ADITYA ENGINEERING COLLEGE (A)

DBMS
Entity Relationship Model

By

Dr. S Rama Sree
Professor in CSE & Dean(Academics)
Aditya Engineering College(A)
Surampalem.

CONTENTS

- Introduction
- Representation of entities, attributes, entity set
- Relationship, relationship set
- Constraints
- Sub classes, super class
- Inheritance, specialization, generalization using ER Diagrams.

Introduction

- The Entity-Relationship(ER) data model allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships and is widely used to develop an initial database design
- The ER model is used in the phase called conceptual database design
- The database design process consists of 6 steps
- Step 1: Requirements Analysis

  Understand what data is to be stored in the database, what applications must be built on top of it, and what operations are most frequent and subject to performance requirements. In other words, we must find out what the users want from the database.

- Step 2: Conceptual Database Design:

  The information gathered in the requirements analysis step is used to develop a high-level description of the data to be stored in the database, along with the constraints known to hold over this data. This step is often carried out using the ER model

- Step 3: Logical Database Design:

  We must choose a DBMS to implement our database design, and convert the conceptual database design into a database schema in the data model of the chosen DBMS. the task in the logical design step is to convert an ER schema into a relational database schema.

- Step 4:Schema Refinement

  Identify the potential problems and refine the database schema. Ex: normalizing the relations

- Step 5:Physical Database Design

  This step may simply involve building indexes on some tables and clustering some tables, or it may involve a substantial redesign of parts of the database schema obtained from the earlier design steps to meet the desired performance criteria.

- Step 6:Application and Security Design

  we must identify the parts of the database that must be accessible and the parts of the database that must not be accessible, and we must take steps to ensure that these access rules are enforced.

Entities, Attributes and Entity Sets

- An **entity** is an object in the real world that is distinguishable from other objects. Ex: EMP
- A collection of similar entities is called an **entity set.**

  Ex:Toys dept. Emp & Appliance dept Emp.
- An entity is described using a set of **attributes.** All entities in a given entity set have the same attributes; this is what we mean by similar.

  Ex: ENAME, EID etc.
- For each attribute associated with an entity set, we must identify a domain of possible values.

  Ex: ename takes a set of characters, eid is an integer
- In ER diagram, An entity set is represented by a **rectangle**, and an attribute is represented by an **oval**. Each attribute in the **primary key is underlined.**

Attribute Types

- Simple and composite attributes.
  - Simple Attribute:-They cannot be divided into sub parts.
  - Composite Attribute:-They can be divided into sub parts. These may appear as a hierarchy.

    Ex: the attribute "name" can be divided into first name, middle name and last name.
- Single-valued and multi-valued attributes
  - Single-valued attribute:-Those attributes which can take only one value for a entity.
  - Multi-valued attribute:-An attribute has a set of values for a specific entity.

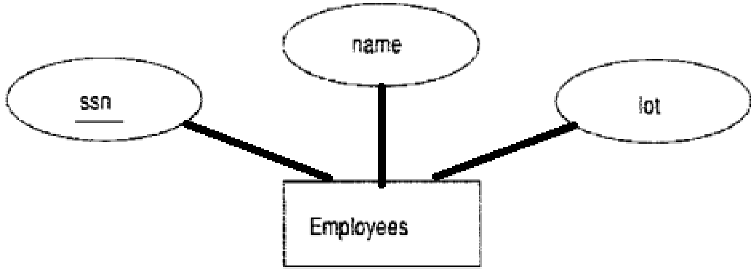
    Ex: phone number attribute
- Derived attributes
  - Derived attribute:- Value of this attribute can be derived from the value of other related attributes.
  - Ex: If date_of_birth is an attribute for employee, then age attribute can be derived from the date_of_birth attribute

RELATIONSHIPS AND RELATIONSHIP SETS

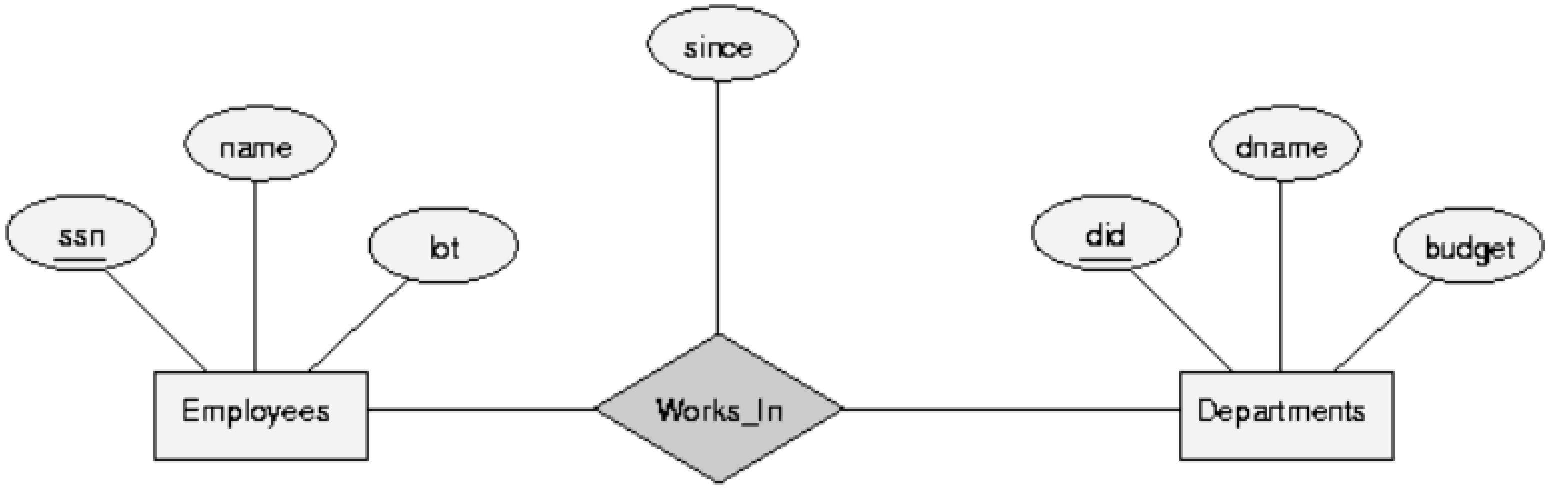
- A **relationship** is an association among two or more entities.
- A collection of a set of similar relationships is called a relationship set. A relationship set can be thought of as a set of n-tuples:

