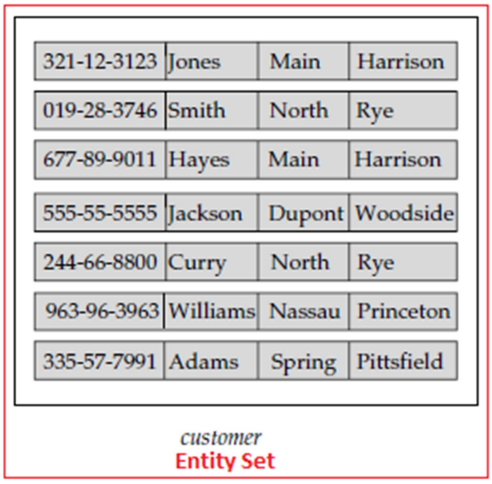
What is Entity, Relationship?

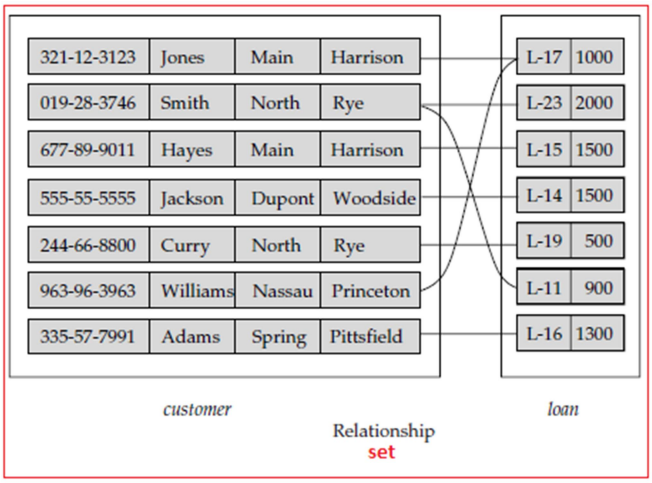
- **Entity:**
  "Thing" or "Object" in the real world that is distinguishable from other objects.
  Example: Savings Account, Student

- **Entity Set:**
  Set of all Entities of same type
  Example: List SB Account holders, List Customers, List Students

- **Attribute:**
  Description of Entity
  Example: SB Account (Entity) - A/c Holder Name, Age, Address
  Student (Entity) - Student Name, Age, Sex, Class, Section
- **Relationship:**
  Association among several Entities
  Example: SB Account Vs Customer

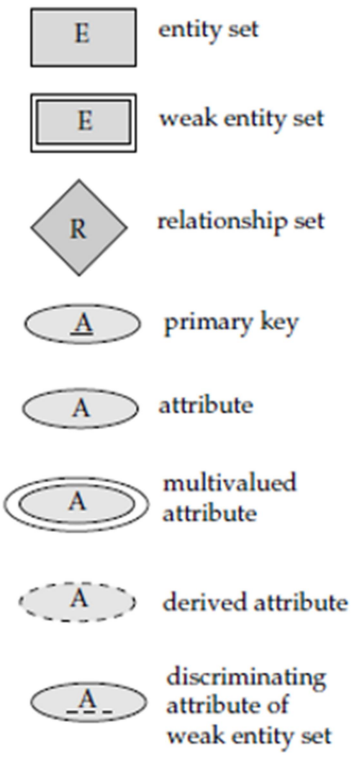
- **Relationship Set:**
  Set of all Relationships of same type
  Example: SB Account Vs **Customers**





Attributes

- ✓ **Simple Attribute:** Which Cannot be further derived
  Example: First Name, Last Name, Surname
- ✓ **Derived Attribute:** Values can be derived from other attributes
  Example: Age (i.e., from Date of Birth)
- ✓ **Composite Attribute:** Which can be further derived
  Example: Name -> First Name, Last Name
- ✓ **Single Valued Attribute:** Which takes only one value
  Example: Gender, Age
- ✓ **Multivalued Attribute:** Which stores more than one value
  Example: Phone Number, Name
- ✓ **Stored Attribute:** Which do not require any updation
  Example: Sex
- ✓ **Descriptive Attribute:** Which gives information about relationship
  Example: Relation
- ✓ **Complex Attribute:** Collection of both Multivalued & Composite Attributes
  Example: Address
- ✓ **Key Attribute:** Uniquely identifies an entity in the entity set
  Example: Employee Id, Student Roll Number

