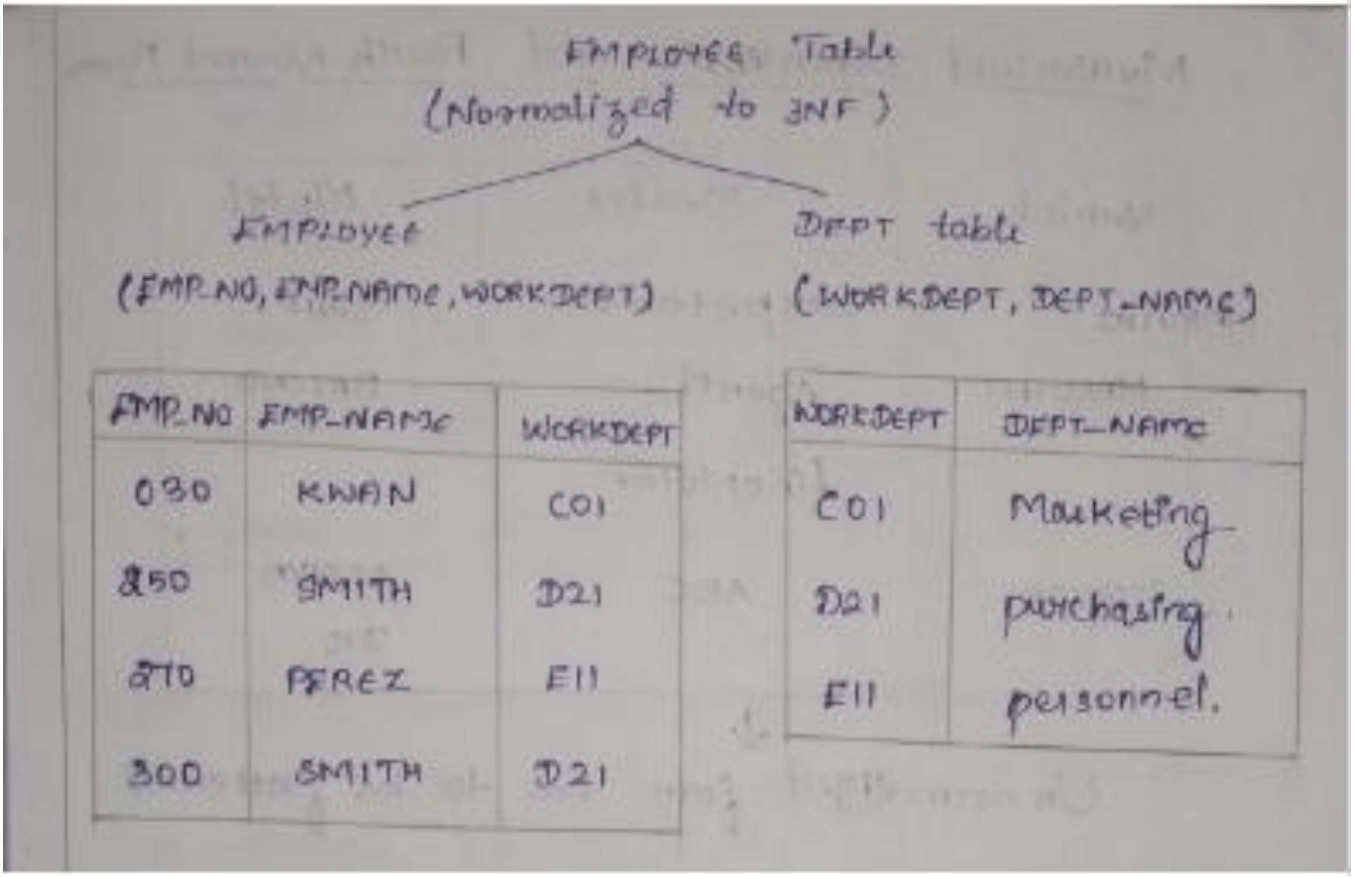
EMP_NO	SKILL_ID
030	141
250	002
250	011
250	067
270	415
270	447
300	011
300	032

SKILL_ID	SKILL NAME
141	Research
002	Bid preparation
011	Negotiation.
067	product specification
415	Benefit Analysis
447	Testing
032	Inventory control

Third Normal Form (3NF)

The objective of 3NF is to eliminate transitive dependency.