  $\{(e_1,\ldots.e_n) \mid e_1 \in E_1,\ldots.., e_n \in E_n\}$

  The Employees Entity Set

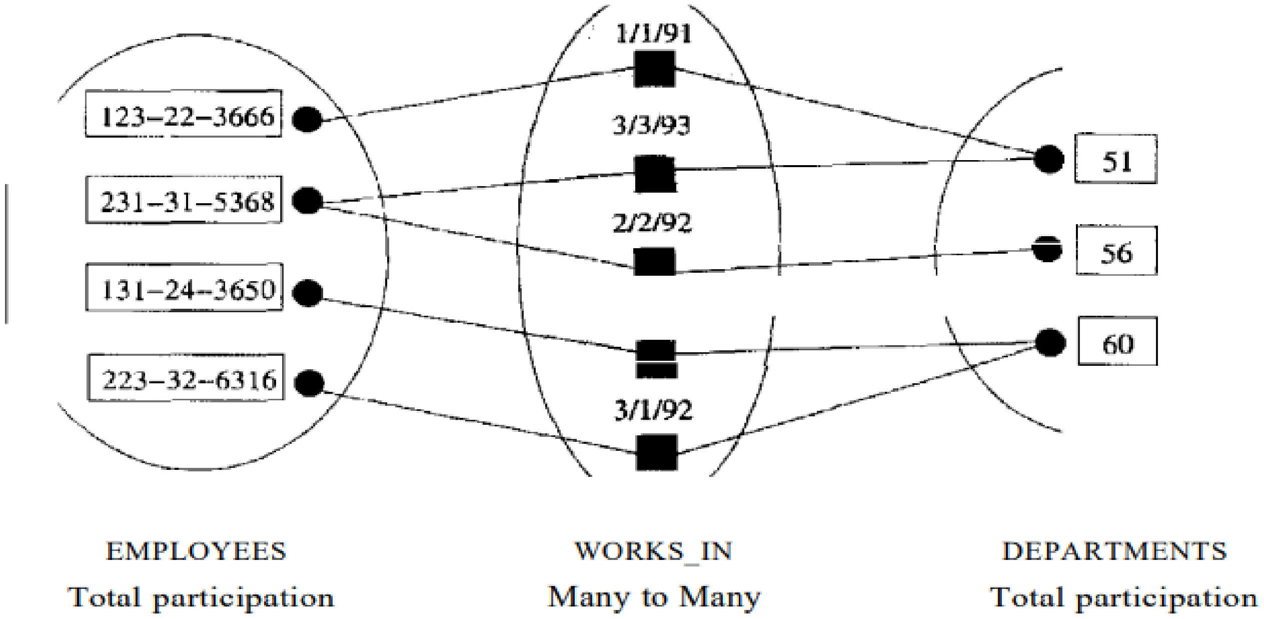


- Each n-tuple denotes a relationship involving n entities e_1 through e_n, where entity e_i is in entity set E_i .
- In the Figure, we show the relationship set Works_In, in which each relationship indicates a department in which an employee works.



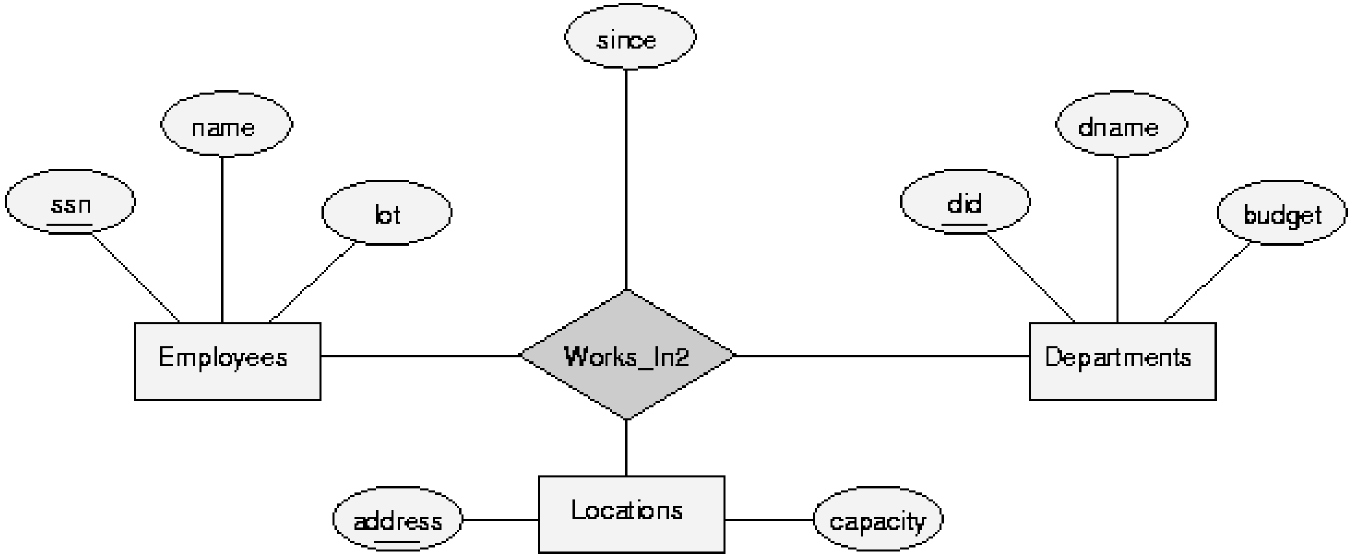
- A relationship can also have **descriptive attributes**. Descriptive attributes are used to record information about the relationship, rather than about any one of the participating entities. Ex: since attribute for Works_In relationship
- A relationship must be uniquely identified by the participating entities, without reference to the descriptive attributes. In the Works_In relationship set, for example, each Works_In relationship must be uniquely identified by the combination of employee ssn and department did. Thus, for a given employee-department pair, we cannot have more than one associated since value

- An instance of a relationship set is a set of relationships.
- An instance can be thought of as a 'snapshot' of the relationship set at some instant in time. An instance of the Works_In relationship set is shown in Figure. The attribute **ssn** for Employees and **did** for Departments is shown.

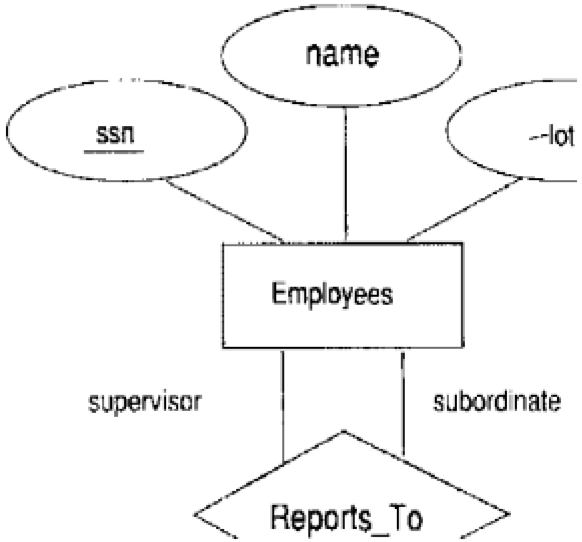


An instance of Works_In Relationship

- Suppose that each department has offices in several locations and we want to record the locations at which each employee works. This relationship is **ternary relation** because we must record an association between 3 entities employee, a department, and a location. The ER diagram for this is Works_In2



- Sometimes a relationship might involve two entities in the same entity set. For example, consider the Reports_T relationship set shown.
- Employees report to other employees, every relationshi in Reports_To is of the form (emp1, emp2) , where bot emp1 and emp2 are entities in Employees. However, the play different roles: empl reports to the managir employee emp2, which is reflected in the role indicato supervisor and subordinate.
- If an entity set plays more than one role, the role indicat is concatenated with the attribute name from the entity set.
- Ex: The Reports_To relationship set has attributes corresponding to the ssn of the supervisor and the ssn of the subordinate, and the names of these attributes are supervisor ssn and subordinate ssn.



