NAME:- Bidisha Chirng Baruah
ROLL NO:- 200102022
BRANCH:- IT
SEMESTER:- 5th sem

Assignment_2

Q1) Explain the various cardinalities associated with an ER diagram
<u>Answer</u>:-There are 4 types of cardinality ratios-

1. Many-to-Many Cardinality: By this cardinality constraint.
   a) An entity in set A can be associated with any number (zero or more) of entities in set B.
   b) An entity in set B can be associated with any number (zero or more) of entities in set A.
2. Many-to-One Cardinality: By this cardinality constraint.
   a) An entity in set A can be associated with at most one entity in set B.
   b) An entity in set B can be associated with any number (zero or more) of entities in set A
3. One-to-Many Cardinality: By this cardinality constraint.
   a) An entity in set A can be associated with any number (zero or more) of entities in set B.
   b) An entity in set B can be associated with at most one entity in set A.
4. One-to-One Cardinality: By this cardinality constraint.
   a) An entity in set A can be associated with at most one entity in set B.
   b) An entity in set B can be associated with at most one entity in set A.

Q2) Explain Generalisation and Specialisation with Constraints.

<u>Answer:</u>- It works on the principle of bottom up approach. In generalisation lower level functions are combined to form higher level functions which are called as entities. This process is repeated further to make advanced level entities.

We can say that Specialisation is the opposite of Generalisation. In Specialisation things are broken down into smaller things to simplify it further. We can also say that in Specialisation a particular entity gets divided into sub entities and it's done on the basis of its characteristics. Also in Specialisation Inheritance takes place.

The different types which we need to consider while designing generalisation and specialisation in the Database Management System (DBMS) are as follows :

a)Conditional Definition:Create one database, and keep conditions on one attribute for example attendance. This type of constraint is defined on a single attribute which is further dividing an entity into two sub entity sets which will give information for the given attribute.

b)**User Defined**: In this constraint the decision is left to the super class that how many instances of the super class will be participating in the sub class.

c)Disjoint Constraints: Disjoint is nothing but intersection, the number of instances specified for the given superclass can participate in only one of the sub classes.Account users can participate in saving account and current account but both are different so, it can be participated one at a time.

d)Overlapping Constraint: Two or more instances of the super class are participating in two or more sub classes then it is called overlapping constraints.A person who knows Java and PHP can participate in both teams.

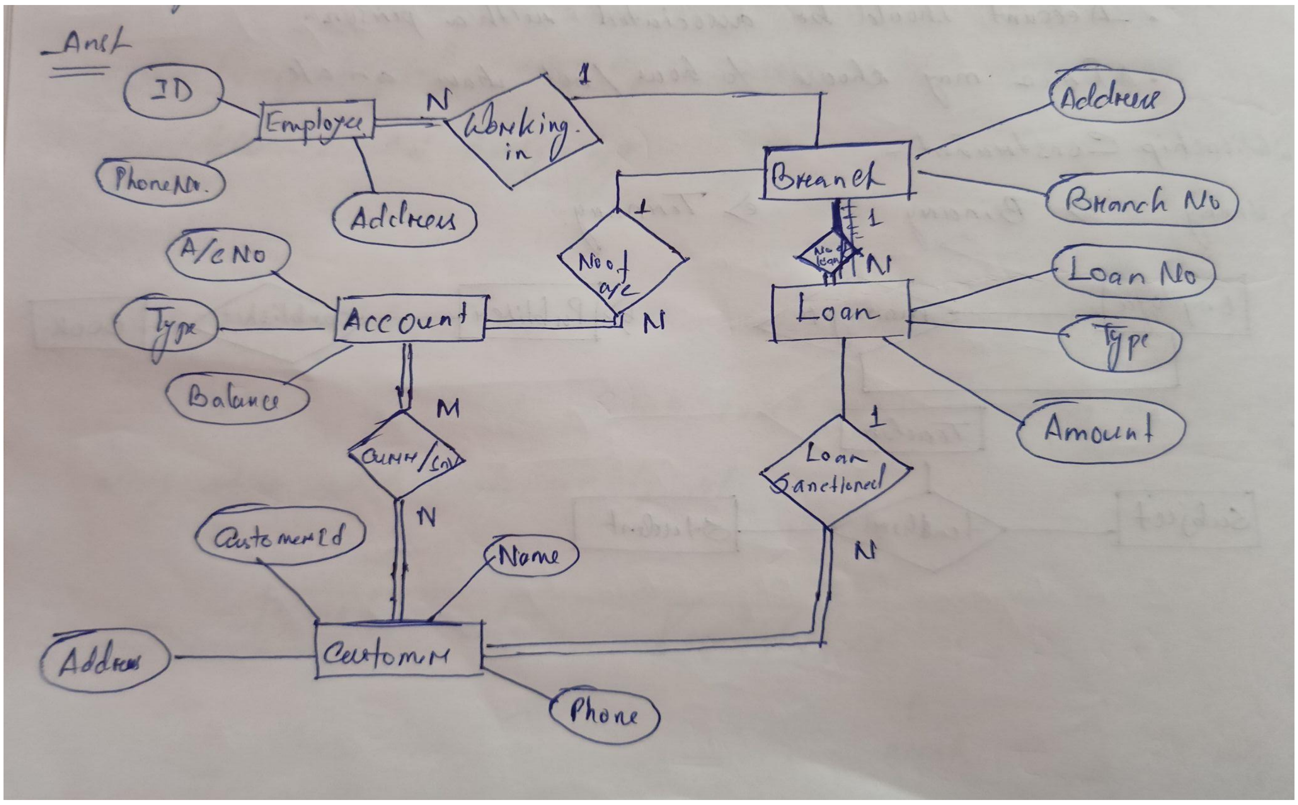
e)Complete Constraint: Every instance participates in a relationship. All the instances of the superclass must participate in a relationship or into the sub class.

