3.

A. Explain the different types of attributes with an appropriate example.

There are eight such types of attributes: Simple, Composite, Single-valued, Multi-valued, Derived, Stored and attribute. One more attribute is their, i.e. Complex Attribute, this is the rarely used attribute.

VVI

1. **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.

2. **Composite attribute :**

An attribute that can be split into components is a composite attribute.
Example: The name can be split into first name middle name, and last name.

3. **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.

4. **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.

5. **Derived attribute :**

An attribute that can be derived from other attributes is derived attributes.
Example: Total and average marks of a student.

6. **Stored attribute:** The stored attribute are those attribute which doesn't require any type of further update since they are stored in the database.
   Example: DOB(Date of birth) is the stored attribute.
7. **Key attribute:** Key attributes are those attributes that can uniquely identify the entity in the entity set.
   Example: Roll-No is the key attribute because it can uniquely identify the student.
8. **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.

B. What is the significance of normalization in database design? Describe different types of normalization with an appropriate example. VVI

- **Significance of normalization:**

- **Data Integrity:** Normalization helps maintain data integrity by reducing data redundancy.
- **Data Consistency:** By adhering to normalization rules, data consistency is improved.
- **Efficient Data Retrieval and Manipulation:** Normalization allows for efficient querying and manipulation of data.

Now, let's illustrate different types of normalization with appropriate examples by table:

Consider an example of a database for a library:

Table:

BookID	Title	Author	Genre	Publisher
1	Book A	Author X	Fiction	Publisher 1
2	Book B	Author Y	Non-fiction	Publisher 2
3	Book C	Author X	Fiction	Publisher 1

In the above table, we can observe that the Genre attribute is not atomic because it contains multiple values. To achieve 1NF, we separate the multivalued attribute into a separate table:

- **1NF:** remove repeating groups of data.

Table: Books (1NF corrected)

BookID	Title	Author	Publisher
1	Book A	Author X	Publisher 1

2	Book B	Author Y	Publisher 2
3	Book C	Author X	Publisher 1

Table: BookGenres (1NF corrected)

BookID	**Genre**
1	Fiction
2	Non-fiction
3	Fiction

- **Now, let's continue with 2NF:** remove partial dependency.

Table: Books (2NF example)

BookID	**Title**	**Author**	**Publisher**
1	Book A	Author X	1
2	Book B	Author Y	2
3	Book C	Author X	1

Table: Publishers (2NF example)

PublisherID	Publisher
1	Publisher 1
2	Publisher 2

In the 2NF example, we have split the publisher information into a separate table, and the Books table references the PublisherID as a foreign key.

- **Lastly, let's proceed to 3NF:** remove transitive dependency.

Table: Books (3NF example)

BookID	Title	AuthorID
1	Book A	1
2	Book B	2
3	Book C	1

Table: Authors (3NF example)

AuthorID	Author
1	Author X
2	Author Y

3	Author X

In the 3NF example, we have separated the author information into a separate table, and the Books table references the AuthorID as a foreign key.

By applying normalization upto 3NF in this example, we have reduced data redundancy, ensured data integrity, and improved data consistency in the database design. The tables are now organized in a logical and efficient manner, allowing for easier data retrieval and manipulation.

4

VVI

A. Define various types of keys with appropriate examples and distinguish them from one another.

- **Key:** A key refers to an attribute/a set of attributes that help us identify a row (or tuple) uniquely in a table (or relation).
- **Types of keys:**
  1. **Super key:** A super key is a group of single or multiple keys which identifies rows in a table.
     In the "Employees" table, EmployeeID, FirstName, LastName, Email, and Phone.
  2. **Primary Key –** is a column or group of columns in a table that uniquely identify every row in that table.
     In the “Employees” table, EmployeeID.
  3. **Candidate Key –**Candidate Key is a super key with no repeated attributes.
     In the "Employees" table, both EmployeeID and Email
  4. **Alternate key**: all those keys that did not become a primary key are known as alternate keys
     In the “Employees” table, Social Security Number (SSN)
  5. **Composite key:** A composite key would be the combination of two keys. For example: the combination of house number and street might qualify as a composite key
  6. **Foreign key: A** FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table
     DepartmentID of department table.
  7. **Unique key:** A unique key is a set of one or more than one fields/columns of a table that uniquely identify a record in a database table.
     In the "Employees" table, the Email attribute as a unique key.

