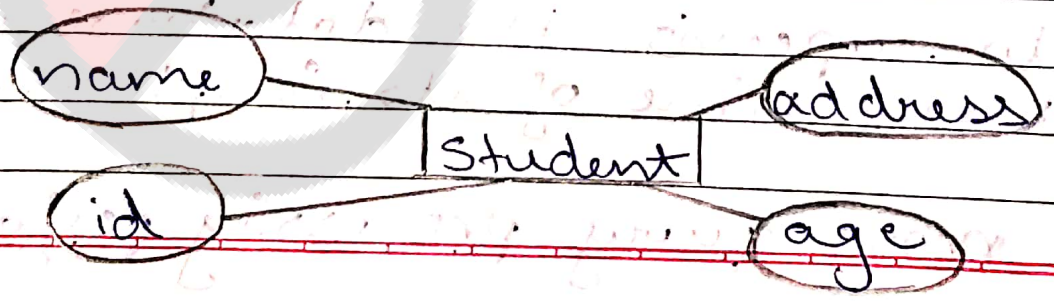
Unit 2

Q] Entity Relationship Model :-

ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age etc. The address can be another entity with attributes like city, street name, pin code etc. and there will be a relationship between them.



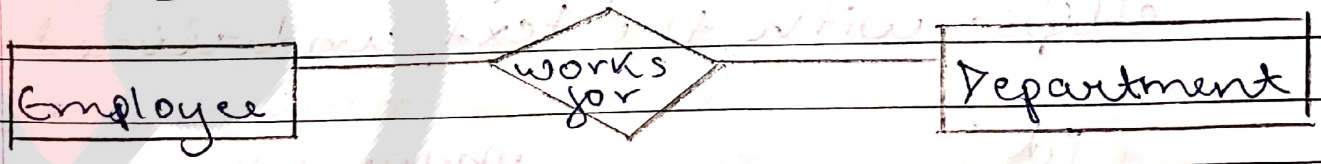
PAGE No.
DATE / /

Component of ER diagram

- ER Model
  - Entity
    - Weak Entity
  - Attribute
    - Key Attribute
    - Composite Attribute
    - Multivalued Attribute
    - Derived Attribute
  - Relation
    - One to one
    - One to many
    - Many to one
    - Many to many

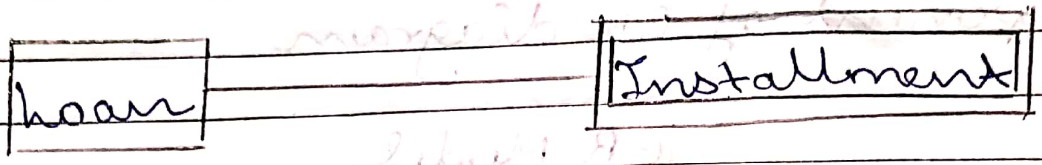
1) Entity:-

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
Consider an organization as an example - manager, product, employee, department etc can be taken as an entity



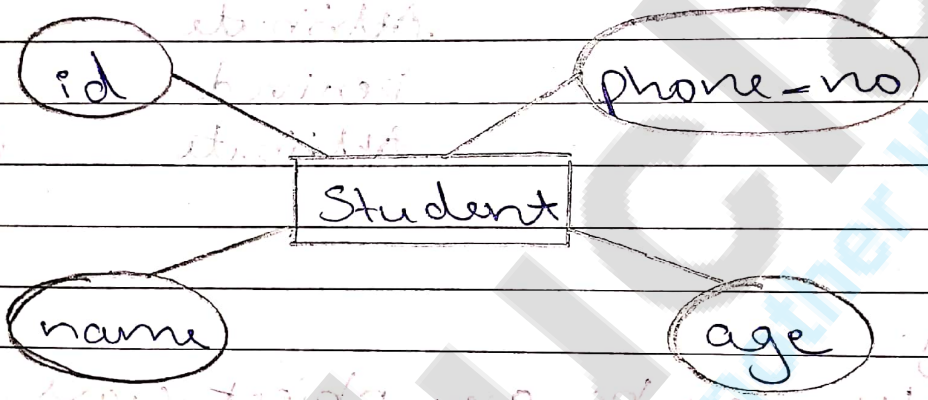
a) Weak Entity

An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



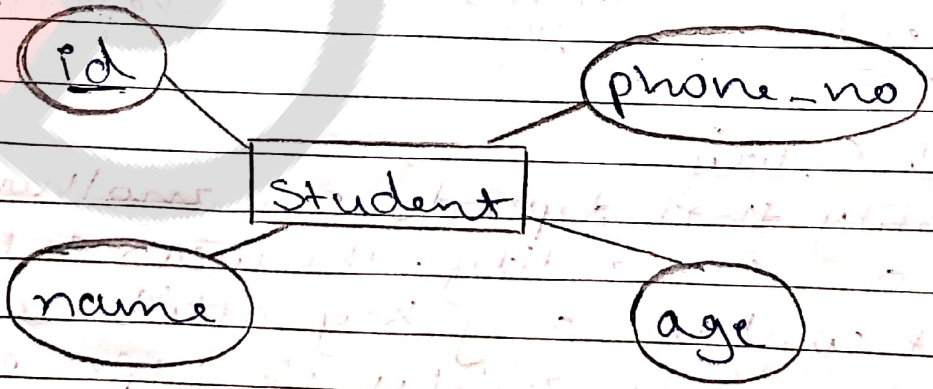
2) Attribute

The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.
For example, id, age, contact number, name etc can be attributes of a student.



a) Key Attribute

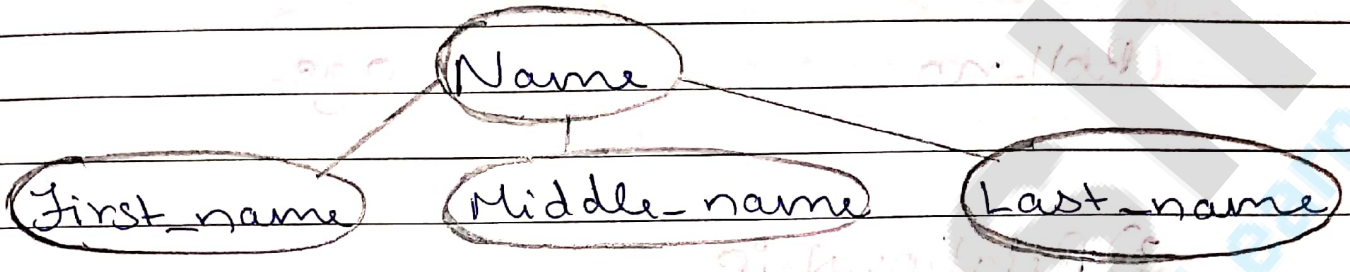
The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



b) Composite Attribute

An attribute that composed of many

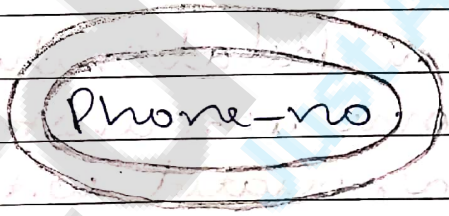
other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c) Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