f)Attribute Defined:This refers to specifying conditions on more than one attribute. Consider a database for the marks and attendance.This type of constraint is defined on two or more attributes for the given entity which is further divided into subclass entities.

Q3) Draw the ER diagram for the following:

Consider a BANK database having customer,loan,account,employee and branch as entity types.Each branch allows customers to open accounts and borrow loans. A customer can open more than one account , and one account may belong to more than one customer.Similar is the case for loans to.The bank has a number of employees working in different branches of the bank.Add appropriate attributes for each entity, represent the key attributes , cardinality ratios and weak entity sets (if any).

__Answer__:-



Q4) Define the term data integrity and explain its types.

Answer:-Data integrity is the overall accuracy, completeness, and consistency of data. Data integrity also refers to the safety of data in regard to regulatory compliance — such as GDPR compliance — and security. It is maintained by a collection of processes, rules, and standards implemented during the design phase. When the integrity of data is secure, the information stored in a database will remain complete, accurate, and reliable no matter how long it's stored or how often it's accessed.

Types of Data Integrity:

a) **Physical integrity** is the protection of the wholeness and accuracy of that data as it's stored and retrieved. When natural disasters strike, power goes out, or hackers disrupt database functions, physical integrity is compromised. Human error, storage erosion, and a host of other issues can also make it impossible for data processing managers, system programmers, applications programmers, and internal auditors to obtain accurate data.

b) **Logical integrity** keeps data unchanged as it's used in different ways in a relational database. Logical integrity protects data from human error and hackers as well, but in a much different way than physical integrity does. There are four types of logical integrity:

c) **Entity integrity:** Entity integrity relies on the creation of primary keys — the unique values that identify pieces of data — to ensure that data isn't listed more than once and that no field in a table is null. It's a feature of relational systems which store data in tables that can be linked and used in a variety of ways.

d) **Referential integrity:** Referential integrity refers to the series of processes that make sure data is stored and used uniformly. Rules embedded into the database's structure about how foreign keys are used ensure that only appropriate changes, additions, or deletions of data occur. Rules may include constraints that eliminate the entry of duplicate data, guarantee that data entry is accurate, and/or disallow the entry of data that doesn't apply.

e) **Domain integrity:** Domain integrity is the collection of processes that ensure the accuracy of each piece of data in a domain. In this context, a domain is a set of acceptable values that a column is allowed to contain. It can include constraints and other measures that limit the format, type, and amount of data entered.

f) **User-defined integrity:** User-defined integrity involves the rules and constraints created by the user to fit their particular needs. Sometimes entity, referential, and domain integrity aren't enough to safeguard data. Often, specific business rules must be taken into account and incorporated into data integrity measures.

Q5) Explain not null constraint.How does it enforce domain integrity.

<u>Answer:</u>-Every attribute has a nullability characteristic that shows whether a particular attribute accepts a null value or not. By default, every attribute accepts null values but this default nullability characteristic can be overridden by using the not null constraint. The not null constraint ensures that the attribute must not accept null values. For example, consider a tuple in the BOOK relation where the attribute ISBN contains a null value. Such a tuple provides book information for an unknown ISBN and, hence, it does not provide appropriate information. In such a case, null value must not be allowed and this can be done by constraining the attribute ISBN using not null constraint. Here, the not null constraint prevents the null value to be inserted into the attribute ISBN. Any attempt to change the attribute value of ISBN to null value results in an error. The null values are not allowed in the primary key attribute of a relation. Since ISBN is a primary key attribute of BOOK relation, it cannot accept null values. Hence, it is not necessary to explicitly specify not null constraint for the attribute ISBN.

The not null constraint enforces domain integrity by ensuring that the attribute of a particular domain is not permitted to take null values. For example, a domain, say dom2, can be restricted to take non-null values. If the domain dom2 is assigned to the attribute Category of a BOOK relation, it ensures that the attribute Category must have some values. As a result, if null value is inserted into the constrained attribute, it will not be accepted as it violates the not null constraint.