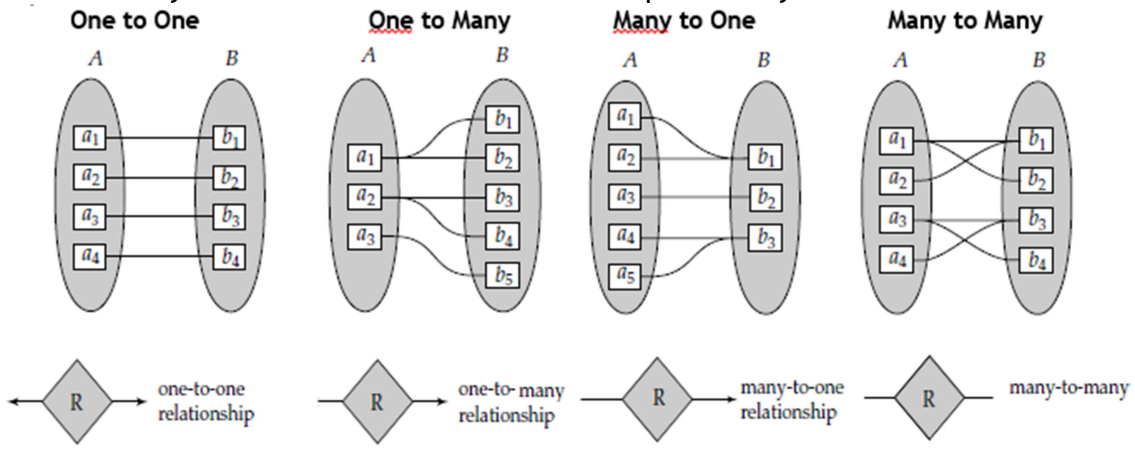


Constraints

Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the **data integrity** during an update/delete/insert into a table.

Constraints are divided into

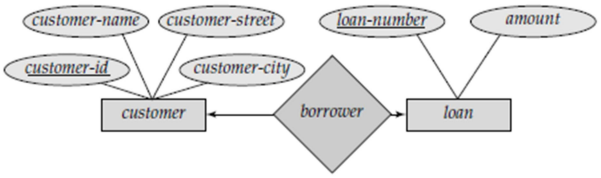
1. **Mapping Cardinalities**
   Mapping cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set. They are:



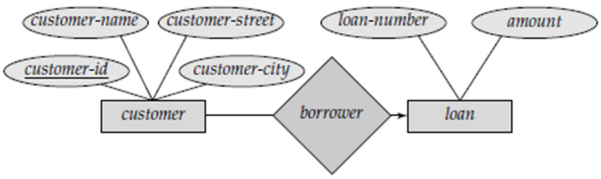
2. **Participation Constraints**
   Total Participation
   The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R
   Partial Participation
   If only some entities in E participate in relationships in R, the participation of entity set E in relationship R is said to be partial

ER Diagram Representation:

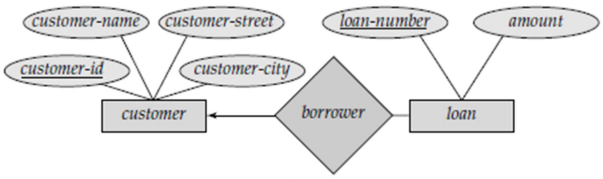
1. **One-to-One Relationship: From Customer to Loan**



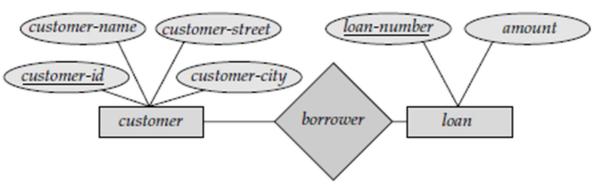
3. **Many to One: From Customer to Loan**



2. **One-to-Many: From Customer to Loan**



4. **Many to Many: From Customer to Loan**



Keys

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.

A key allows us to identify a set of attributes that suffice to distinguish entities from each other. Keys also help uniquely identify relationships, and thus distinguish relationships from each other.