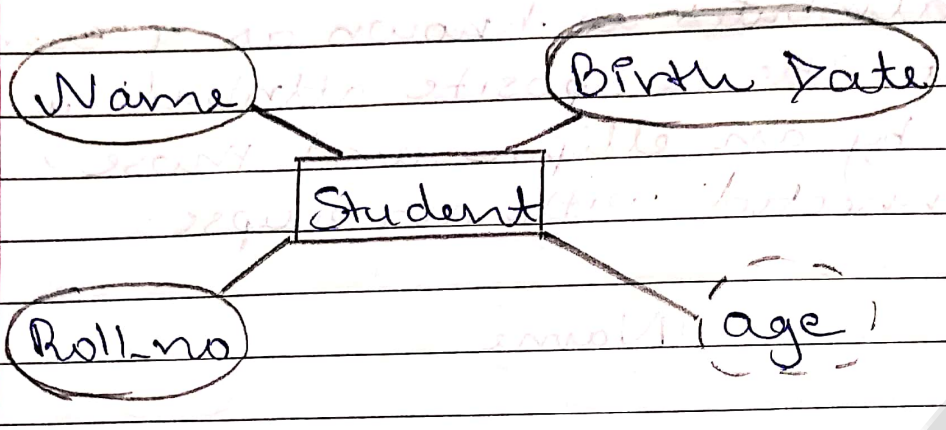
For example, a student can have more than one phone number.



d) Derived Attribute

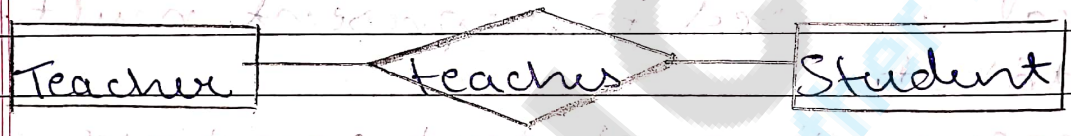
An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Data of birth.



3) Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

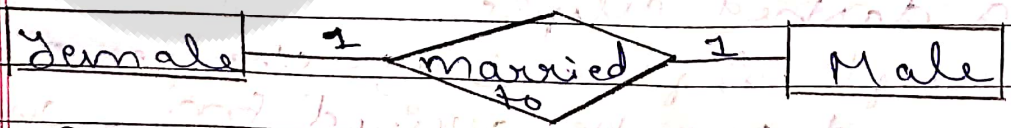


Types of relationship are as follows:

a) One-to-One Relationship

When only one ~~is~~ to instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example:- A female can marry to one male, and a male can marry to one female.

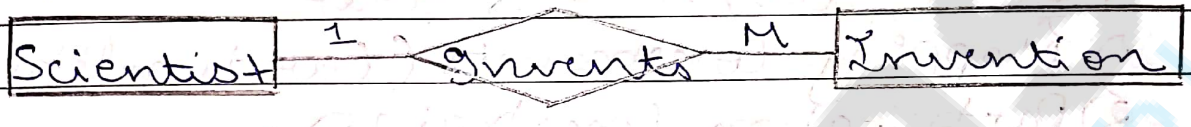


b) One-to-many relationship

When only one instance of the entity on the left, and more than

one instance of an entity on the right associates with to the relationship then this is known as a one-to-~~one~~ many relationship.

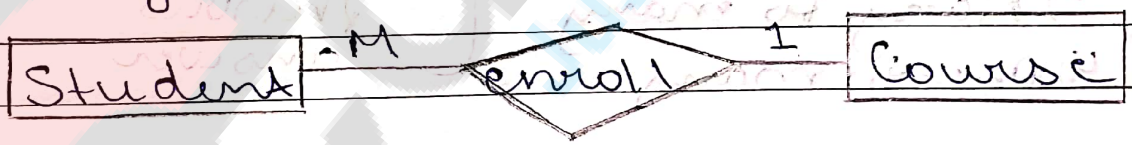
for example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



c) Many-to-~~one~~ relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

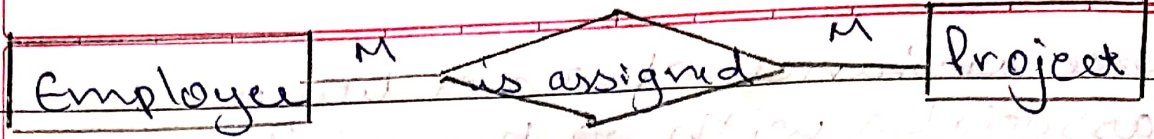
for example, Student enrolls for only one course, but a course can have many students



d) Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

for example, Employee can assign by many projects and project can have many employees.



Q) Constraints

An E-R enterprise schema may define certain constraints to which the contents of a database must confirm.

Types of constraints are:-

1) Mapping Constraints (Cardinalities)
2) Participation Constraints
3) Key Constraints

1) Mapping Constraints (Cardinalities):-

The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be different types:-

1) One to one
2) Many to one
3) Many to many
4) One-to many

} Explained in previous answer.

2) Participation Constraint:-

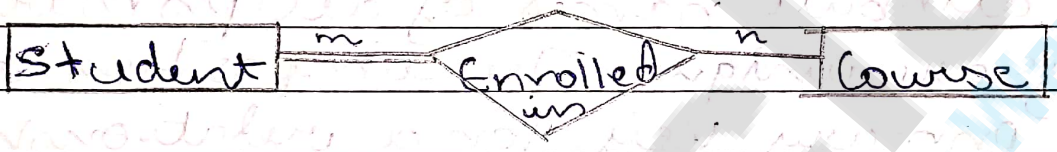
Participation Constraint is applied on the entity participating in the relationship set.

1) Total Participation:- Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of student will be total. Total participation is shown

by double line in ER diagram.

2) Partial Participation:- The entity in the entity set may or may not participate in the relationship. If some courses are not enrolled by any of the student, the participation of course will be partial.

The diagram depicts the 'Enrolled in' relationship set with Student Entity set having total participation and Course Entity set having partial participation.



3) Key Constraints

The values of the attribute values of an entity must be such that they can uniquely identify the entity. In other words no two entities in an entity set are allowed to have exactly the same value for all attributes.

The notion of a key for a relation schema applies directly to entity sets. That is, a key for an entity is a set of attributes that suffice to distinguish entities from each other.

The concepts of super key, candidate key and primary key are applicable to entity sets just as they are applicable to relation schemas.

Keys also help to identify relationships

uniquely and thus distinguish relationships from each other. Below, we define the corresponding notions of keys for relationships.

The primary key of an entity set allows us to distinguish among the various entities of the set. Let R be a relationship set involving entity set $E_1, E_2 \ldots E_n$. Let primary-key(E_i) denote the set of attributes that forms the primary key for entity set E_i. Assume for now that the attribute names of all primary keys are unique. The composition of the primary key for a relationship set depends on the set of attributes associated with the relationship set R.

If the relationship set R has no attributes associated with it, then the set of attributes primary-key(E_1) ∪ primary-key(E_2) ∪ ... ∪ primary-key(E_n) describes an individual relationship in set R.

If the relationship set R has attributes $a_1, a_2, \ldots a_m$ associated with it, then the set of attributes primary-key(E_1) ∪ primary-key(E_2) ∪ ... ∪ primary-key(E_n) ∪ $(a_1, a_2 \ldots a_m)$ describes an individual relationship in set R.