ER to Relational model

Entity sets to Tables

Entity set is mapped to a relation. Each attribute of the entity set becomes an attribute of table.

```
Create table employees
(
ssn integer primary key,
name varchar2(20),
lot integer
);
```

```
Create table departments
(
did integer primary key,
dname varchar2(20),
budget integer
);
```

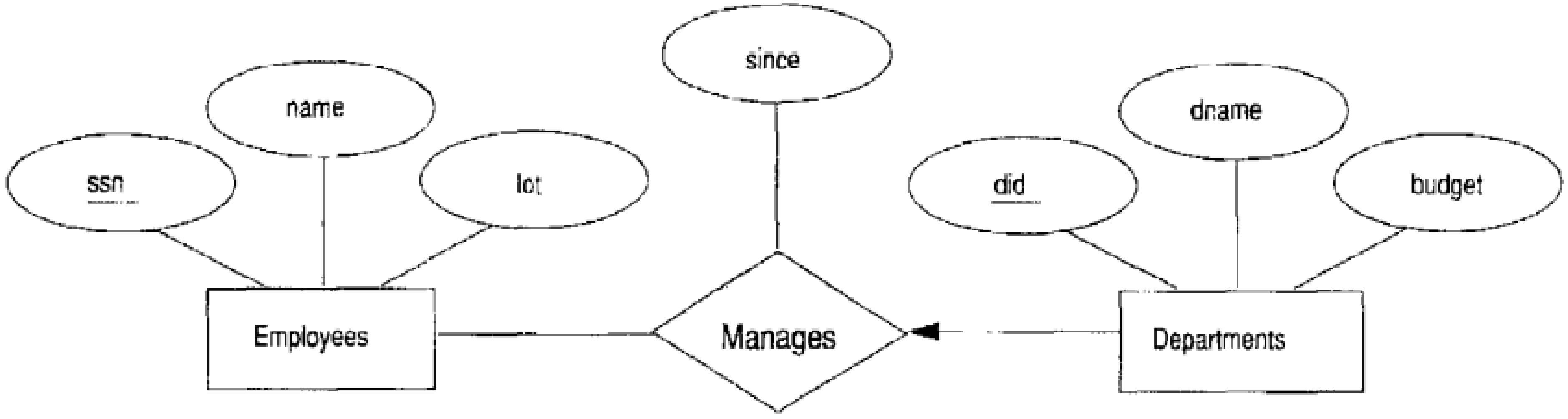
```
Create table locations
(
Address varchar2(20) primary key,
capacity integer
);
```

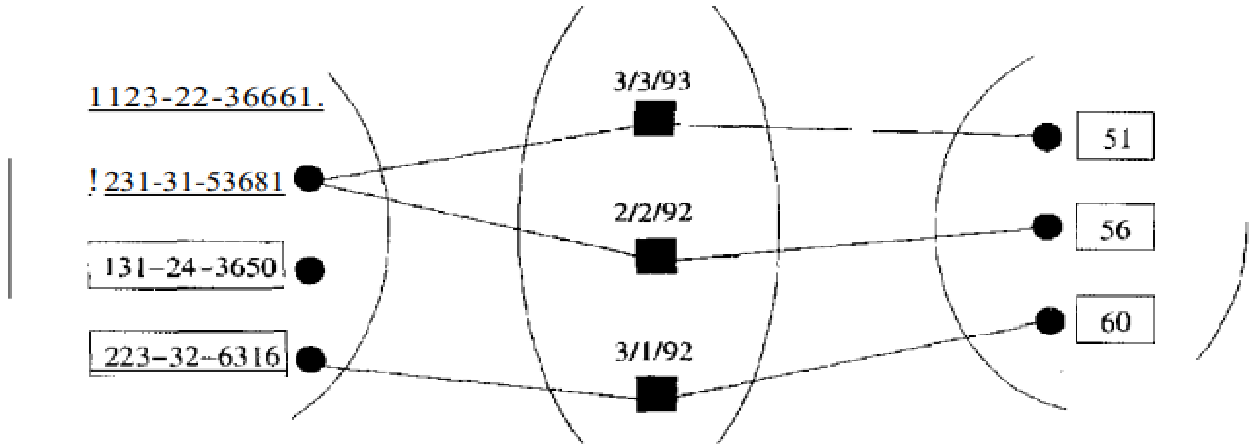
ER to Relational model

```
Create table works_in
(
ssn integer,
did integer,
address varchar2(20),
Since date,
Primary key(ssn,did,address),
Foreign key(ssn) references employees(ssn),
Foreign key(did) references departments(did),
Foreign key(address) references locations(address));
```

Key Constraints

- Consider the Works_In relationship shown earlier. An employee can work in several departments. Ex: Employee 231-31-5368 worked in Department 51 since 3/3/93 and in Department 56 since 2/2/92.
- Now consider another relationship set called Manages between the Employees and Departments entity sets such that each department has at most one manager, although a single employee is allowed to manage more than one department. The restriction that each department has at most one manager is an example of a **key constraint**.
- It implies that each Departments entity appears in at most one 1Jlanages relationship
- This restriction is indicated in the ER diagram by using an arrow from Departments to Manages