1. Super Key (or Key)
2. Candidate Key
3. Primary Key
4. Foreign Key
5. Secondary/Alternate Key
6. Simple Key
7. Compound Key
8. Composite Key
9. Surrogate Key
10. Unique key
11. Partial Key

Super Key (or Key): Super Key is defined as a set of attributes within a table that can uniquely identify each record within a table.

Super Key is a superset of Candidate key.

Example: In the table defined above super key would include

1. Student ID
2. (Student ID, Student Name)
3. Phone

Candidate Key: Candidate keys are defined as the minimal set of fields which can uniquely identify each record in a table. It is an attribute or a set of attributes that can act as a Primary Key for a table to uniquely identify each record in that table.

NOTE:

- ✓ There can be more than one candidate key. Minimal Super Key is called Candidate Key
- ✓ A candidate key can never be NULL or empty. And its value should be unique.
- ✓ There can be more than one candidate keys for a table.
- ✓ A candidate key can be a combination of more than one columns (attributes).

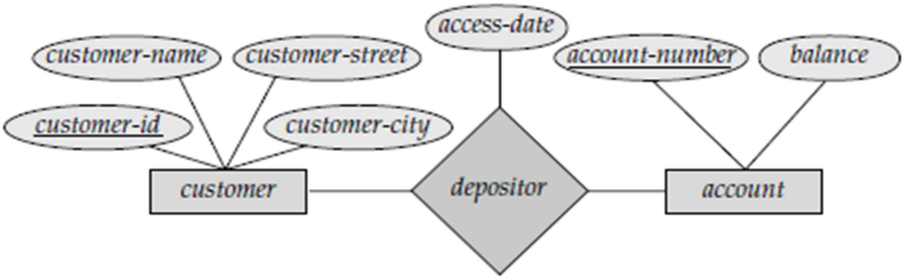
Example: In our example, below are candidate keys for table Student.

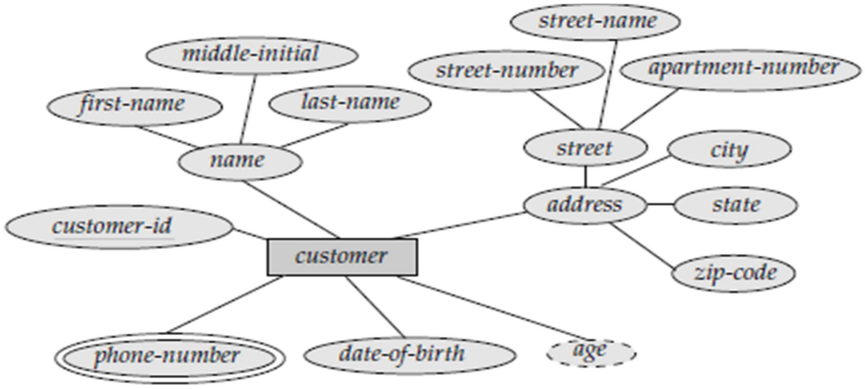
1. Student ID
2. Phone
3. both

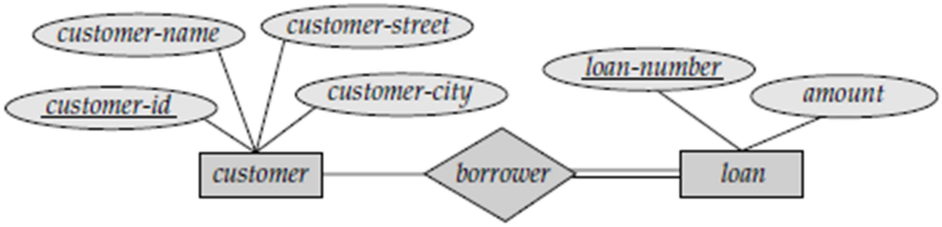
Primary Key: Primary key is a candidate key that is most appropriate to become the main key for any table. It is a key that can uniquely identify each record in a table.

Example: For the table Student we can make the Student ID column as the primary key.