In both of the above cases, the set of attributes primary-key(E_1) ∪ primary-key(E_2) ∪ ... ∪ primary-key(E_n) forms

a super key for the relationship set. If the attribute names of primary keys are not unique across entity sets, the attributes are renamed to distinguish them; the name of the entity set combined with the name of the attribute would form a unique name.

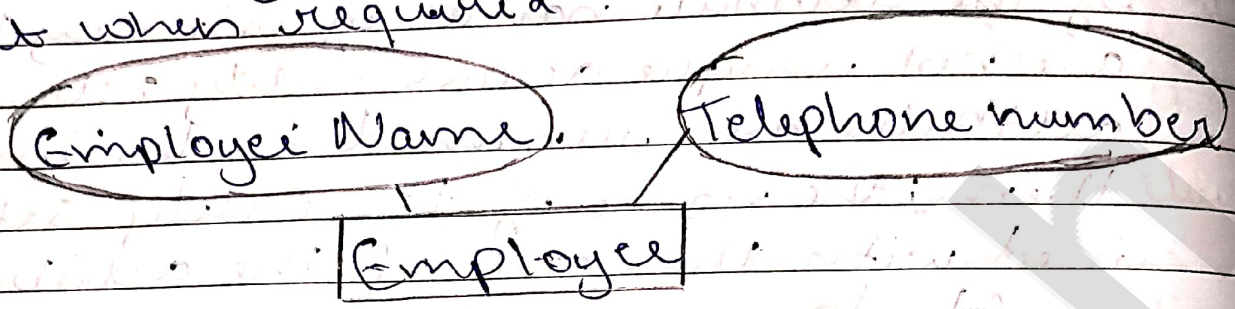
9] Entity-Relationship Design Issues

Basic issues in the design of an ER database schema are:-

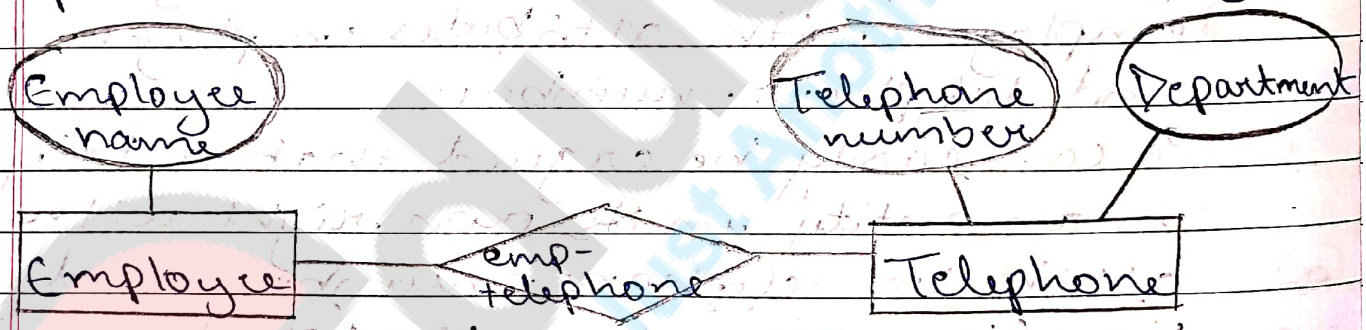
1) Use of entity set vs attributes :- In these cases we treat an attribute as an entity for example- Consider the entity set employee with attributes employee-name and telephone-number. It can easily be argued that a telephone is an entity in its own right with attributes telephone-number and location (the office where the telephone is located). If we take this point of view, we must redefine the employee entity set as:

1) The employee entity set with attribute employee-name
2) The telephone entity set with attributes telephone-number and location
3) The relationship set emp-telephone, which denotes the association between employees and the telephones that they have. Such a conversion of attribute

helps to give extra information about it when required.



Here it should be noted that the attribute Telephone number itself may contain several sub attributes like the corresponding Department of the telephone number, thus it become difficult to manage such kind of attributes in simple ER Diagram, this problem is addressed in below diagram.



Using attribute as Entity

This figure depicts, how an Entity is attribute can be treated as entity, here note that the entity EMPLOYEE is broken into two different entities namely Employee and Telephone, further a relationship emp-telephone is defined between these two entities.

2) Use of Entity Sets vs Relationship Sets -

At several instances, it is not always clear whether an object can be best expressed by an entity set or a relationship set. This approach is used to break such dilemma where a set of relationship is defined between entities with designated action that has to occur between them. This approach can also be useful in deciding whether certain attributes may be more appropriately expressed as relationships.

Consider a bank database as given below:-

321-12-3123	Jones	Main	Harrison		L-17	1000
019-28-3746	Smith	North	Rye		L-23	2000
677-89-9011	Hayes	Main	Harrison		L-15	1500
555-55-5555	Jackson	Dupont	Woodside		L-14	1500
244-66-8800	Curry	North	Rye		L-19	500
963-96-3963	Williams	Nassau	Princeton		L-11	900
335-57-7991	Adams	Spring	Pittsfield		L-16	1300

Customer

Attributes (AccNo, CustomerName, CustomerStreet, CustomerCity)

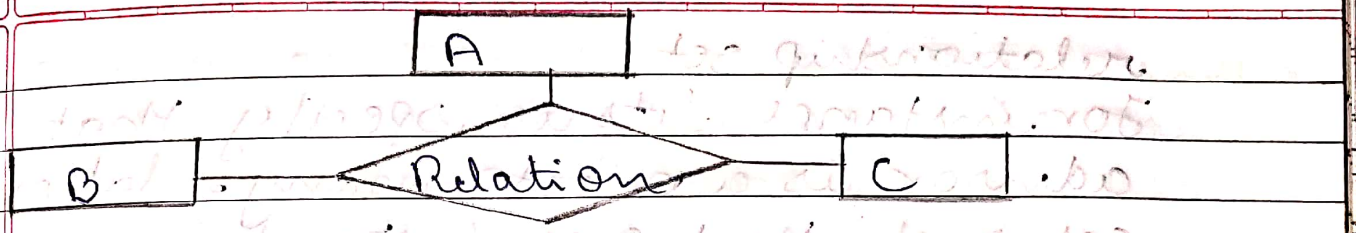
Loan

Attributes (LoanNo, Amount)

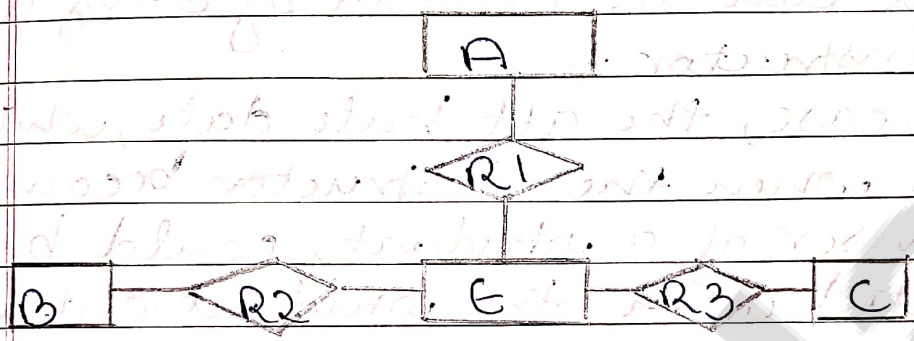
We assumed that a bank is modled as an entity. An alternative is to model a loan not as an entity, but rather as a relationship between customers and branches, with loan-number and amount as descriptive attributes. Each loan is represented by a relationship between a customer and

a branch. If every loan is held by exactly one customer and is associated with exactly one branch, we may find satisfactory the design where a loan is represented as a relationship. However, with this design, we cannot represent conveniently a situation in which several customers hold a loan jointly. To handle such a situation, we must define a separate relationship for each holder of the joint loan. Then, we must replicate the values for the descriptive attributes loan-number and amount in each such relationship. Each such relationship must of course, have the same value for the descriptive attributes loan_number and amount.