Eliminate columns that don't depend on the key at all.



EMPLOYEE Table
(Normalized to 3NF)

EMPLOYEE
(EMP_NO, EMP_NAME, WORKDEPT)

DEPT table
(WORKDEPT, DEPT_NAME)

EMP_NO	EMP_NAME	WORKDEPT
030	KWAN	C01
250	SMITH	D21
270	PEREZ	E11
300	SMITH	D21

WORKDEPT	DEPT_NAME
C01	Marketing
D21	purchasing
E11	personnel.

4.

Normalize the following invoice into 1NF, 2NF, and 3NF.

HILLTOP ANIMAL HOSPITAL DATE: JAN 13/2018 INVOICE # 987

MR. RICHARD COOK 123 THIS STREET MY
CITY, ONTARIO Z5Z 6G6

PET PROCEDURE AMOUNT ROVER RABIES

VACCINATION 30.00
MORRIS RABIES VACCINATION 24.00

TOTAL 54.00
TAX (8%) 4.32

AMOUNT OWING 58.32

Solution to Normalization Problem

INVOICE

Hilltop Animal Hospital
Invoice #987

Date: Jan 13/2002

Mr. Richard Cook
123 This Street
My City, Ontario
Z5Z 6G6

Pet	Procedure	Amount
Rover	Rabbies Vaccination	30
Morris	Rabbies Vaccination	24

Total	54.00
Tax (8%)	4.32
Amount owing	58.32

The above invoice is transformed into Relational Model

UNF: [Invoice_No , Invoice_Date, Cust_Name, Cust_Add, Pet_Name, Procedure, Amount]

Unnormalized Form:

Invoice

Invoice_No	Invoice_Date	Cust_Name	Cust_Add	Pet_Name	Procedure	Amt
987	13-JAN-2002	Mr. Richard	123 This Street My City, Ontario Z5Z 6G6	Rover Morris	Rabbies V Rabbies V	30 24
988	12-FEB-2002	Mr. Lawrence	Plot 10 F5 Arun Flats Avadi 600054	Puppy Johny	Rabbies V Rabbies V	30 24

In the above relation (Cust_Add, Pet_Name, Procedure, Amt) attributes contain 2 or more values i.e Repeating groups found. Hence it is UNF, so have to be normalized to 1NF.

SECOND NORMAL FORM:

Invoice_Pet [Invoice_No, ~~Invoice_Date~~ Pet_id, Procedure, Amount]

Pet [Pet_id, Pet_Name]

⇓

Invoice_Pet Table

Invoice_No	Pet_id	Procedure	Amount
987	1	Rabies_Vacc	30.00
987	2	Rabies_Vacc	24.00
988	3	Rabies_Vacc	30.00
988	4	Rabies_Vacc	24.00

Pet Table

Pet_id	Pet_name
1	Rover
2	Morris
3	Puppy
4	Johny

The above relations are in 2NF as partial dependency does not exist.

THIRD NORMAL FORM: (All attributes must completely depend on PK)

Invoice [Invoice_No, Invoice_Date, Cust_Name, Cust_Street, Cust_City, Cust_Pstkd]

Transitive Dependency exists in the above (↑) relation:

Invoice_No → Invoice_Date (Invoice_Date depends completely on Invoice_No.)

Invoice_No → Cust_Name (Completely Dependent)

Cust_Name → Cust_Street

Cust_Name → Cust_City

Cust_Name → Cust_Pstkd

[Transitive dependency exists as Cust_Street, City, Pstkd do not depend on Invoice_No]

Hence the above relation has to be normalized into 3NF.