B.

i. Why and in which case Triggers can be used? what system privileges are required to create a trigger on a table?

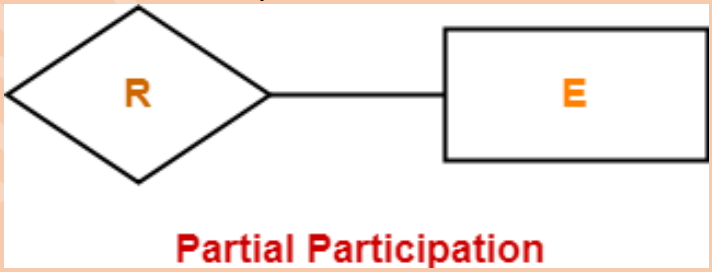
Because a trigger resides in the database and anyone who has the required privilege can use it, a trigger lets you write a set of SQL statements that multiple applications can use. It lets you avoid redundant code when multiple programs need to perform the same database operation.

In addition to the preceding privileges, to create a trigger on DATABASE, you must have the ADMINISTER DATABASE TRIGGER system privilege.
If the trigger issues SQL statements or calls procedures or functions, then the owner of the trigger must have the privileges necessary to perform these operations. These privileges must be granted directly to the owner rather than acquired through roles.

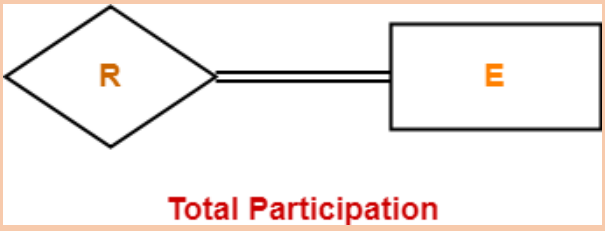
ii. Specify the total and partial participation in the database management system seen.

VVI

1. **Partial participation:**
   - ✓ It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set.
   - ✓ That is why, it is also called as optional participation.
   - ✓ Partial participation is represented using a single line between the entity set and relationship set.



2. **Total participation:**
   - ✓ It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set.
   - ✓ That is why, it is also called as mandatory participation.
   - ✓ Total participation is represented using a double line between the entity set and relationship set.



5.

A.

i. Define relationship and relationship set.

- **Relationship:** A relationship in databases is a situation where there is a logical association between two or more database tables.
- **Relationship set:** A collection of various relationships that belongs to the same relationship type is called a relationship set.

ii. If you want to Lock after five consecutive failed connection attempts to the JANE account, What will have to do in Oracle for this?

To lock the 'JANE' account after four consecutive failed connection attempts, we can use the following steps in Oracle: VVI

- Create a database profile that includes the FAILED_LOGIN_ATTEMPTS parameter set to 5 (or the desired number of attempts) and assign it to the user account 'JANE'.
- CREATE PROFILE jane_profile LIMIT FAILED_LOGIN_ATTEMPTS 5;
- ALTER USER jane PROFILE jane_profile;
- Enable the account lock feature in the database.
- ALTER SYSTEM SET FAILED_LOGIN_ATTEMPTS 5;

By setting the FAILED_LOGIN_ATTEMPTS parameter to 5, the database will automatically lock the 'JANE' account after five consecutive failed login attempts. The account will remain locked until it is manually unlocked by a privileged user.

Remember to replace 'JANE' with the actual account name you want to apply this lock on.

B. University wants to give scholarship to some students on the following criteria:

i. Students must be female.

ii. Student do not get any other private scholarships such like DBBL scholarship.

iii. Grade must be at least 3.50. VVI

iv. Student should not be punished for any awful activity.

Create necessary table and write necessary query for I,ii,iii,iv.