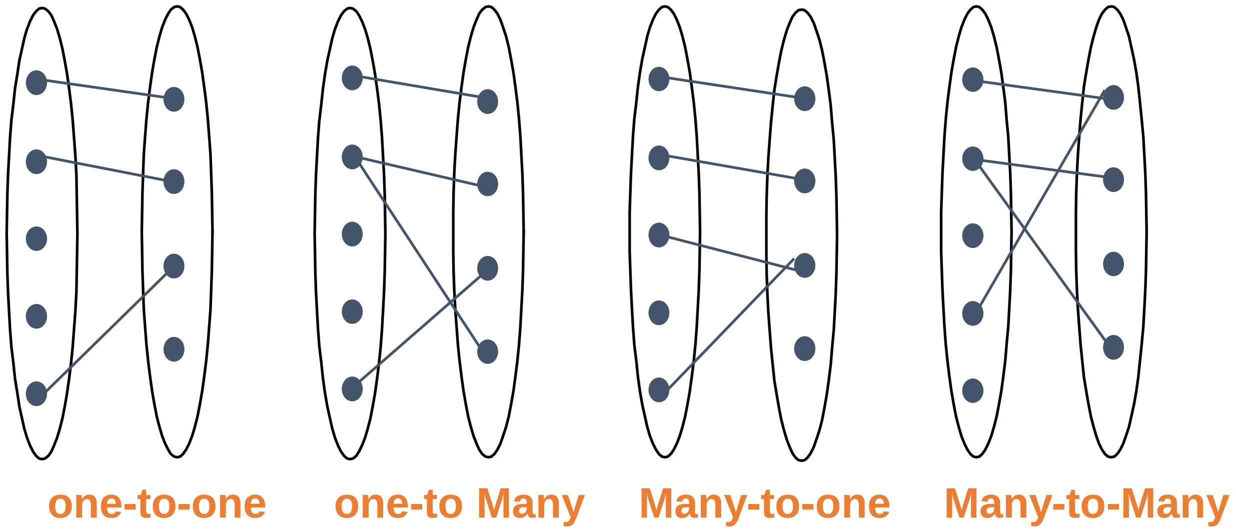
EMPLOYEES
Partial participation

MANAGES
One to Many

DEPARTMENTS
Total participation

An instance of Manages relationship

- A relationship set like Manages is sometimes said to be **one-to-many**, to indicate that one employee can be associated with many departments.
- The Works_In relationship set, in which an employee is allowed to work in several departments and a department is allowed to have several employees, is said to be **many-to-many.**
- The mapping cardinalities are **one-to-one, one-to-many, many-to-one &many-to-many.**



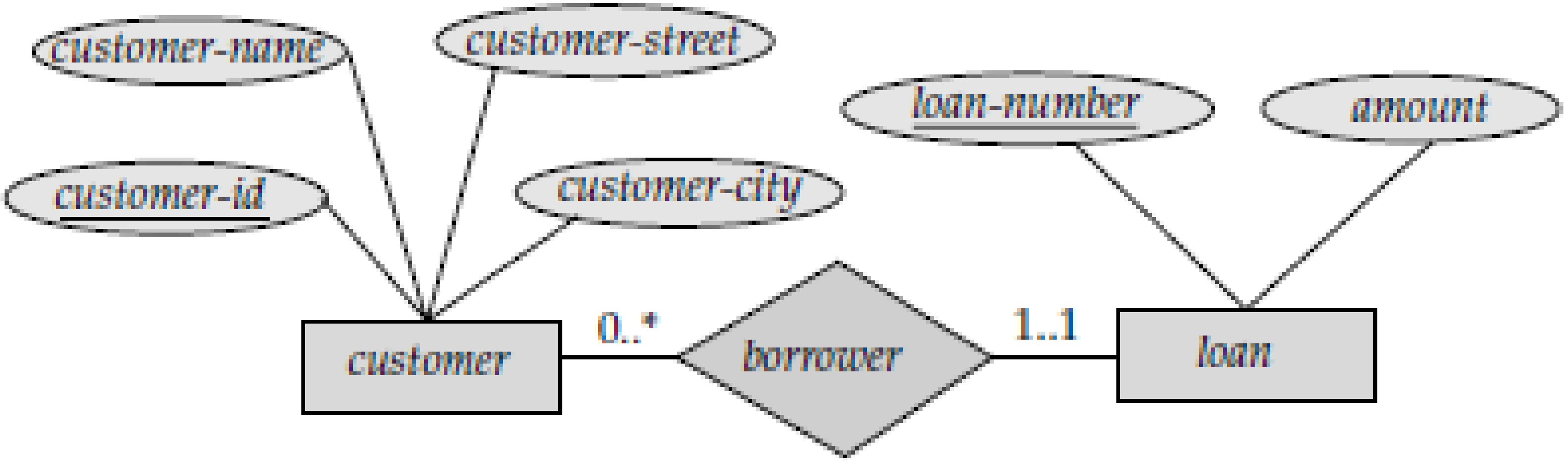
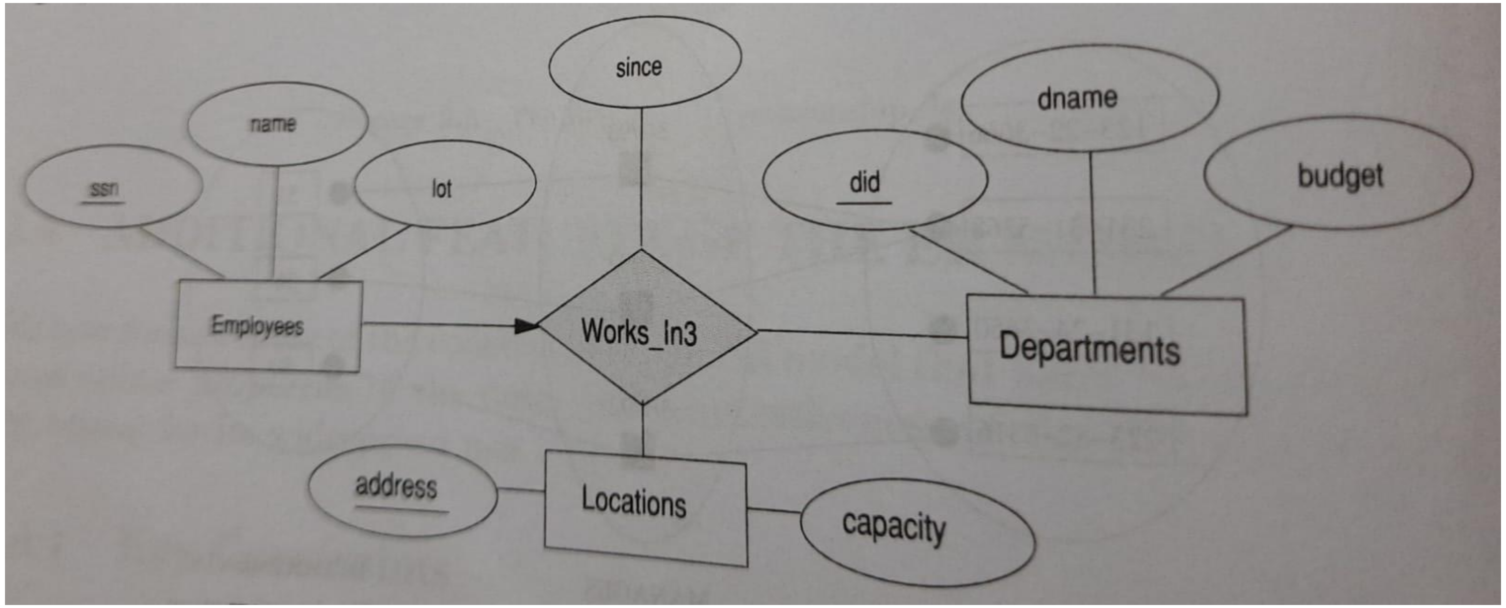


