Bhumi Gupta - EN19CS301094

Page No.:
Date:

DBMS

Assignment - 3

Part - 1

A) Q.1 DRAW and explain Two examples of ER diagram.

1) → ER Diagram of Library Management System.

The library manggement system database keeps tracks of readers with following consideration -

a) The system keeps track of the staff with a single point authentication system comprising login Id and password.

b) Staff maintain the book catalog with its ISBN, book title, price (in ₹), category (Novel, general, story), edition, author Number & details.

c) A publisher has publisher Id, year when the book was publisher & name of the book.

d) Staff also generate report that has reader ID, registration no. of report, book no (o return / Issue Info).

→ Entities of their Attributes.

a) Book Entity It has authors, ISBN number, title, edition, category price, ISBN is the primary key book entity.

b) Publisher Entity It has publisher is, year of publication, name, publisher Id is the primary key.

c) Authentication system Entity It has login ID and password with login ID as primary key.

d) Staff Entity It has name and staff ID with staff ID as primary key.

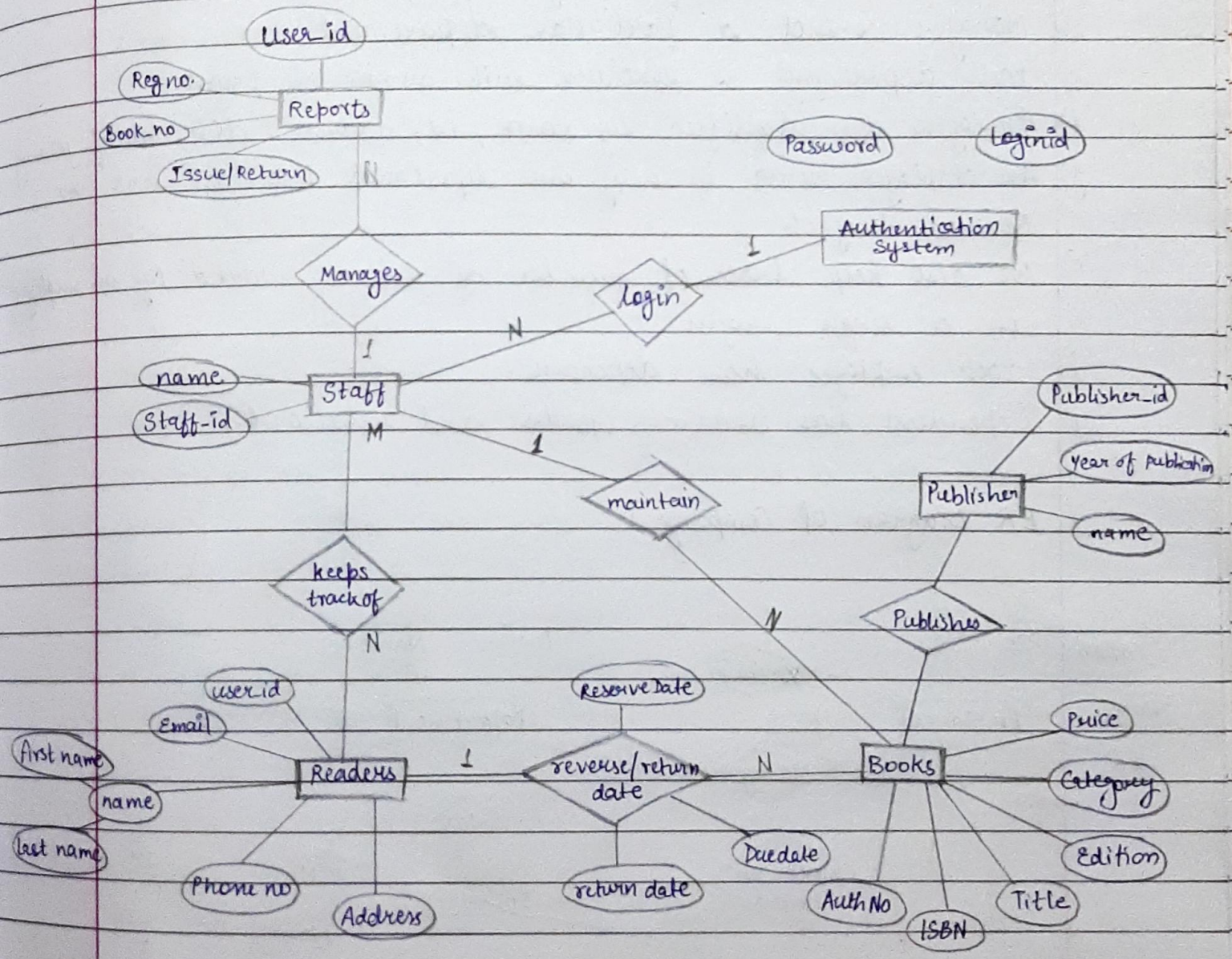
e) Reverse Relationship Set - It has three attributes: Reverse date, Due date, Return date.