- **Create table:**

```
CREATE TABLE Students (
 student_id INT PRIMARY KEY,
 name VARCHAR(50),
 gender VARCHAR(10),
 scholarship VARCHAR(50),
 grade DECIMAL(4, 2),
 punishment VARCHAR(10)
);
```

- **Insert values:**

```
INSERT INTO Students (student_id, name, gender, scholarship, grade, punishment)
VALUES
 (1, 'Alice', 'Female', 'UGC scholarship', 3.75, 'No'),
 (2, 'Bob', 'Male', NULL, 3.90, 'No'),
 (3, 'Carol', 'Female', NULL, 3.40, 'Yes'),
 (4, 'David', 'Male', 'DBBL scholarship', 3.60, 'No'),
 (5, 'Eve', 'Female', NULL, 4.00, 'No');
```

- **Query:**

i. SELECT * FROM Students WHERE gender = 'Female';
ii. SELECT * FROM Students WHERE scholarship IS NULL;
iii. SELECT * FROM Students WHERE grade >= 3.50;
iv. SELECT * FROM Students WHERE punishment = 'No';

```
SELECT *
FROM Students
WHERE gender = 'Female'
  AND (scholarship IS NULL)
  AND grade >= 3.50
AND punishment = 'No';
```

July-December: 2019

4. A UGC wants to give scholarships to some students on the following criteria:

I. **Student must be female.**
II. **Student do not get any other private scholarships such like DBBL scholarship.**
III. **Grade must be at least 3.50**
IV. **Student should not be punished for nay awful activity.**

Create necessary table and write necessary query i,ii,iii and iv.

Previously noted.

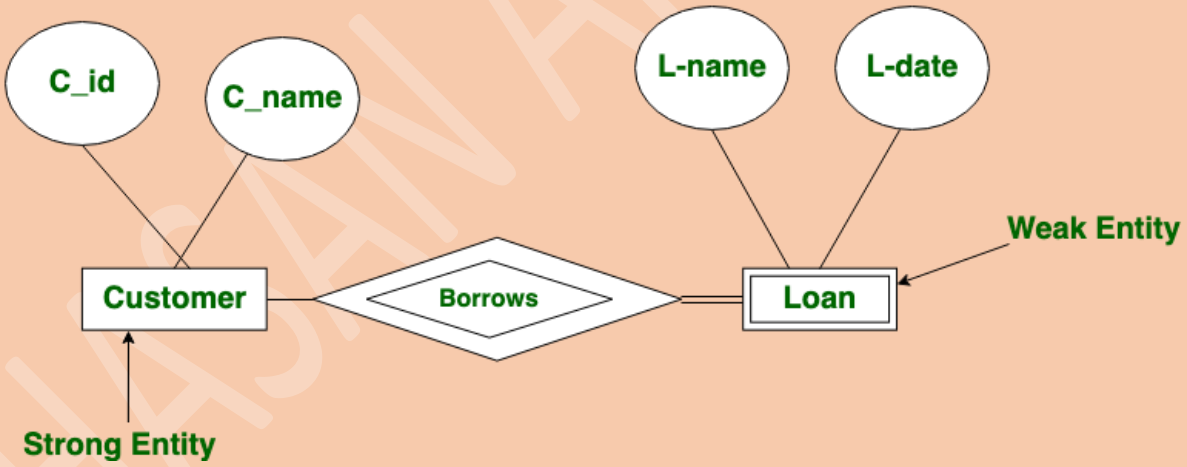
4 B Define weak entity set, Describe weak entity set with a suitable example.

The entity sets which do not have sufficient attributes to form a primary key are known as weak entity sets.

VVI

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 depend 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.