Normalized to 3NF :

Invoice [Invoice_No, Invoice_Date, Cust_Name]

Customer [Cust_Name, Cust_Street, Cust_City, Cust_pstlcd]

Invoice Table

Invoice_No	Invoice_Date	Cust_Name
987	13-JAN-2002	Mr. Richard Cook
988	12-FEB-2002	Mr. Lawrence

Customer Table

Cust_Name	Cust_Street	Cust_City	Cust_pstlcd
Mr. Richard Cook	123 This Street	My_City_Ontario	Z5Z6G6
Mr. Lawrence	Plot10_F5_Arun Flats	Avadi	600054

The above relations are in 3NF as Transitive Dependency does not exist i.e. all columns in the above two tables completely depend on the Primary Key.

FIRST NORMAL FORM:

Invoice [Invoice_No, Invoice_Date, Cust_Name, Cust_Street, Cust_City, Cust_pstcd]

Invoice_Pet [Invoice_No, Pet_Id, Pet_Name, Procedure, Amt]

⇓

Invoice Table

Invoice_No	Invoice_Date	Cust_Name	Cust_Street	Cust_City	Cust_Pstcd
987	13-JAN-2002	Mr. Richard Cook	123_This Street	My_City	Z5Z...
988	12-FEB-2002	Mr. Lawrence	Plot 10_F5_Anup Flats	Avadi	60005...

Invoice_Pet Table

Invoice_No	Pet_Id	Pet_Name	Procedure	Amount
987	1	Rover	Rabbies_Vacc	30.00
987	2	Morris	Rabbies_Vacc	24.00
988	3	Puppy	Rabies_Vacc	30.00
988	4	Johny	Rabies_Vacc	24.00

The above two relations are in 1NF as it saties the following rules: [No repeating groups]

① It has only single (atomic) valued attributes/columns.

② Values stored in a column is of same domain

③ All the columns in the above 2 relations have unique names

④ And the order in which data is stored does not matter.

Partial Dependency

In the above relation Invoice_Pet table partial dependency exists.

Pet_Id ⟶ Pet_Name [Partial dependency exists, i.e Pet_Name partially depends on the composite primary key]

Hence the relation Invoice_Pet has to normalized to 2NF.

(i)

5. (i)Compare BCNF and 3NF with appropriate example.

(ii) Give an example of a relation that is in 3NF and not in BCNF. How will you convert that relation into BCNF?

(i)

S.NO.	3NF	BCNF
1.	3NF stands for Third Normal Form.	BCNF stands for Boyce Codd Normal Form.
2.	In 3NF there should be no transitive dependency that is no non prime attribute should be transitively dependent on the candidate key.	In BCNF for any relation A->B, A should be a super key of relation.
3.	It is less stronger than BCNF.	It is comparatively more stronger than 3NF.
4.	In 3NF the functional dependencies are already in 1NF and 2NF.	In BCNF the functional dependencies are already in 1NF, 2NF and 3NF.
5.	The redundancy is high in 3NF.	The redundancy is comparatively low in BCNF.
6.	In 3NF there is preservation of all functional dependencies.	In BCNF there may or may not be preservation of all functional dependencies.
7.	It is comparatively easier to achieve.	It is difficult to achieve.
8.	Lossless decomposition can be achieved by 3NF.	Lossless decomposition is hard to achieve in BCNF.
9.	The table is in 3NF if it is in 2NF and for each functional dependency X->Y at least following condition hold: (i) X is a super key, (ii) Y is prime attribute of table.	The table is in BCNF if it is in 3rd normal form and for each relation X->Y X should be super key.
10.	3NF can be obtained without sacrificing all dependencies.	Dependencies may not be preserved in BCNF.
11.	3NF can be achieved without losing any information from the old table.	For obtaining BCNF we may lose some information from old table.

a. **Third Normal Form (3NF):** Eliminates transitive dependencies, where a non-prime attribute depends on another non-prime attribute through a primary key.

Example:

Retail_Outlet_ID	Street	City	Pincode
R1001	King Street	Hyderabad	530001

Retail_Outlet_ID	Street	Pincode
R1001	King Street	530001

Pincode	City
53001	Hyderabad

b. **Boyce-Codd Normal Form (BCNF):** Takes 3NF further by eliminating all non-trivial functional dependencies where the determinant is not a superkey.