→ Relationship between Entities.

1) A reader can reverse N books but one book can be reserved by only one reader. The relationship is 1:N.

2) staff keeps track of readers. The relationship is M:N.

3) Staff maintains multiple books. The relationship is 1:N.
4) Authentication system provides login to multiple staffs. The relation is 1:N.

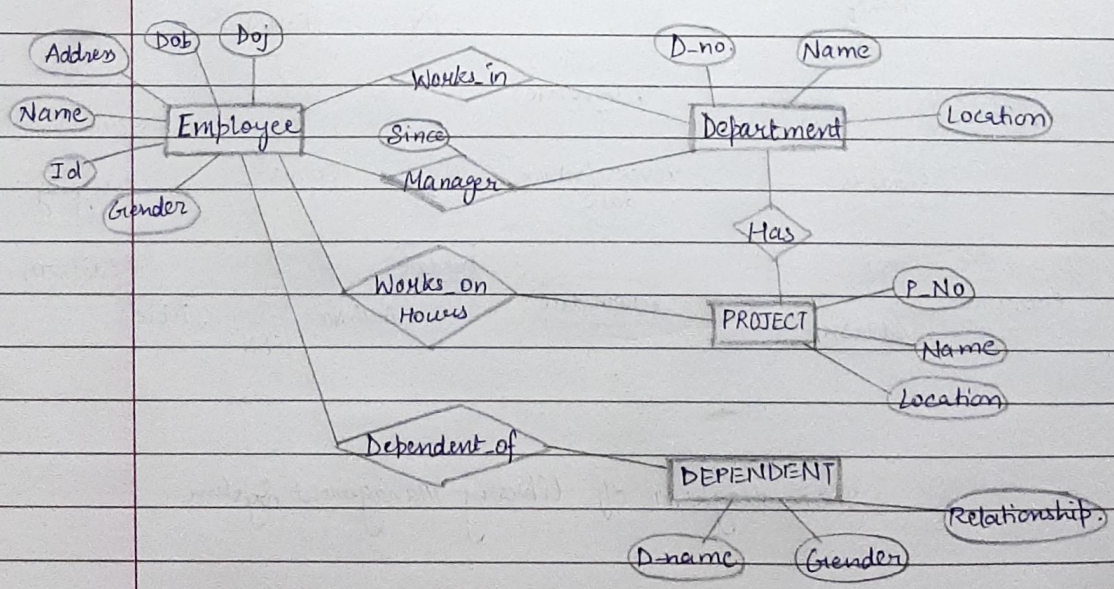


∴ ER diagram of Library Management System

2) ER Diagram of a company.

a) Company has several departments.
b) Each department may have several location.
c) Department are identified by a name, D.no, location.
d) Manager control a particular department
e) Each department is associated with number of projects.
f) Employees are identified by name, id, address, dob, date_of_joining
g) An employee works in only one department but can work on several projects.
h) We also keep track of number of hours worked by an employee on a single project.
i) Each employee has dependent.
j) Dependent has D-name, Gender and relationship.

ER Diagram of Company :



This company ER Diagram illustrates key information about Company, including entities such as employee, department, project and dependent. It allows to understand the relationships between entities.