5 A Describe each line of the following Trigger in Oracle. VVI

i. **create or replace trigger BOOKSHELF_BEF_UPD_ROW:** This line starts the trigger definition and gives it a name BOOKSHELF_BEF_UPD_ROW.
ii. **before update on BOOKSHELF**: This line specifies that the trigger will be triggered before an update operation is performed on the BOOKSHELF table.
iii. **for each row:** This line indicates that the trigger is a row-level trigger, meaning it will be executed for each row affected by the update operation.
iv. **when (new.Rating < old.Rating):** This line specifies a condition that must be satisfied for the trigger to be executed. It checks if the new value of the Rating column is less than the old value of the Rating column.
v. **begin:** This line marks the beginning of the trigger's executable code.
vi. **insert into BOOKSHELF_AUDIT:** This line begins an INSERT statement to insert data into the BOOKSHELF_AUDIT table.
vii. **(Title, Publisher, CategoryName, Old_Rating, New_Rating, Audit_Date):** This line specifies the column names of the BOOKSHELF_AUDIT table that will receive the inserted data.
viii. **values (:old.Title, :old.Publisher, :old.CategoryName, :old.Rating, :new.Rating, Sysdate):** This line specifies the values to be inserted into the respective columns of the BOOKSHELF_AUDIT table. It uses the :old and :new keywords to refer to the old and new values of the corresponding columns in the BOOKSHELF table. Sysdate is a built-in function in Oracle that returns the current date and time.
ix. **end;:** This line marks the end of the trigger's executable code.
x. **/:** This line ends the trigger definition. Each line in the trigger serves a specific purpose, from defining the trigger name and event (before update on BOOKSHELF), specifying conditions (when (new.Rating < old.Rating)), to executing the desired code (insert into BOOKSHELF_AUDIT ...).

5 B What is normalization? Describe different types of normalization with appropriate example.

Previously noted.

6 A

i. **What is triggers? What are the types of Triggers?** VVI

- **Triggers** in Oracle are database objects that are automatically executed in response to specific events or changes in the database. They are used to enforce business rules, maintain data integrity, and automate certain actions. There are mainly three types of triggers in Oracle:

1. **DML Triggers:** These triggers are fired in response to Data Manipulation Language (DML) statements such as INSERT, UPDATE, and DELETE. DML triggers can be further classified into row-level triggers and statement-level triggers.
    - **Row-Level Triggers**: These triggers are executed once for each affected row in a DML statement. They can access the old and new values of the affected row.
    - **Statement-Level Triggers:** These triggers are executed once for each DML statement. They cannot access the old and new values of individual rows.
2. **DDL Triggers:** These triggers are fired in response to Data Definition Language (DDL) statements such as CREATE, ALTER, and DROP. DDL triggers can be used to track changes to database schema objects.
3. **Instead Of Triggers:** These triggers are used with views and are executed instead of the corresponding DML statements on the view. Instead Of triggers allow you to define custom logic for manipulating data through views.

ii. If you want to Lock after four executive failed connection attempts to the 'MASUD' account, what will have to do in oracle for this?

Previously noted with JANE user.

6 B What are the different types of attributes? Explain with suitable example.

Previously noted.

Session: 2012-2013

1 UGC wants to give scholarships to some students on the following criteria:

I. Student must be female.
II. Student do not get any other private scholarships such like DBBL scholarship.
III. Grade must be at least 3.50
IV. Student should not be punished for nay awful activity.
Create necessary table and write necessary query i,ii,iii and iv.

Previously noted.

2 A. Create a trigger that is executed whenever an insert or an update occurs. What tables you need to create the trigger? VVI

Here's an example of creating a trigger that is triggered on insert or update for a single table:

CREATE TRIGGER my_trigger

AFTER INSERT OR UPDATE ON my_table

FOR EACH ROW

```
BEGIN
  -- Trigger logic goes here
  -- You can access the inserted or updated data using the NEW alias
END;
```

In the above example, my_trigger is the name of the trigger, my_table is the table on which the trigger is created, and NEW is an alias that represents the newly inserted or updated row(s) in the trigger body. You can access the specific columns of the inserted or updated row using the NEW.column_name syntax.

If we want to create a trigger that applies to multiple tables, we need to create separate triggers for each table. Each trigger would have its own trigger logic specific to the table it is associated with.

2 B. Describe the different kinds of attributes with suitable example.

Previously noted.

3 A What are the significance of normalization in database design? Describe different types of normalization with appropriate example.

Previously noted.