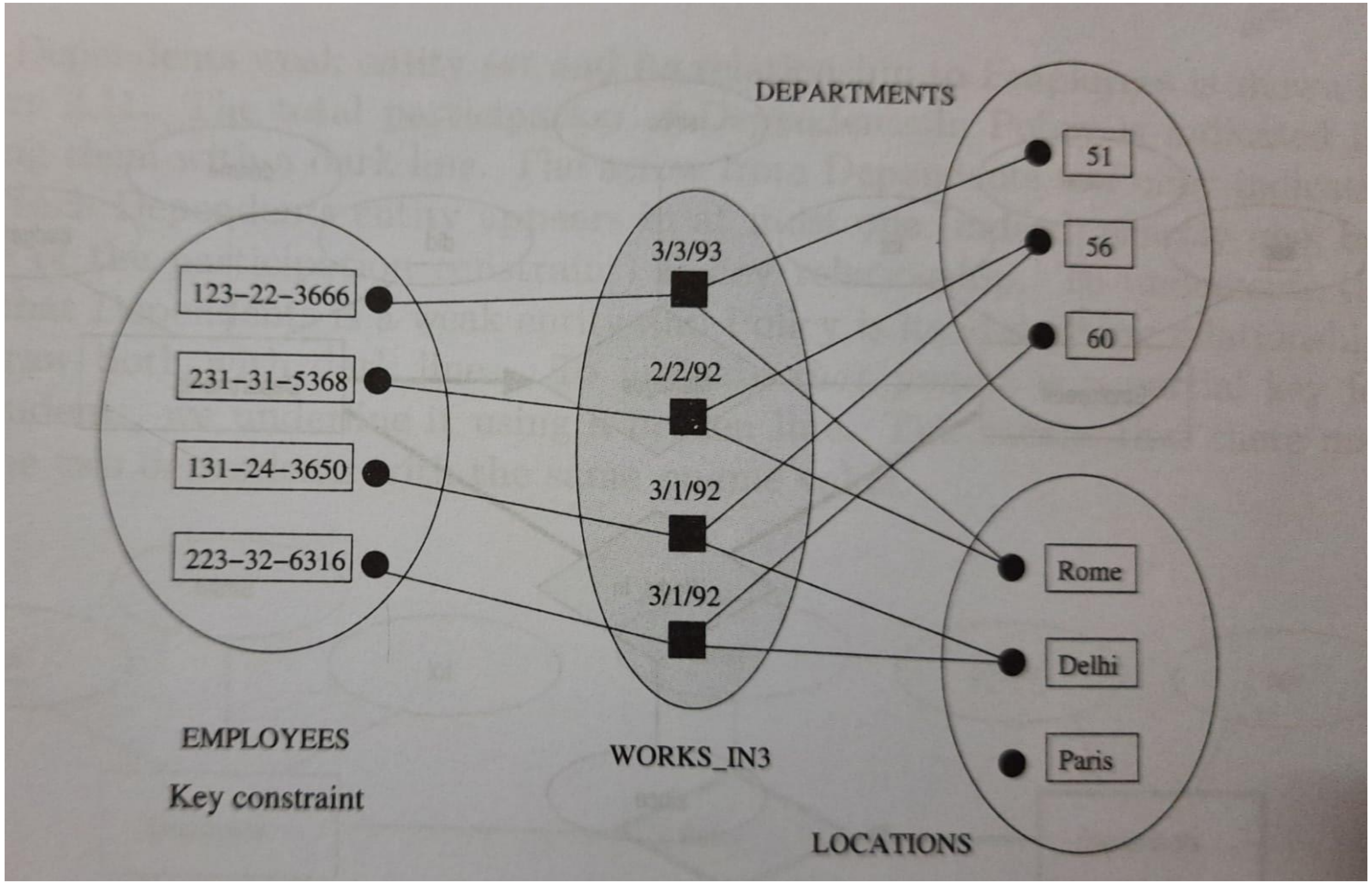
Figure 2.15 Cardinality limits on relationship sets.

- The cardinality of a relationship is the number of instances of entity that can be associated with another entity.
- 0..* from customer to borrower indicates that a customer can have zero or more loans.
- 1..1 every loan is related to atleast one customer.

Key Constraints for Ternary Relationships

- We show a ternary relationship with key constraints. Each employee works in at most one department and at a single location.
- Note that each department can be associated with several employees and locations and each location can be associated with several departments and employees; however, each employee is associated with a single department and location