3) Binary versus nary Relationship Sets- In most cases the relationships in database system is binary. which means it defines the relationship between two entities. Sometimes a system may have non binary (more than 2) relations, under such design those non binary relations are broken down to several binary relationships which in turn gives an easy and better representation of the system.



Ternary Relation



Binary Relation

For simplicity, consider the abstract, ternary (n=3) relationship set R, relating entity sets A, B and C (Fig 1). We replace the relationship set R by an entity set E, and create three relationship sets:

- R1, relating E and A.
- R2, relating E and B
- R3, relating E and C.

and thus the ternary relation is converted into binary relation.

4) Placement of Relationship Attributes - The cardinality ratio of a relationship can affect the placement of relationship attributes. Thus, in such cases attributes of one-to-one or one-to-many relationship sets can be associated with one of the participating entity sets, rather than with the

relationship set.
For instance, lets us specify that advisor is a one-to-many relationship set such that one instructor may advise several students, but each student can be advised by only a single instructor.
In this case, the attribute date, which specifies when the instructor became the advisor of a student, could be associatid with the student entity set.
Since each student entity participates in a relationship with at most one instance of instructor, making this attribute designation has the same meaning as would placing data with the advisor relationship set.
Attributes of a one-to-many relationship set can be repositioned to only the entity set on the "many" side of the relationship. For one-to-one relationship sets, on the other hand, the relationship attribute can be associated with either one of the participating entities.
The design decision of where to place descriptive attributes in such cases as a relationship or entity attribute should reflect the characteristics of the enterprise being modeled.
The designer may choose to retain

date as an attribute of advisor to express explicity that the date refers to the advising relationship and not some other aspect of the student's university status (for example date of acceptance to the university).

The choice of attribute placement is more clear-cut for many-to-many relationship sets. Returning to our example, let us specify the perhaps more realistic case that advisor is a many-to-many relationship set expressing that an instructor may advise one or more students and that a student may be advised by one or more instructors.

If we are to express the date on which a specific instructor became the advisor of a specific student, date must be an attribute of the advisor relationship set, rather than either one of the participating entities.

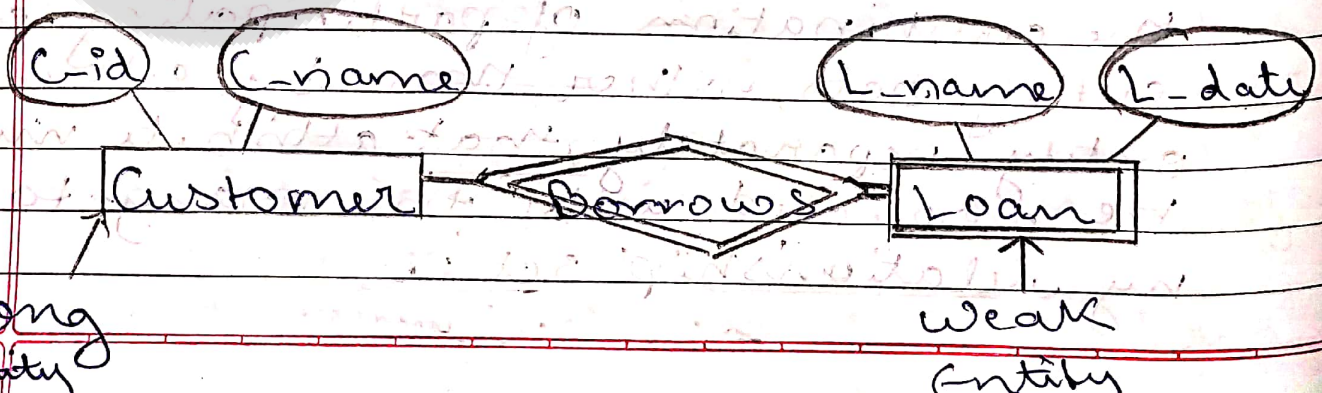
If date were an attribute of student, for ~~ste~~ instance we could not determine which instructor become the advisor on that particular date.

When an attribute is determined by the combination of participating entity sets, rather than by either entity separately, that attribute must be associated with the many-to-many relationship set.

PAGE No.
DATE / /

Q.] Difference between Strong and Weak Entity.

Strong Entity	Weak Entity
1) Strong entity always has primary key.	While weak entity has partial discriminator key.
2) Strong entity is not dependent of any other entity.	Weak entity is dependent on strong entity.
3) Strong entity is represented by single rectangle.	Weak entity is represented by double rectangle.
4) Two Strong entity's relationship is represented by single diamond.	While relation between one strong and one weak entity is represented by double diamond.
5) Strong entity have either total participation or not.	While weak entity always has total participation.