3 B Create statement level BEFORE DELETE trigger on the BOOKSHELF table where conditions are:

When a user attempts to delete a record from the table. The trigger is executed and checks two system conditions: that the day of the week is neither Friday nor Saturday and the Oracle username of the account performing the delete begins with the letters "CSE".

VVI

```
CREATE OR REPLACE TRIGGER bookshelf_delete_trigger
BEFORE DELETE ON bookshelf
FOR EACH ROW
DECLARE
   day_of_week VARCHAR2(10);
   current_user VARCHAR2(30);
BEGIN
   -- Get the current day of the week.
```

```
    day_of_week := TO_CHAR(SYSDATE, 'DAY');

    -- Get the username of the user who is trying to delete the record.
    current_user := USER;

    -- Check if the day of the week is Friday or Saturday.
    IF day_of_week IN ('FRIDAY', 'SATURDAY') THEN
        RAISE_APPLICATION_ERROR(-20000, 'Deletions are not allowed on weekends');
    END IF;

    -- Check if the username of the user begins with "CSE".
    IF current_user NOT LIKE 'CSE%' THEN
        RAISE_APPLICATION_ERROR(-20001, 'Deletions are only allowed by CSE users');
    END IF;
END;
```

This trigger will be executed before any record is deleted from the bookshelf table. The trigger will first check if the day of the week is Friday or Saturday. If it is, then the trigger will raise an error message and prevent the deletion from happening. The trigger will then check if the username of the user who is trying to delete the record begins with the letters "CSE". If it does not, then the trigger will also raise an error message and prevent the deletion from happening.

4 A Describe each line of the following Trigger in Oracle.

Previously noted.

4 B "The cardinality ratio of a relationship can affect the placement of relationship attributes"----------justify the above statement. VVI

Cardinality ratio is a measure of how many entities of one type can be related to one entity of another type. There are three possible cardinality ratios:

- **One-to-one:** One entity of one type can be related to only one entity of another type. For example, a person can only have one driver's license.

- **One-to-many:** One entity of one type can be related to many entities of another type. For example, one book can have many authors.
- **Many-to-many:** Many entities of one type can be related to many entities of another type. For example, many students can be enrolled in many courses.

The cardinality ratio of a relationship can affect the placement of relationship attributes in a database schema. For example, in a one-to-one relationship, the relationship attributes would typically be placed in the entity that is the "one" side of the relationship. This is because there is only one instance of the relationship for each entity, so the relationship attributes do not need to be repeated in multiple entities.

In a one-to-many relationship, the relationship attributes would typically be placed in the entity that is the "many" side of the relationship. This is because there are many instances of the relationship for each entity, so the relationship attributes need to be repeated in multiple entities.

In a many-to-many relationship, the relationship attributes would typically be placed in a separate table. This is because there are many instances of the relationship for each entity, and the relationship attributes need to be shared by both entitie

July-Dec 2015

1

a) **"The cardinality ratio of a relationship can affect the placement of relationship attributes"----------justify the above statement.**

Previously noted.

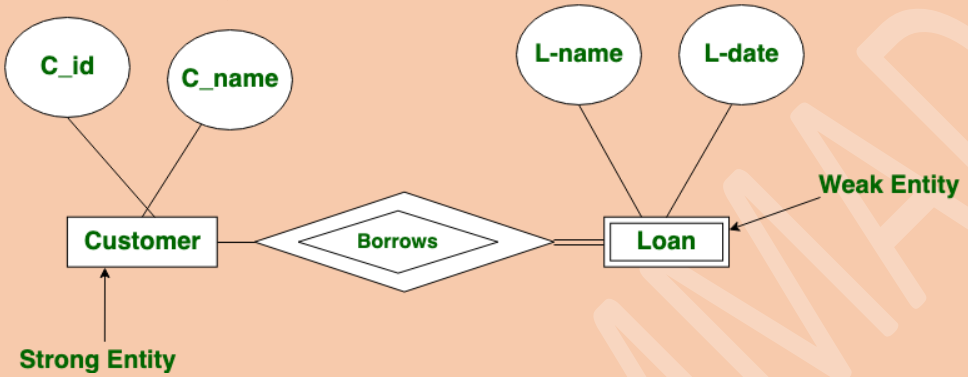
b) **What is weak entity set? How do you determine the primary key of a weak entity set? Show E-R Diagram of weak entity set.** VVI

weak entity set is an entity set in a relational database that does not have sufficient attributes to form a primary key on its own

To determine the primary key of a weak entity set, we need to consider two factors:
Partial Key: A weak entity set has a partial key, which is a set of attributes that, combined with the primary key of the strong entity set, uniquely identifies instances of the weak entity set. In the example above, the partial key of the "Enrollment" entity set might include attributes like "Course_ID" and "Student_ID".