→ Entities and their Attributes are

- Employee Entity : Attributes of Employee Entity are Name, Id, Address, Gender, Dob and Doj.
  Id is primary key and employee Entity.
- Department Entity: Attributes of Department Entity are D_no, Name and location. D_no is primary key for department Entity.
- Project Entity: Attributes of project Entity are P_No, Name and location. P_no is primary key for project Entity.
- Dependent Entity : Attributes of Dependent Entity are D_no., Gender and relationships.

→ Relationships are :

- Employees works in Departments –
  Many employee works in one Department but one Employee can not work in many departments.
- Manager controls a Department –
  Employee works under the manager of the department and the manager records the date of joining of employee in the department.
- Departments has many projects –
  One department has many projects but one project can not come under many departments.
- Employee works on project –
  One employee works on several projects and the number of hours worked by the employee on a single project is recorded.
- Employee has dependents –
  Each employee has dependents. Each dependent is dependent of only one employee

B) How many tables are created in given problem.

a)

M : M_1, M_2, M_3

P : P_1, P_2

N : N_1, N_2

Here, 3 tables are required.

b)

A : a_1, a_2

B : b_1, b_2

C : c_1, c_2

So, 3 tables are created.

c)

A - a_1, a_2

B - b_1, b_2

C - c_1, c_2

D - d_1, d_2

In this, 4 tables are created.

d)

E_1 : a_1, a_2

E_2 : b_1, b_2, b_3

E_3 : C_1, C_2

So, 3 tables are required

e) Account - Acc_no., Balance

Branch - b_name, b_city, Assets

Loan - L_no, Amt.

Customer - C_city, C_name, C_street.

Here, 4 tables are created.

Page No.:
Date:

Part-II

Q. Explain normalisation with example.

Ans Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like insertion, Update and Deletion Anomalies.
Normalization rules divides larger table into smaller tables and links them using relationships. The purpose of Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

The inventor of the relational model Edgar Codd proposed the theory of normalization of data with the introduction of the First Normal form, and he continued to extend theory with second and Third normal form. Later he joined Raymond F. Boyce to develop the theory of Boyce-Codd Normal form.

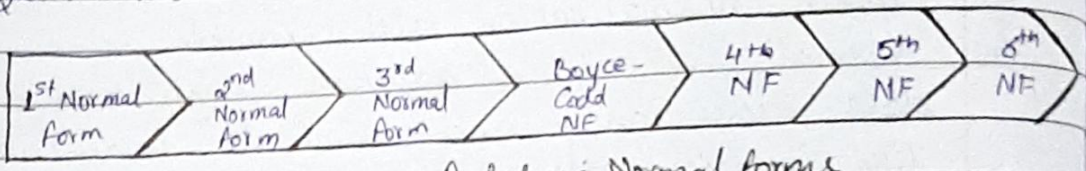
Database Normal Forms

Here is a list of Normal forms.

- 1NF (First Normal Form)
- 2NF (Second normal form)
- 3NF (Third normal form)
- BCNF (Boyce-Codd Normal form)
- 4NF (Fourth Normal form)
- 5NF (Fifth Normal form)
- 6NF (Sixth Normal Form)

The theory of Data Normalization in SQL server is still being developed further. For example, there are discussions even on 6th Normal form. However, in most practical applications, normalizations achieves its best in 3rd Normal form. The evolution of SQL

Normalization theories is illustrated below:-



: Database Normal forms

→ Database Normalization with examples.

Database Normalization Example can be easily understood with the help of a case study. Assume, a video library maintains a database of movies rented out. without any normalization in database, all information is stored in one table as shown below. Let's understand Normalisation in database with tables example:

FULL NAME	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATIONS
Janet Jones	First Street PLOT No. 4	Pirates of the Caribbean, Clash of the Titans.	Ms.
Robert Phill	3rd Street 34	forgetting sarah Marshal, Daddy's Little Girls	Mr.
Robert Phill	5th Avenue	Clash of the Titans	Mr.

Here you see Movies Rented column has multiple values. Now lets move into 1st Normal forms:-

- **1NF** (First Normal form) Rules.

→ Each table cell should contain a single value.
→ Each record needs to be unique.

The above table in 1NF Examples.

FULL NAME	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the titans	Mr.

Before we proceed lets understand a few things.

What is a KEY?