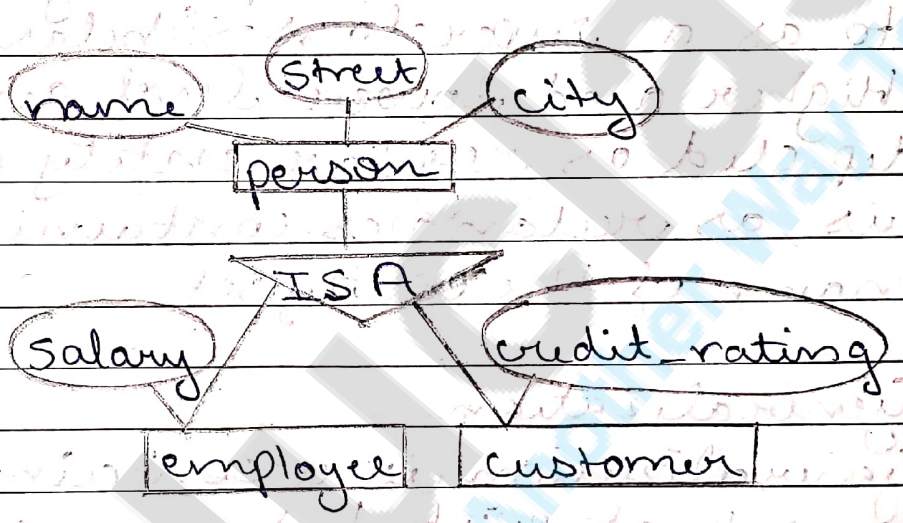


PAGE No.
DATE / /

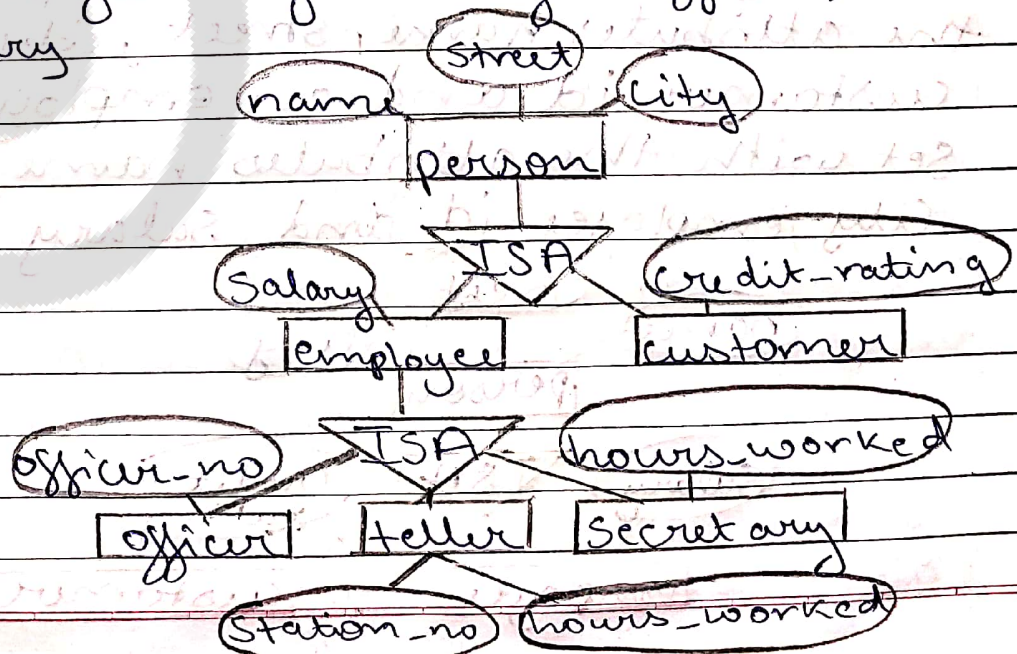
Q) Extended ER features:

Specialization

The process of designating sub-groupings within an entity set is called specialization. Specialization is depicted by a triangle component labelled ISA. The specialization of person allows us to distinguish among persons according to whether they are employees or customers



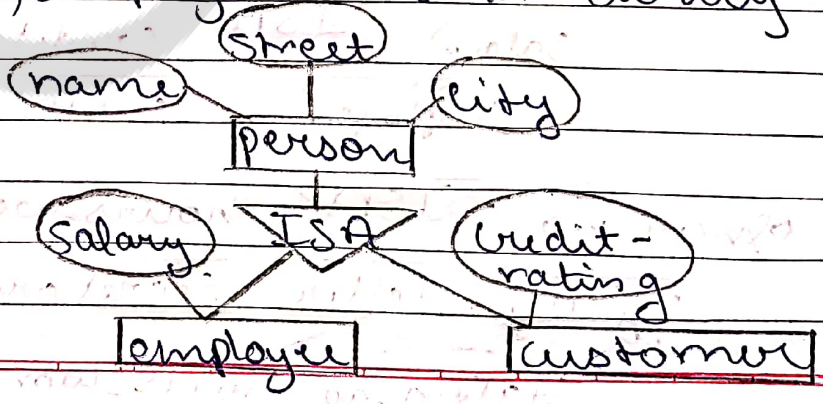
We can apply specialization repeatedly to refine a design scheme. For instance, bank employees may be further classified as one of the following :- officer, teller, secretary



An entity set may be specialized by more than one distinguishing feature. In our example, specialization could be based on whether the person is a temporary employee or a permanent employee, resulting in the entity sets temporary-employee and permanent employee.
The ISA relationship may also be referred to as a superclass-subclass relationship. Higher and lower-level entity sets are depicted as regular entity a set - that is, as rectangles containing the name of the entity set.

Generalization

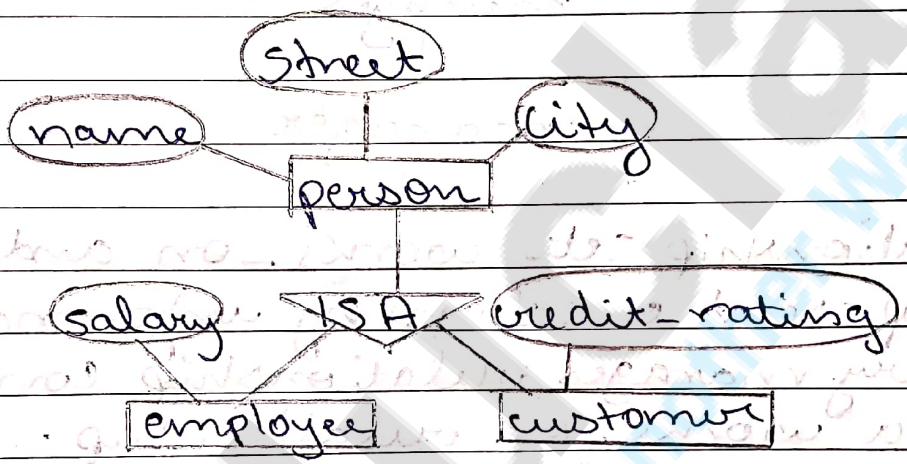
Generalization is a containment relationship that exists between a higher-level entity set and one or more lower level entity sets.
The database designer may have first identified a customer entity set with the attributes name, street, city and customer-id and an employee entity set with the attributes name, street, city, employee-id and salary



PAGE No.
DATE / /

Attribute Inheritance

A crucial property of the higher and lower-level entities created by specialization and generalization is attribute inheritance.
The attributes of the higher-level entity sets are said to be inherited by the lower-level entity sets.
For example, customer and employee inherit the attributes of person.



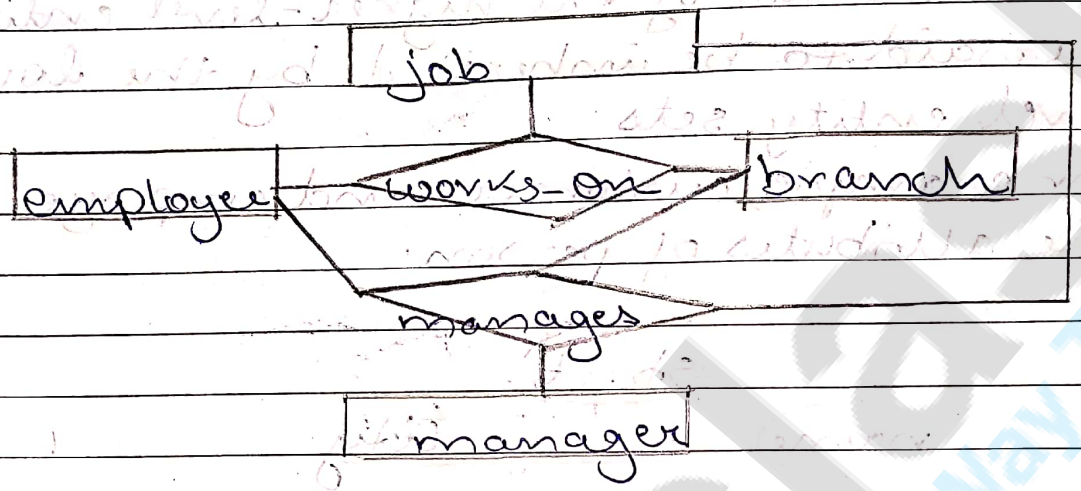
A lower-level entity set (or subclass) also inherits participation in the relationship sets in which its higher-level entity (or superclass) participates.
If an entity set is a lower-level entity set in only one ISA relationship, then the entity set has single inheritance.
If an entity set is a lower-level entity set is more than one ISA relationship, then the entity set has multiple inheritance, and the resulting structure is said to be a lattice.