Identifying Relationship: An identifying relationship is a relationship between a weak entity set and its associated strong entity set. It is used to establish the dependency of the weak entity set on the strong entity set. In the example, the identifying relationship between "Course" and "Enrollment" would indicate that an enrollment is identified by the combination of a course and a student.

E-R diagram of weak entity set:



c) Why the identifying relationship is many to one from weak entity set to identifying entity set and the participation of the weak entity set in the relationship is total?

In a relational database, the identifying relationship between a weak entity set and its associated strong entity set is typically represented as a many-to-one relationship from the weak entity set to the identifying entity set. This is because multiple instances of the weak entity set can be associated with a single instance of the identifying entity set.

Let's consider the example of a weak entity set called "Enrollment" and a strong entity set called "Course" in a university database. Each enrollment in the "Enrollment" entity set is related to a specific course in the "Course" entity set. However, multiple enrollments can be associated with the same course. For instance, many students can be enrolled in the same course.

Regarding the participation of the weak entity set in the identifying relationship, it is typically total. Total participation means that every instance of the weak entity set must be associated with an instance of the identifying entity set. In other words, every enrollment must be related to a course.

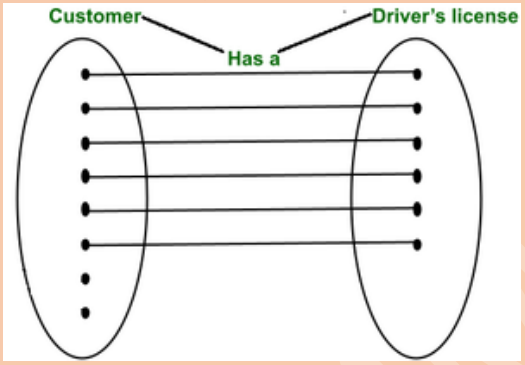
This total participation is necessary because the weak entity set depends on the existence of the strong entity set. Without the identifying entity set (Course in this case), there would be no basis for the existence of the weak entity set (Enrollment). The total participation ensures that every instance of the weak entity set has a corresponding instance of the identifying entity set, thus maintaining the integrity and completeness of the database.

2.

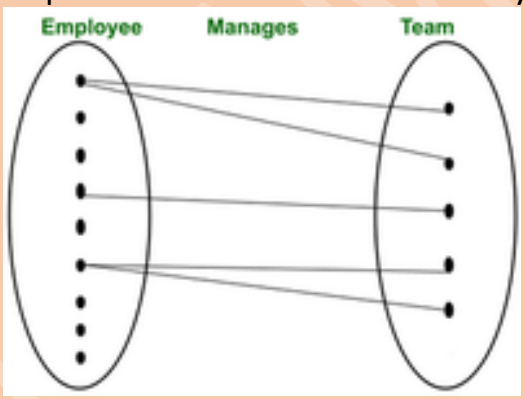
VVI

a) Briefly describe different types mapping cardinality and participation constraints.

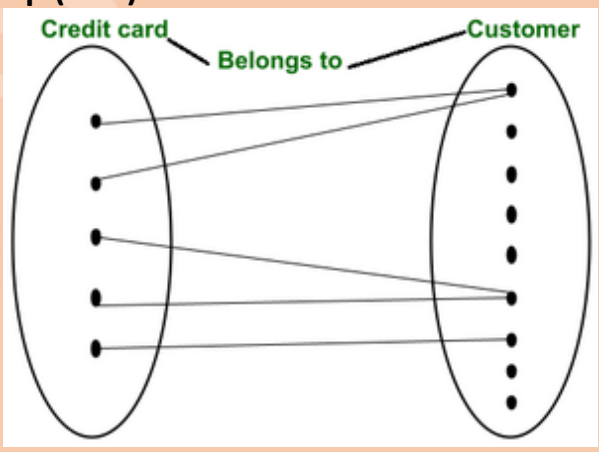
1) **One to one relationship(1:1) :** It is represented using an arrow(⇢,⇠)(There can be many notations possible for the ER diagram).