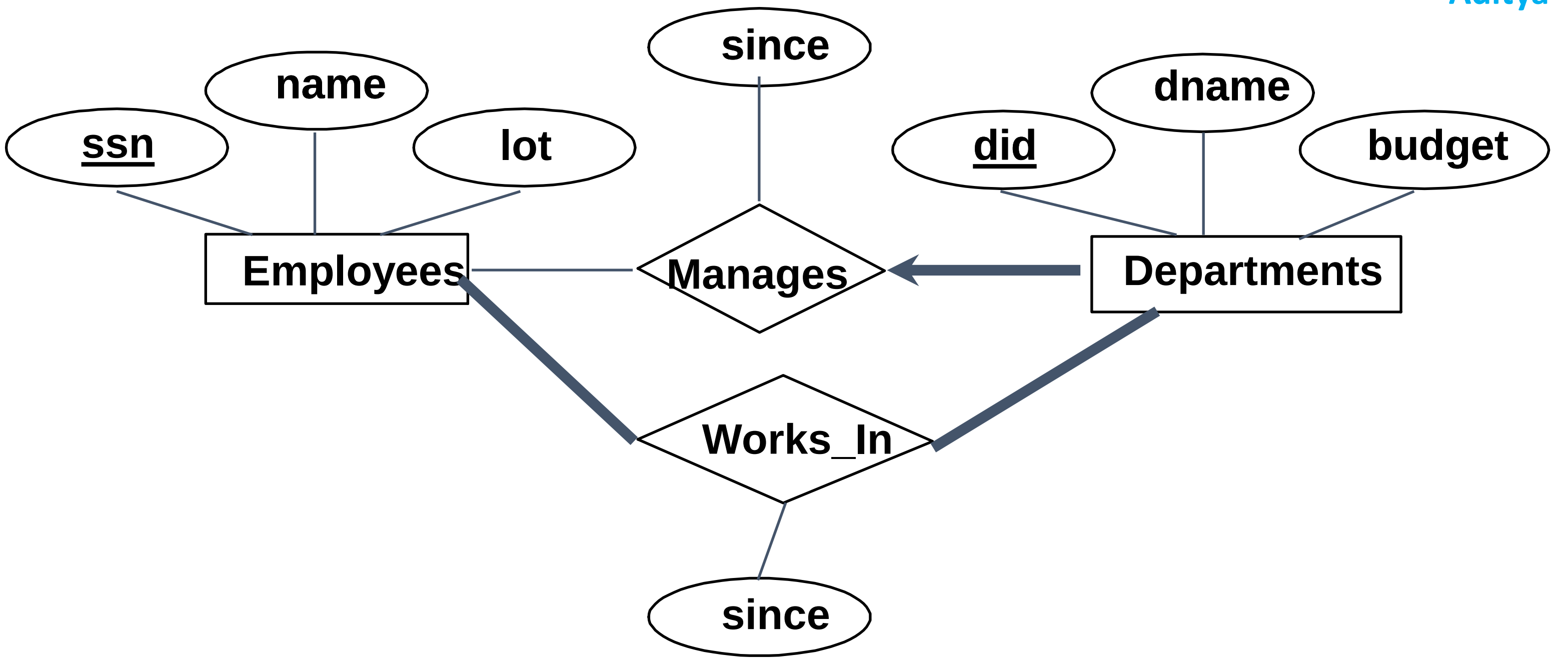




An instance of Works_In3

Participation Constraints

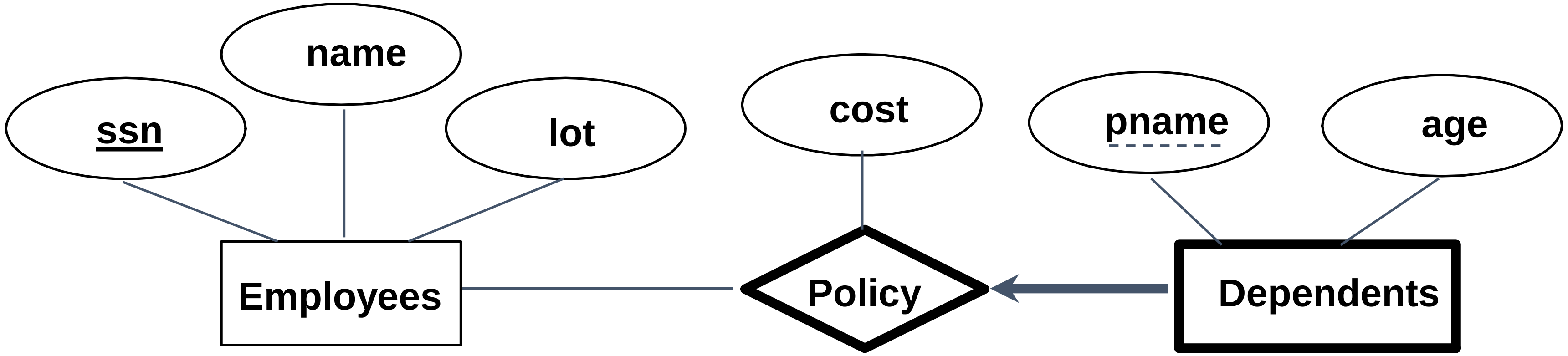
- Total Participation:-If every entity in E participates in atleast one relationship set in R. It is represented using thick line between the entity set and relationship set.
- Partial Participation:- If only some entities in E participates in relationship R. It is represented using single line between the entity set and relationship set.
- Ex: If the requirement is Every department is required to have a manager. The participation of the entity set Departments in the relationship set Manages is said to be total. A participation that is not total is said to be partial.
- Ex: The participation of the entity set Employees in Manages is partial, since not every employee gets to manage a department



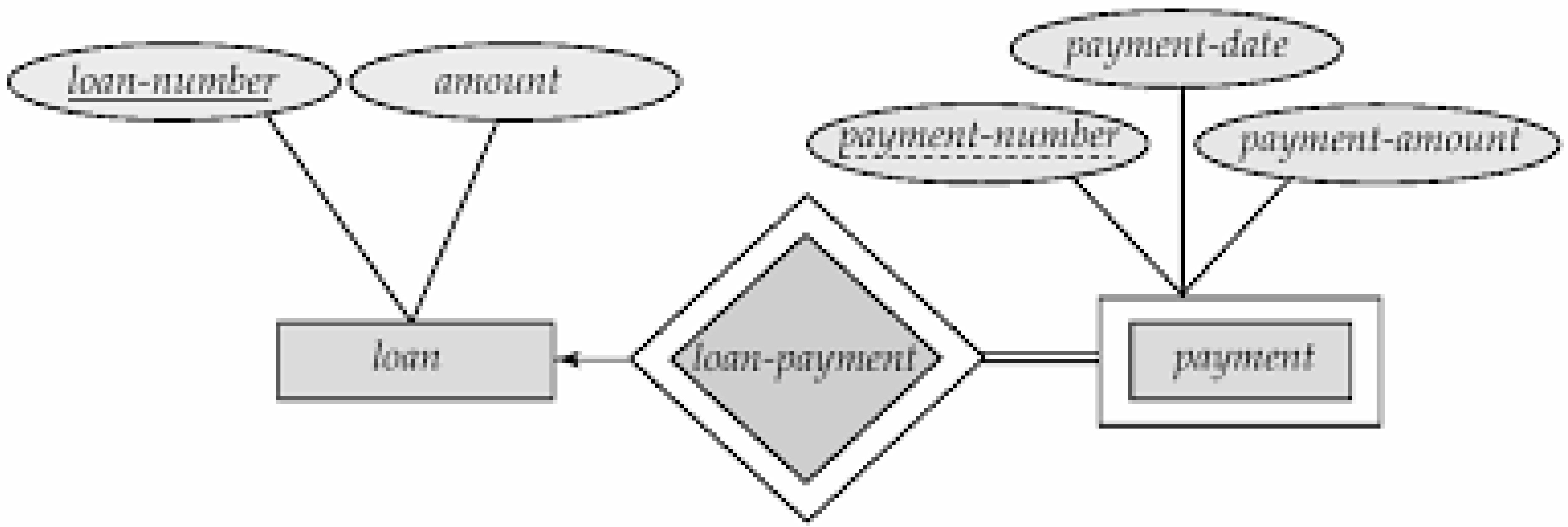
- It is natural to expect that each employee works in at least one department and that each department has at least one employee. This means that the participation of both Employees and Departments in Works._In is total.
- If the participation of an entity set in a relationship set is total, the two are connected by a thick line or double line.
- The presence of an arrow indicates a key constraint.

Weak Constraints

- Let Dependents be an entity set with attributes pname and age. This entity is used to hold the policy details for the dependents of the employees.
- A weak entity can be identified uniquely only by considering some of its attributes in conjunction with the primary key of another entity, which is called the identifying owner
- The following restrictions must hold:
- *The owner entity set and the weak entity set must participate in a one-to-many relationship set (one owner entity is associated with one or more weak entities, but each weak entity has a single owner). This relationship set is called the identifying relationship set of the weak entity set.*
- *The weak entity set must have total participation in the identifying relationship set.*
- Ex: The Dependents entity can be identified uniquely only if we take the key of the owning Employees entity and the pname of the Dependents entity. The set of attributes of a weak entity set that uniquely identify a weak entity for a given owner entity is called a partial key of the weak entity set. In our example, pname is a partial key for Dependents



- The total participation of Dependents in Policy is indicated by linking them with a dark line or double line. The arrow from Dependents to Policy indicates that each Dependents entity appears in at most one Policy relationship.
- To indicate that pname is a partial key for Dependents, we underline it using a broken line. This means that there may well be two dependents with the same pname value



- An entity set that does not have a primary key is referred to as a **weak entity set.**
- The **discriminator** (*or partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The primary key of a weak entity set is formed by the primary key of the strong entity set plus the weak entity set's discriminator
- Identifying relationship is depicted using a double diamond or thick diamond shape.