Aggregation

Consider the ternary relationship ~~work~~.

works_on

Suppose we want to record managers for tasks performed by an employee at a branch.



Relationship sets works_on and manages represent overlapping information.

Every manages relationship corresponds to a works_on relationship.

However, some works_on relationships may not correspond to any manages relationships

So we can't discard the works_on relationship

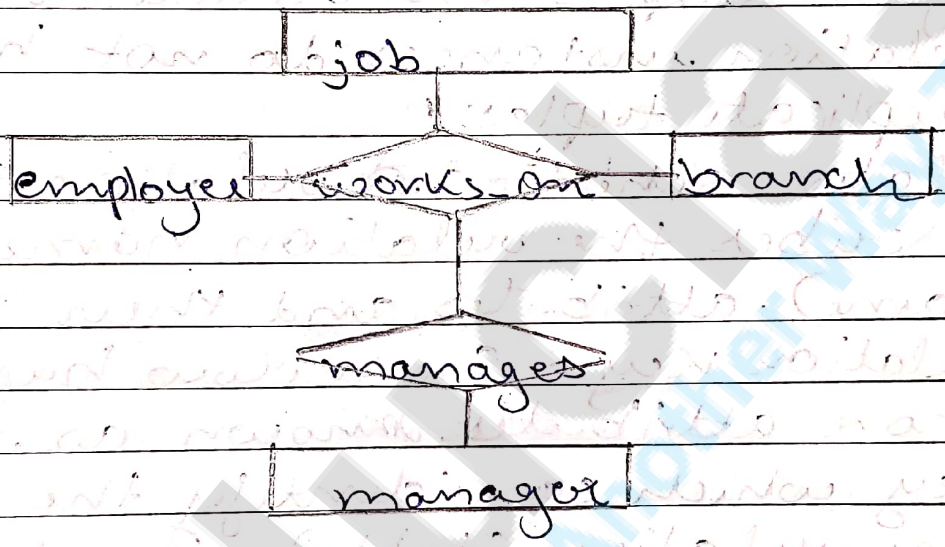
Eliminate this redundancy via aggregation

- Treat relationship as an abstract entity
- Allows relationship between relationships.
- Abstraction of relationship into new entity.

Without introducing redundancy, the following diagram represents:

- An employee works on a particular

branch
An employee, branch, job combination may have an associated manager
The best way to model a situation is to use aggregation.
Aggregation is an abstraction through which relationships are treated as higher level entities.



Q) Relational Data model:-
Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and capabilities required to process data with storage efficiency.
Concepts
Tables - In relational data-model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows

and columns, where rows represents records and columns represent the attributes.

Tuple - A single row of a table, which contains a single record for that relation is called a tuple.

Relation instance - A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.

Relation schema - A relation schema describes the relation name (table name), attributes, and their names.

Relation key - Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.

Attribute domain - Every attribute has some pre-defined value scope, known as attribute domain.

Constraints

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called Relational Integrity Constraints. There are three main integrity constraints.-

1) Key constraints
2) Domain constraints
3) Referential integrity constraints

PAGE No.
DATE / /

Key Constraints

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called key for that relation. If there are more than one such minimal subsets, these are called candidate keys.

Key constraints force that -

- in a relation with a key attribute, no two tuples can have identical values for key attributes.
- a key attribute can not have NULL values.
- Key constraints are also referred to as Entity Constraints.

Domain Constraints

Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

Referential Integrity Constraints

Referential integrity constraints work on the concept of foreign keys.

A foreign key is a key attribute of a relation that can be referred in other relation.
Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

Q) Integrity Constraints over Relations:-
Database integrity refers to the validy and consistency of stored data. Integrity is usually expressed in terms of constraints, which are consistency rules that the database is not permitted to violate. Constraints may apply to each attribute or they may apply to relationships between tables.
Integrity constraints ensure that changes (update, deletion, insertion) made to the database by authorized users do not result in loss of data consistency. Thus, integrity constraints guard against accidental damage to the database.
Example:- A brood group must be 'A' or 'B' or 'AB' or 'O' only (can not any other values else).

Types of Integrity constraints

PAGE No.
DATE / /

Various types of integrity constraints are-
1) Domain Integrity
2) Entity Integrity Constraint
3) Referential Integrity Constraint
4) Key Constraints

1) Domain integrity - Domain integrity means the definition of a valid set of values. for an attribute you define data type, length or size is null value allowed, is the value unique or not for an attribute, the default value, the range and/or specific values for the attribute.

2) Entity Integrity Constraint - This rule states that in any database relation value of attribute of a primary key can't be null.
Example - Consider a relation "STUDENT" where "Stu_id" is a primary key and it must not contain any null value whereas other attributes may contain null value eg:- "Branch" in the following relation contains one null value.

Stu_id	Name	Branch
11255234	Aman	CSE
11255369	Kapil	ESE
11255324	Ajay	
11255237	Raman	CSE
11255678	Aastha	ECE

3) Referential Integrity Constraint - It states that if a foreign key exists in a relation then either the foreign key value must match a primary key value of some tuple in its home relation or the foreign key value must be null.

The rules are:

1) You can't delete a record from a primary table if matching records exist in a related table.

2) You can't change a primary key value in the primary table if that record has related records.

3) You can't enter a value in the foreign key field of the related table that doesn't exist in the primary key of the primary table.

4) However, you can enter a NULL value in the foreign key, specifying that the records are unrelated.

Example :- Consider 2 relations "stu" and "stu_1" where "stu_id" is the primary key in the "stu" relation and foreign key in the "stu_1" relation.

Relation "stu" :-

Stu_id	Name	Branch
1125S234	Arman	CSE
1125S369	Kapil	ECE
1125S324	Ajay	ME
1125S237	Raman	CSE
1125S678	Aastha	ECE

Relation "Stu_1"

Stu_id	Cource	Duration
11255234	B.TECH	4 years
11255369	B.TECH	4 years
11255324	B.TECH	4 years
11255237	B-TECH	4 years
11255678	B-TECH	4 years

Examples

Rule 1 :- You can't delete any of the rows in the "Stu" relation that are visible since all the "Stu" are in use in the "Stu_1" relation

Rule 2 :- You can't change any of the "Stu_id" in the "Stu" relation since all the "Stu_id" are in use in the "stu_1" relation.

Rule 3 :- The values that you can enter in the "Stu_id" field in the "stu_1" relation must be in the "Stu_id" field in the "Stu" relation.

Rule 4 :- You can enter a null value in the "stu_1" relation if the records are ~~are~~ unrelated.

4) Key Constraints - A key constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple.