Example:

Table 1

user_id	first_name	Last_name	Gender
1	Sam	M	M
2	Sejal	B	F
3	John	M	M
4	Reena	P	F

Table 2

user_id	contact_no
1	8498584849
1	9994848484
2	9383839927
2	7373782882
3	8796958589
4	9585884474

Table 3

Gender	Salutation
M	Mr.
F	Mrs.

II. Relation in 3NF but Not BCNF

Example Relation: Courses (CourseID, Department, Instructor, CourseName, Classroom)

Functional Dependencies:

CourseID -> {Department, Instructor, CourseName, Classroom}

Explanation:

This relation is in 3NF because:

All attributes are atomic (not divisible).

There are no partial dependencies.

Every non-prime attribute (all attributes except CourseID) is fully determined by the candidate key (CourseID).

However, it violates BCNF because:

There exists a non-trivial functional dependency (Department -> Instructor) where the determinant (Department) is not a superkey of the entire relation. This creates redundancy, as Instructor can be determined indirectly from CourseID through Department.

Converting to BCNF:

Decompose the original relation into two relations:

Course (CourseID, Department, CourseName, Classroom)

Teaches (CourseID, Instructor)

Justification:

The first relation preserves the original functional dependency CourseID -> {Department, CourseName, Classroom}.

The second relation captures the functional dependency Department -> Instructor, ensuring a lossless decomposition and no redundancy.

This decomposition results in two BCNF relations, eliminating the issue in the original relation.

6. **What is the need of normalization? Analyze and implement fourth normal form and fifth normal form strategy with examples in detail.**

Normalization:

Normalization is a database design technique used to organize data in a relational database, aiming to minimize redundancy and dependency.

It involves dividing a database into two or more tables and defining relationships between them to achieve data integrity and efficiency.

<u>THE NEED FOR NORMALIZATION</u>

- Normalization is typically used in conjunction with the entity relationship modeling.
- There are two common situations in which database designers use normalization.
- When designing a new database structure based on the business requirements of the end users, the database designer will construct a data model using a technique such as Crow's Foot notation ERDs.
- After the initial design is complete, the designer can use normalization to analyze the relationships that exist among the attributes within each entity, to determine if the structure can be improved through normalization.
- Alternatively, database designers are often asked to modify existing data structures that can be in the form of flat files, spreadsheets, or older database structures.
- Again, through an analysis of the relationships among the attributes or fields in the data structure, the database designer can use the normalization process to improve the existing data structure to create an appropriate database design.
- Whether designing a new database structure or modifying an existing one, the normalization process is the same.
- 1. Data Integrity
- 2. Minimization of Redundancy
- 3. Efficient Storage Utilization
- 4. Improved Query Performance
- 5. Simplified Updates and Maintenance.

Fourth Normal Form (4NF):

A table is said to be in the Fourth Normal Form when,

1. It is in the Boyce-Codd Normal Form.

2. And, it doesn't have Multi-Valued Dependency.

Multi-valued Dependency A table is said to have multi-valued dependency, if the following conditions are true,

1. For a dependency A → B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.

2. Also, a table should have at-least 3 columns for it to have a multi-valued dependency.

3. And, for a relation R (A, B, C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

Example Below we have a college enrolment table with columns s_id, course and hobby.

s_id	cours e	hobby
1	Scienc e	Cricket
1	Maths	Hockey
2	C#	Cricket
2	Php	Hockey

From the table above, student with s_id 1 has opted for two courses, Science and Maths, and has two hobbies, Cricket and Hockey. You must be thinking what problem this can lead to, right? Well the two records for student with s_id 1, will give rise to two more records, as shown below, because for one student, two hobbies exists, hence along with both the courses, these hobbies should be specified.

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
1	Science	Hockey
1	Maths	Cricket

And, in the table above, there is no relationship between the columns course and hobby. They are independent of each other. So there is multi-value dependency, which leads to un-necessary repetition of data and other anomalies as well.

How to satisfy 4th Normal Form?

To make the above relation satisfy the 4th normal form, we can decompose the table into 2 tables.