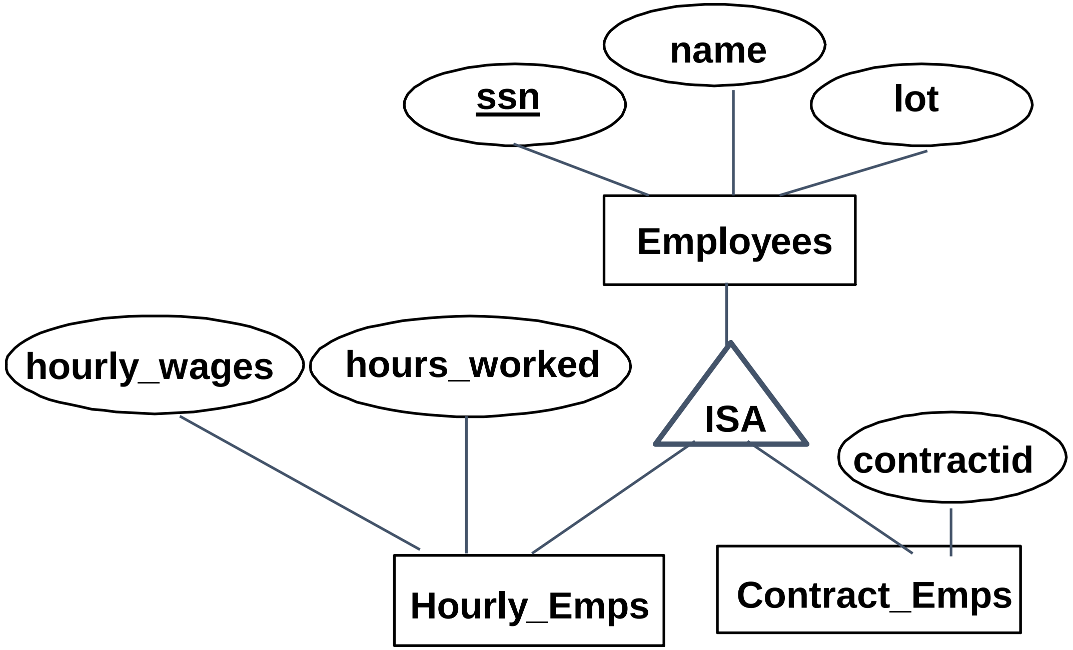
Class Hierarchies

- In some cases, the attributes of one entity set need to be inherited by other entity sets. In all such cases, the entity hierarchies are used.

- The attributes defined for an Hourly_Emps entity are the attributes for Employees plus Hourly_Emps

- We say that the attributes for the entity set Employees are inherited by the entity set Hourly_Emps and that Hourly-Emps ISA (read as “ is a”) Employees entity.

- We might also identify a subset of employees as Senior_Emps.



- A class hierarchy can be viewed in one of two ways:

  1. Specialization 2. Generalization

Specialization:

- Employees is specialized into subclasses.
- Specialization is the process of identifying subsets of an entity set (the superclass) that share some distinguishing characteristic. It is a top down process.
- Typically, the superclass is defined first, the subclasses are defined next, and subclass-specific attributes and relationship sets are then added.

Generalization

- Hourly_Emps and Contract_Emps are generalized by Employees.
- Generalization consists of identifying some common characteristics of a collection of entity sets and creating a new entity set that contains entities possessing these common characteristics. It is a bottom up process
- Typically, the subclasses are defined first, the superclass is defined next, and any relationship sets that involve the superclass are then defined.

- There are 2 kinds of constraints with respect to ISA hierarchies –

  overlap and covering constraints.

- Overlap constraints determine whether two subclasses are allowed to contain the same entity.
- Ex: Can Attishoo be both an Hourly_Emps entity and a Contrac_Emps entity? Intuitively, no.
- Can he be both a Contrac_Emps entity and a Senior_Emps entity? Intuitively, yes. We denote this by writing 'Contrac_Emps OVERLAPS Senior_Emps.'
- In the absence of such a statement, we assume by default that entity sets are constrained to have no overlap.

- Covering constraints determine whether the entities in the subclasses collectively include all entities in the superclass.
- For example, does every Employees entity have to belong to one of its subclasses? Intuitively, no.
- Does every Motor_Vehicles entity have to be either a Motorboats entity or a Cars entity? Intuitively, yes;
- A characteristic property of generalization hierarchies is that every instance of a superclass is an instance of a subclass.
- We denote this by writing 'Motorboats AND Cars COVER Motor-Vehicles.' In the absence of such a statement, we assume by default that there is no covering constraint.

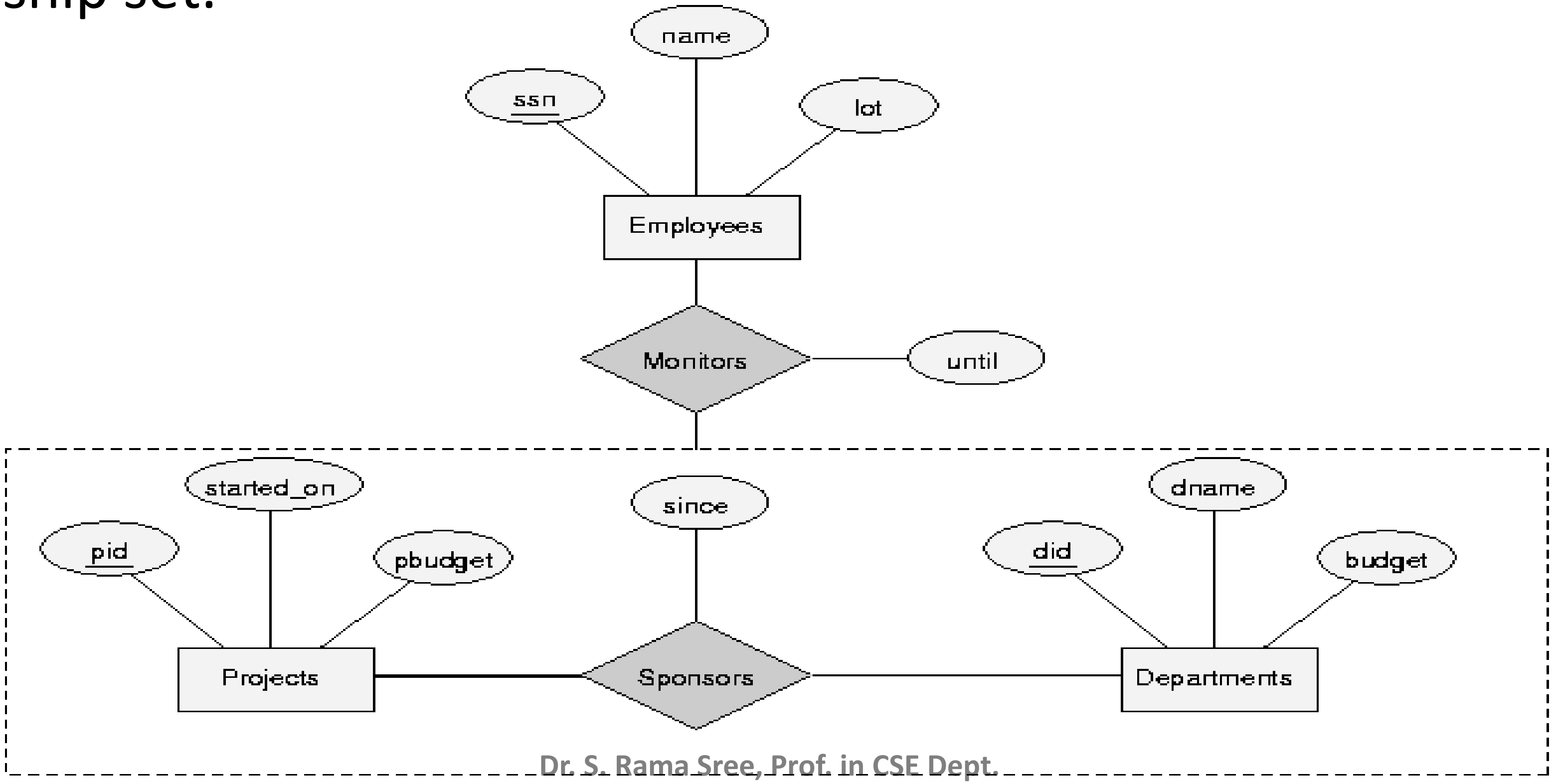
- There are two basic reasons for identifying subclasses (by specialization or generalization):