There are 4 types of key constraints -

1) Candidate Key

2) Super Key
3) Primary Key
4) Foreign Key

Q) Enforcing Integrity Constraints (ICs)
ICs are specified when a relation is created and enforced when a relation is modified.
If an insert, delete or update command causes a violation, it is rejected.
Violation is generally checked at the end of each SQL statement execution.
The following insertion violates the primary key constraint because there is already a tuple with the sid 53688 and it will be rejected by the DBMS:
INSERT INTO Students (sid, name, login, age, gpa) VALUES (53688, 'MIKE', 'mike@ee', 17, 3.4)

The following insertion violates the constraint that the primary key cannot be null:
INSERT INTO Students (sid, name, login, age, gpa) VALUES (null, 'MIKE', 'mike@ee', 17, 3.4)

Deletion does not cause a violation of primary key or unique constraints. Updata can cause violations, similar to an insertion:

PAGE No.
DATE / /

UPDATE Student SET sid = 50000 WHERE sid = 53688.
This update violates the primary key constraint because there is already a tuple with sid 50000.
Foreign key constraints is more complex because SQL sometimes tries to rectify a foreign key constraint violation instead of simply rejecting the change.

5] Querying Relational Data

SQL is the most popular commercial query language for a relational DBMS. We can retrieve rows corresponding to students who are younger than 18 with the following SQL query:-

```
SELECT * FROM Students S WHERE S.age < 18
```

The system '*' means that we retain all fields of selected tuples in the result.
The condition S.age < 18 in the WHERE clause specifies that we want * to select only tuples in which the age field has a value less than 18.
In addition to selecting a subset of tuples, a query can extract a subset of the fields of each selected tuple.
We can compute the names and logins of students who are younger than 18 with the following query:

SELECT S.name, S.login FROM Students S WHERE S.age < 18.

It is obtained by applying the selection to the instance S1 of followed by removing unwanted fields.

(J) Logical Database Design: ER to Relational

1) Entity SETS to tables

Given the following ER diagram:

The Employees Entity set.

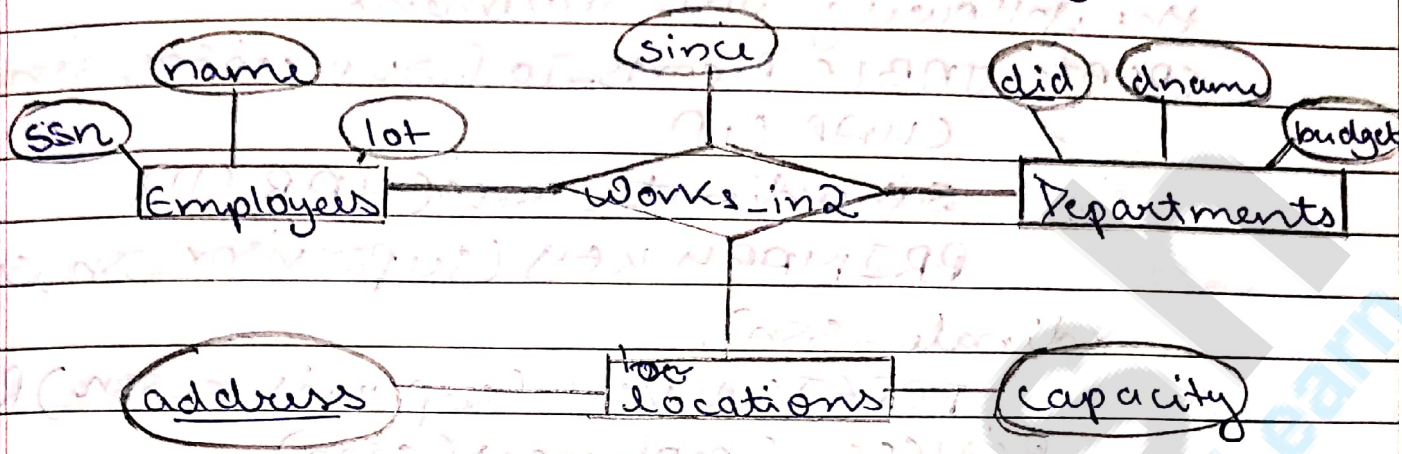
We can translate it directly into an RMD model by the following SQL Statement:

```
CREATE TABLE Employees(
    SSn CHAR(11),
    name CHAR(30),
    lot INTEGER,
    PRIMERY KEY (ssn)
```

2) Relationship Sets (without Constraints) to Tables

Given the ER diagram shown below, we can specify the two entity sets, while the relationship set is transla-

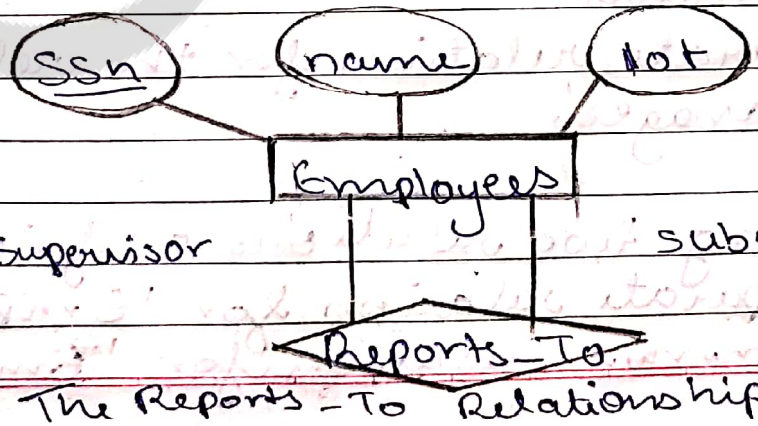
ted into an RMD relation as follows



A Ternary Relationship Set

```
Create Table Works_In2(ssn CHAR(11),
                       did INTEGER,
                       address CHAR(20),
                       since DATE,
                       PRIMARY KEY(ssn, did,
                         address),
                       FOREIGN KEY(ssn) REFER-
                       ENES Employees
                       FOREIGN KEY(address)
                       REFERENCES LOCATIONS
                       FOREIGN KEY(did) REFER-
                       ENCES Departments)
```

3) The following ER diagram:-

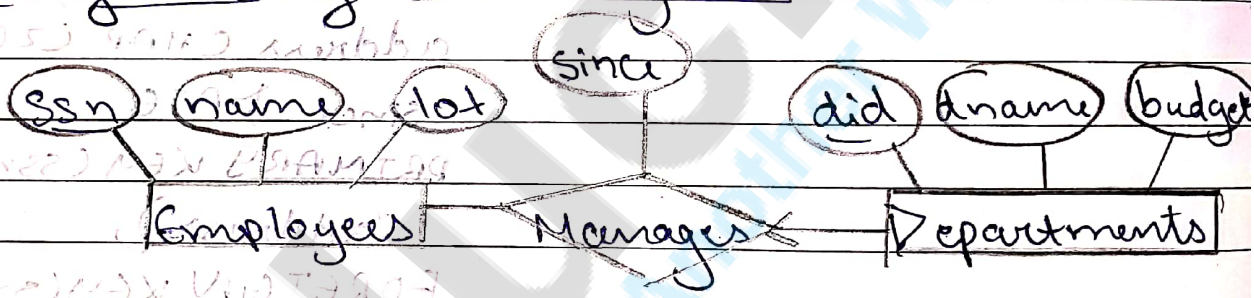


The Reports_To Relationship Set

is translated into an RMD relation by the following SQL statement:

```
CREATE TABLE Reports_To (supervisor_ssn
        CHAR (11),
        subordinate_ssn (CHAR (11),
        PRIMARY KEY (supervisor_ssn, subo-
rdinate_ssn),
        FOREIGN KEY (supervisor_ssn) REFE-
RENCES Employees(ssn),
        FOREIGN KEY (subordinate_ssn)
REFERENCES Employees (ssn))
```

The following ER diagram:



Key Constraint on Manages can be translated into relations using either of two approaches:-

1) Defining three relations as follows:-
   - A seperate relation for the entity set 'Employees'.
   - A seperate relation for the entity set 'Departments'.
   - A seperate relation for the relationship set 'Manages'.