CourseOpted Table

s_id	course
1	Science
1	Maths
2	C#
2	Php

Hobbies Table,

s_id	hobby
1	Cricket
1	Hockey
2	Cricket
2	Hockey

Now this relation satisfies the fourth normal form. A table can also have functional dependency along with multi-valued dependency. In that case, the functionally dependent columns are moved in a separate table and the multi-valued dependent columns are moved to separate tables.

Fifth Normal Form (5NF):

A database is said to be in 5NF, if and only if,

1. It's in 4NF

2. If we can decompose table further to eliminate redundancy and anomaly, and when we re-join the decomposed tables by means of candidate keys, we should not be losing the original data or any new record set should not arise. In simple words, joining two or more decomposed table should not lose records nor create new records.

What is Join Dependency

If a table can be recreated by joining multiple tables and each of this table have a subset of the attributes of the table, then the table is in Join Dependency. It is a generalization of Multivalued Dependency Join Dependency can be related to 5NF, wherein a relation is in 5NF, only if it is already in 4NF and it cannot be decomposed further.

Example:

<Employee>

EmpName	**EmpSkills**	**EmpJob** (Assigned Work)
Tom	Networking	EJ001
Harry	Web Development	EJ002
Katie	Programming	EJ002

The above table can be decomposed into the following three tables; therefore it is not in 5NF:

<EmployeeSkills>

EmpName	**EmpSkills**
Tom	Networking
Harry	Web Development
Katie	Programming

<EmployeeJob>

EmpName	**EmpJob**
Tom	EJ001
Harry	EJ002
Katie	EJ002

<JobSkills>

EmpSkills	**EmpJob**
Networking	EJ001
Web Development	EJ002
Programming	EJ002

Our Join Dependency: {(EmpName, EmpSkills), (EmpName, EmpJob), (EmpSkills, EmpJob)}

The above relations have join dependency, so they are not in 5NF. That would mean that a join relation of the above three relations is equal to our original relation <Employee>.

FIFTH NORMAL FORM EXAMPLE

Consider an example of different Subjects taught by different lecturers and the lecturers taking classes for different semesters. Note: Please consider that Semester 1 has Mathematics, Physics and Chemistry and Semester 2 has only Mathematics in its academic year!!

COURSE
SUBJECT
LECTURER
CLASS

SUBJECT	LECTURER	CLASS
Mathematics	Alex	SEMESTER 1
Mathematics	Rose	SEMESTER 1
Physics	Rose	SEMESTER 1
Physics	Joseph	SEMESTER 2
Chemistry	Adam	SEMESTER 1

In above table, Rose takes both Mathematics and Physics class for Semester 1, but she does not take Physics class for Semester 2. In this case, combination of all these 3 fields is required to identify a valid data. Imagine we want to add a new class - Semester3 but do not know which Subject and who will be taking that subject. We would be simply inserting a new entry with Class as Semester3 and leaving Lecturer and subject as NULL. As we discussed above, it's not a good to have such entries. Moreover, all the three columns together act as a primary key, we cannot leave other two columns blank! Hence we have to decompose the table in such a way that it satisfies all the rules till 4NF and when join them by using keys, it should yield correct record. Here, we can represent each lecturer's Subject area and their classes in a better way. We can divide above table into three - (SUBJECT, LECTURER), (LECTURER, CLASS), (SUBJECT, CLASS) .

SUBJECT	LECTURER
Mathematics	Alex
Mathematics	Rose
Physics	Rose
Physics	Joseph
Chemistry	Adam

	5NF
CLASS	LECTURER
SEMESTER 1	Alex
SEMESTER 1	Rose
SEMESTER 1	Rose
SEMESTER 2	Joseph
SEMESTER 1	Adam

CLASS	SUBJECT
SEMESTER 1	Mathematics
SEMESTER 1	Physics
SEMESTER 1	Chemistry
SEMESTER 2	Physics

Now, each of combinations is in three different tables. If we need to identify who is teaching which subject to which semester, we need join the keys of each table and get the result. For example, who teaches Physics to Semester 1, we would be selecting Physics and Semester1 from table 3 above, join with table1 using Subject to filter out the lecturer names. Then join with table2 using Lecturer to get correct lecturer name. That is we joined key columns of each table to get the correct data. Hence there is no lose or new data - satisfying 5NF condition.