A key is a value used to identify a record in a table uniquely. A key could be a single column or combination of multiple columns.

Note:- Columns in a table that are NOT used to identify a record uniquely are called non-key columns.

What is a Primary key?

A primary is a single column value used to identify a database record uniquely.

It has following attributes.

- A primary key cannot be NULL.
- A primary key value must be unique.
- The primary key values should rarely be changed.
- The primary key must be given a value when a new record is inserted.

What is a Composite key?

A composite key is a primary key composed of multiple columns used to identify a record uniquely.

In our database, we have two people with the same name Robert Phil, but they live in different places.

We have introduced a new column called Membership_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership_id.

→ Database - Foreign key

In table 2, Membership_id is the foreign key.

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Carribean
1	Clash of the titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the titans

Foreign key references the primary key of another table! It helps connect your Tables.

→ A foreign key have a different name from its primary key.
→ It ensures rows in one table have corresponding rows is another.
→ Unlike the Primary key, they do not have to be unique. Most often they aren't.
→ Foreign keys can be null even though primary keys can not.

Foreign key

Membership_id	Movies Rented
1	Pirates of the Carribean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans.

→ Foreign key references Primary key.
→ Foreign key can only have values present in primary key.

→ It could have a name other than that of Primary key.

Primary Key

Membership_id	Full Names	Physical Address	Salutation
1	Janet Jones	First Street Plot No.4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Why do you need a foreign key?
Suppose, a novice inserts a record in Table B such as,

~~Insert a record in table2, where Member Id = 101~~

Membership id	movies Rented
101	Mission Impossible

But membership id 101 is not present in Table 1

Membership id	Full Names	Physical Address	Salutation
1	Janet Jones	First street Plot No.4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Database will throw an error. This helps in referential integrity.

You will only be able to insert values into your foreign key that exist in the unique key in the parent table. This helps in referential integrity.
The above problem can be overcome by declaring membership id from table 2 as foreign key of membership id from Table 1.
Now, if somebody tries to insert a value in the membership field that does not exist int the parent table, an error will be shown!

What are transitive functional dependencies?

A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change.

Consider the Table 1. Changing the non-key column Full Name may change Salutation.

Membership id	Full Name	Physical Address	Salutation
1	Janet Jones	First Street Plot No. 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

change in Name

May change Salutation

Let's move into 3NF.

- 3NF (Third Normal Form) Rules
  - Rule 1 - Be in 2NF
  - Rule 2 - Has no transitive functional dependencies.

To move our 2NF table into 3NF, we again need to again divide our table.

3NF Example

Below is a 3NF example in SQL databases:

Membership ID	Full Names	Physical Address	Salutation Id
1	Janet Jones	First Street Plot No. 4	2
2	Robert Phil	3rd Street 34	1
3	Robert Phil	5th Avenue	1

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Carribean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

SALUTATION ID	SALUTATION
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

We have again divided our tables and created a new table which stores Salutations.

There are no transitive functional dependencies, and hence our table is in 3NF.
In table 3 Salutation ID is primary key, and in Table 1 Salutation ID is foreign to primary key in Table 3.
Now our little example is at a level that cannot further be decomposed to attain higher normal forms of normalization. In fact, it is already in higher normalization forms. Separate efforts for moving into next levels of normalizing data are normally needed in complex databases. However, we will be discussing next levels of normalizations in brief.

- BCNF (Boyce - Codd Normal Form)

Even when a database is in 3rd Normal form, still there would be anomalies resulted if it has more than one candidate key. Sometimes is BCNF is also refferred as 3.5 Normal form

Page No.:
Date:

- 4NF (Fourth Normal Form) Rules
  If no database table instance contains two or more, independent and multivated data describing the elevant entity, then it is in 4th Normal form.

- 5NF (Fifth Normal Form) Rules
  A table is in 5th Normal form only if it is in 4NF and it cannot be decomposed into any number of smaller tables without loss of data.

- 6NF (Sixth Normal Form) Rules proposed -
  6th Normal form is not standardized, yet however, it is being discussed by database experts for sometime. Hopefully, we would have a clear & standardized definition for 6th Normal form in the near future.