2) Defining two relations as follows:-
   - A separate relation for 'Employees'.
   - A common relation for 'Employees' and

for 'Manages'.

-Approach (1) is obtained with the following SQL statement for the relationship set 'Manages' (in addition to two SQL statements for the two entity sets):-

```
CREATE TABLE Manages (ssn CHAR(11),
        did INTEGER,
        since DATE,
        PRIMARY KEY (did)
        FOREIGN KEY (ssn) REFERENCES
        Employees,
        FOREIGN Key (did) REFERENCES
        Departments)
```

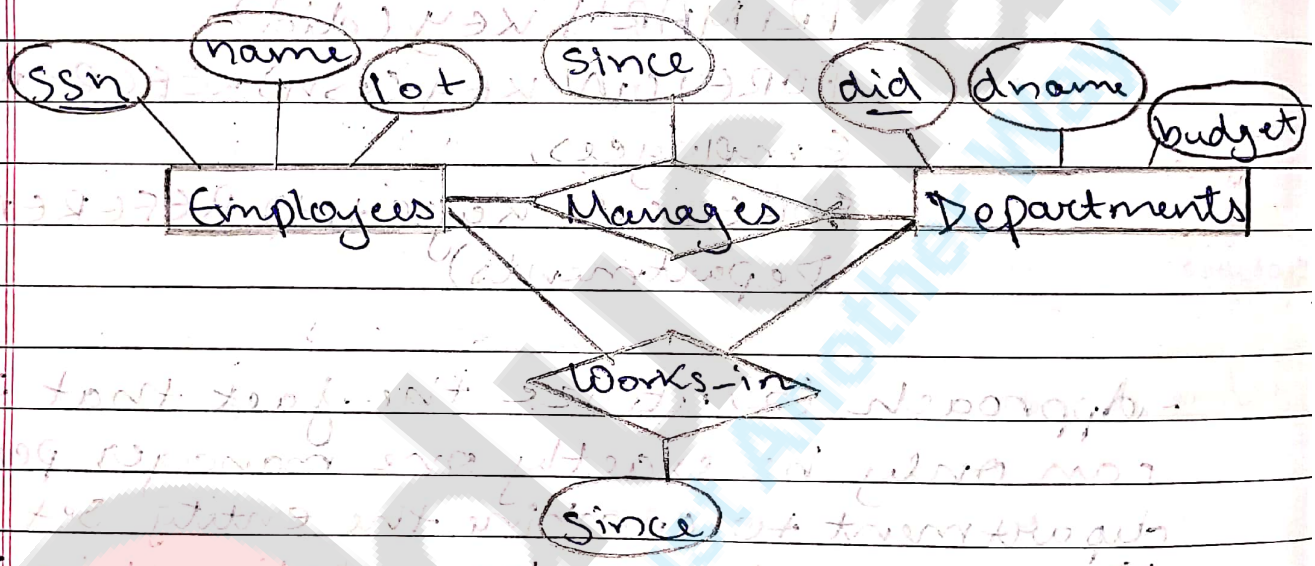
-Approach (2) utilize the fact that there can only be exactly one manager per department to combine the entity set 'Department' with the relationship 'Manages' into a single relation named 'Dept-Mgr' using the following SQL statement:

```
CREATE TABLE Dept_Mgr(did INTEGER,
        dname CHAR(20),
        budget REAL,
        ssn CHAR(11),
        since DATE,
        PRIMARY KEY (did),
        FOREIGN KEY (ssn) REF-
        ERENCES Employees)
```

Remarks:-

Note that ssn can take on null values indicating thereby that the concerned department has no manager which is permissible since there is no total participation constraint on the 'Departments' entity set.

3) Translating Relationship Sets with Participation Constraints



Manages and Works_In

Here we translate the 'Manages' relationship set into an RMD relation by the following SQL statements:

```
CREATE TABLE Dept_Mgr (did INTEGER,
                       dname CHAR(20),
                       budget REAL,
                       ssn CHAR(11) NOT NULL,
                       since DATE,
                       PRIMARY KEY (did));
```

FOREIGN KEY (SSN) REFERENCES Employees ON DELETE NO ACTION)

Remarks

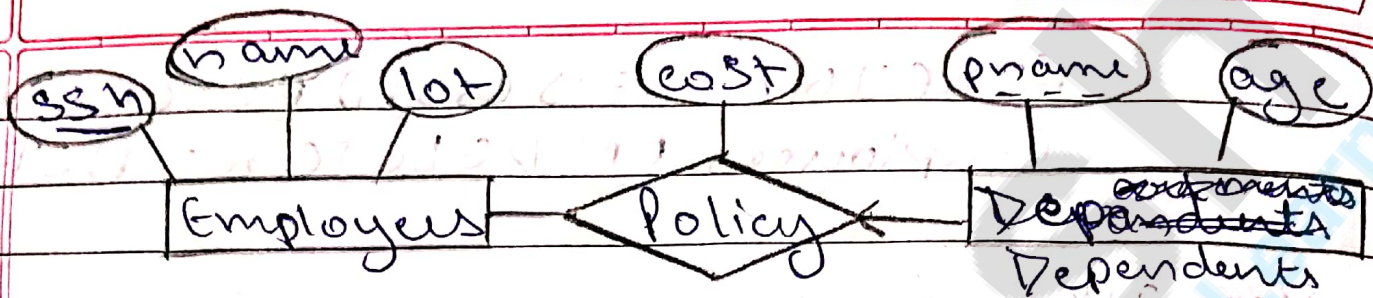
1) We are using here the second approach mentioned in the example because the constraint that every department must have a manager cannot be captured with the first approach.

2) Since the ER diagram shows a total participation constraint on the Department entity set as well as a key constraint, there must be exactly one manager per department. Therefore the attribute Ssn cannot take on null values.

3) The NO ACTION specification (which is not really needed since it is the default) ensures that an Employee tuple cannot be deleted while it is pointed to by a 'Dept_Mgr' tuple.

4) The total participation constraint shown in 'Works_In' relationship set cannot be specified in the above SQL statement unless we use table constraints or assertions.

Translating Weak Entity sets into Relations



The Dependents Weak Entity Set can be translated into a relation by the following SQL statement which uses the second approach described previously (combining 'Policy' and 'Dependents')

```
CREATE TABLE Dep_Policy (pname CHAR(20),
                         age INTEGER,
                         cost REAL,
                         SSN CHAR(11),
                         PRIMARY KEY (pname, SSN),
                         FOREIGN KEY (SSN) REFERENCES
                         Employees ON DELETE CASCADE)
```