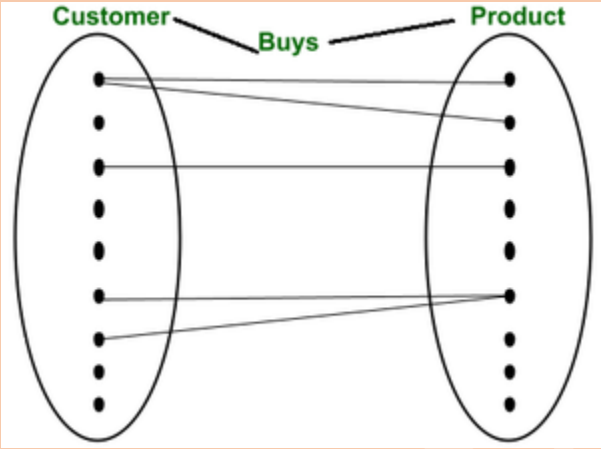
2) **One-to-Many (1:M):** In a one-to-many relationship, each instance of one entity set can be associated with multiple instances of the other entity set.



3) **Many to one relationship (M:1) :**



4) **Many-to-Many (M:N):** In a many-to-many relationship, each instance of one entity set can be associated with multiple instances of the other entity set.



Participation constraints previously noted.

b) "Every candidate key is a super key ~~but every~~ super key is not primary key"-Explain this statement with example. VVI

Let's break down the statement and explain it with an example:

1. **Candidate Key:** A candidate key is a minimal set of attributes that can uniquely identify each tuple or instance in a relation (table) within a database.
2. **Super Key:** A super key is a set of one or more attributes that can uniquely identify each tuple in a relation.
3. **Primary Key:** A primary key is a candidate key selected to uniquely identify each tuple in a relation

Now, let's illustrate the statement with an example:

Consider a relation (table) called "Students" with the following attributes: Student_ID, Name, Email, and Phone_Number.

In this case, the attribute "Student_ID" could be a candidate key since it uniquely identifies each student. However, the combination of attributes "Student_ID" and "Email" would also be a super key since it can uniquely identify each student.

So, in this example:

- Candidate Key: Student_ID
- Super Key: Student_ID, Email

Now, let's apply the statement to this example:

- **"Every candidate key is a super key":** This is true because the candidate key "Student_ID" is also a super key since it satisfies the uniqueness requirement.
- **"Every super key is not a primary key":** This is also true because the super key "Student_ID, Email" contains additional attributes beyond the minimal set required for uniqueness. Although it can uniquely identify each student, it is not chosen as the primary key.

3.

a) **Define normalization. What is the benefit of normalization in the design of database?**
b) **Normalize the following relation upto 3rd normal form.**

Previously noted

Session: 2015-2016

4

a) **What are the significance of normalization in database design?Describe different types of normalization with appropriate example.**

Previously noted.

b) **"The cardinality ratio of a relationship can affect the placement of relationship attributes"----------justify the above statement.**

Previously noted.

5

a) **Describe ins and out of weak entity set with a suitable example.**

Previously noted.

i. **If you want to lock after five consecutive failed connection attempts to JANE account, what will have to do in Oracle for this?**

Previously noted.

ii. **Define different types of keys with appropriate examples and distinguish among them.**

Previously noted.

6.

a) Describe the different kinds of attributes with suitable example.

Previously noted.

b)

i. What is Trigger? What are the types of Trigger?

Previously noted.

2 Create a trigger that is executed whenever an insert or an update occurs. What tables you need to create the Trigger?

Previously noted.