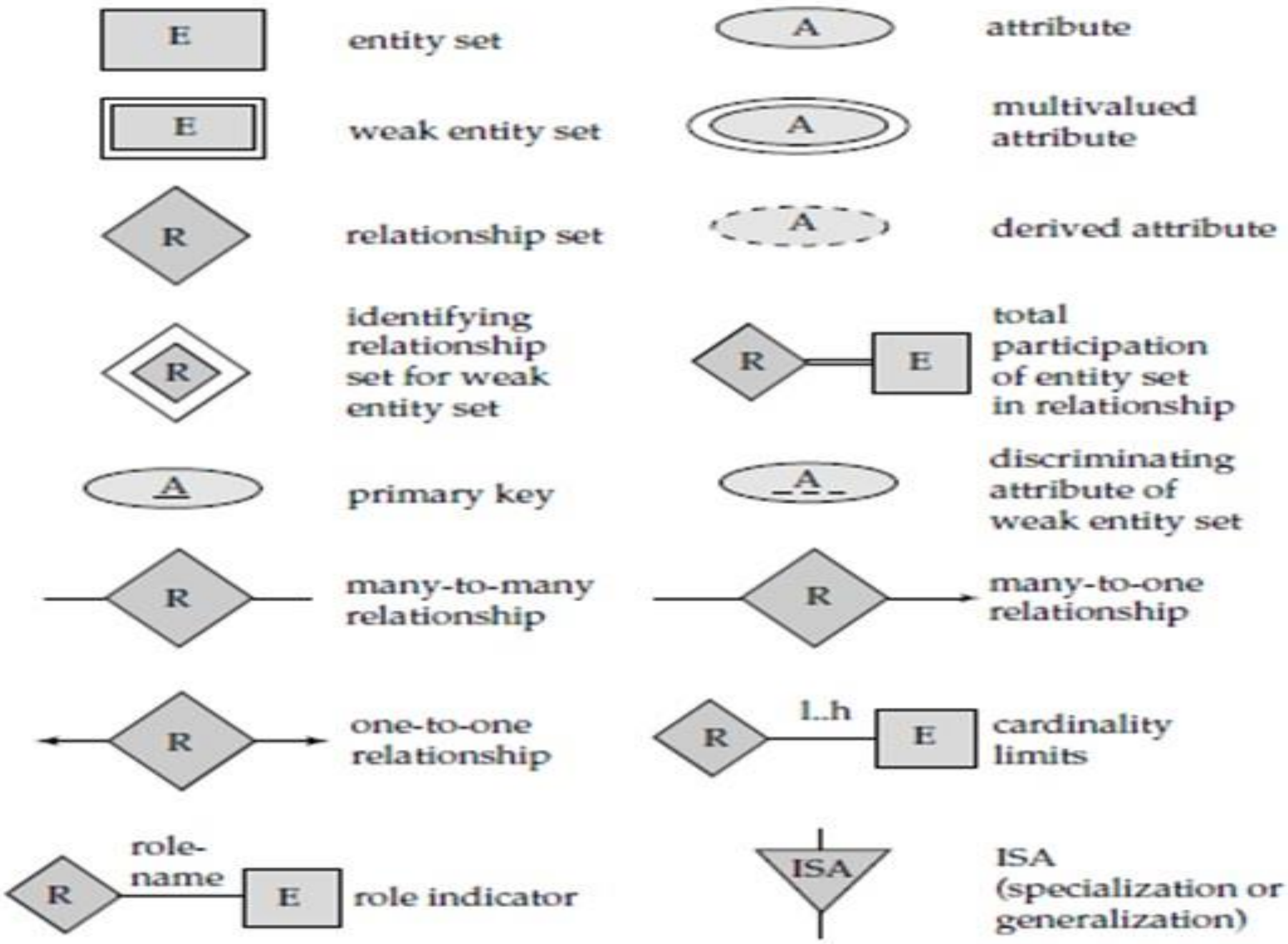
1. We might want to add descriptive attributes that make sense only for the entities in a subclass. For example, hourly_wages does not make sense for a Contract_Emps entity, whose pay is determined by an individual contract.
2. We might want to identify the set of entities that participate in some relationship.

Inheritance and Aggregation

Inheritance:-It allows lower-level entities to inherit the attributes of higher-level entities.

Aggregation:-It allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships. Aggregation allows us to indicate that a relationship set (identified through a dashed box) participates in another relationship set.




E
entity set
A
attribute
E
weak entity set
A
multivalued attribute
R
relationship set
A
derived attribute
R
identifying relationship set for weak entity set
R
E
total participation of entity set in relationship
A
primary key
A
discriminating attribute of weak entity set
R
many-to-many relationship
R
many-to-one relationship
R
one-to-one relationship
R
l..h
E
cardinality limits
R
role-name
E
role indicator
ISA
ISA (specialization or generalization)

References

Text Books:

- Database Management Systems, 3/e, Raghurama Krishnan, Johannes Gehrke, TMH.
- Database System Concepts,5/e, Silberschatz, Korth, TMH.

Reference Books:

- Introduction to Database Systems, 8/e C J Date, PEA.
- Database Management System, 6/e Ramez Elmasri, Shamkant B. Navathe, PEA
- Database Principles Fundamentals of Design Implementation and Management, Corlos Coronel, Steven Morris, Peter Robb, Cengage Learning.

Web Links:

- https://nptel.ac.in/courses/106/105/106105175/
- https://www.geeksforgeeks.org/introduction-to-nosql/
- https://beginnersbook.com/2015/05/normalization-in-dbms/

Thank You