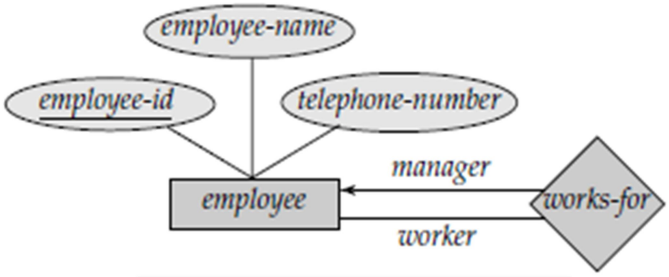
ER Diagram



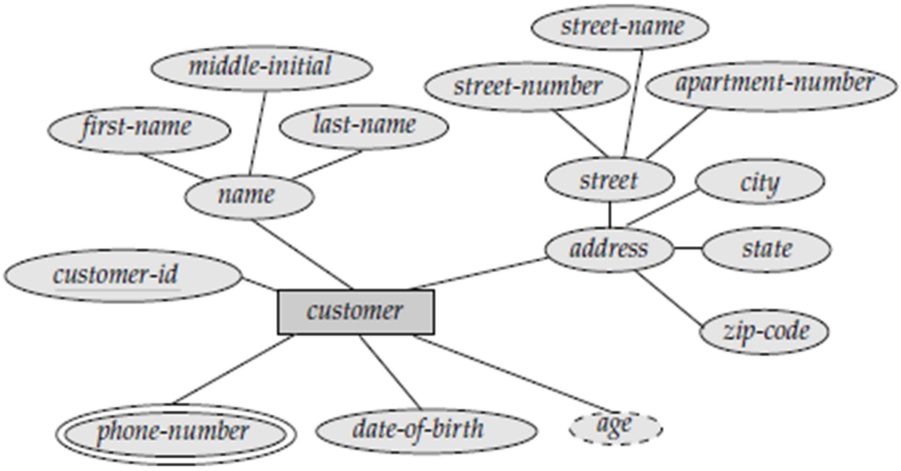




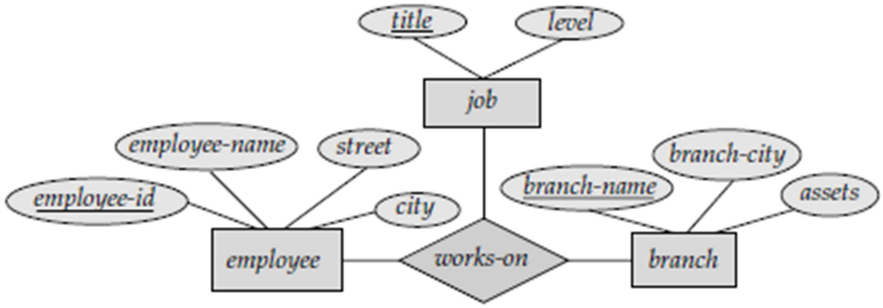
Total participation of an entity set in a relationship set.



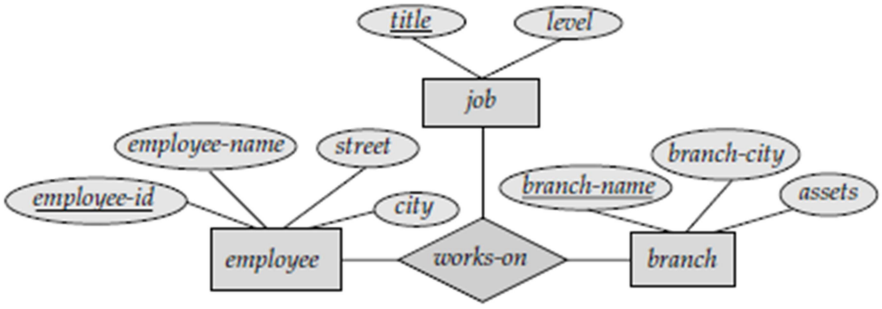
E-R diagram with role indicators.



E-R diagram with composite, multivalued, and derived attributes.

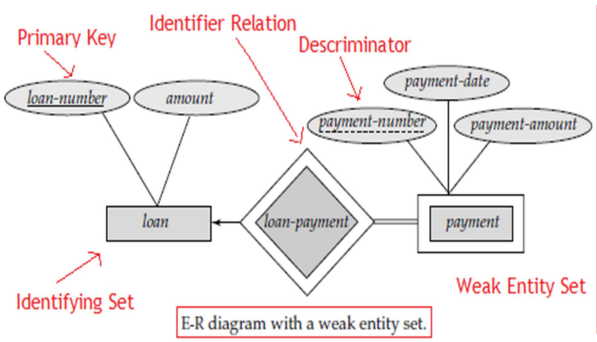


E-R diagram with a ternary relationship.

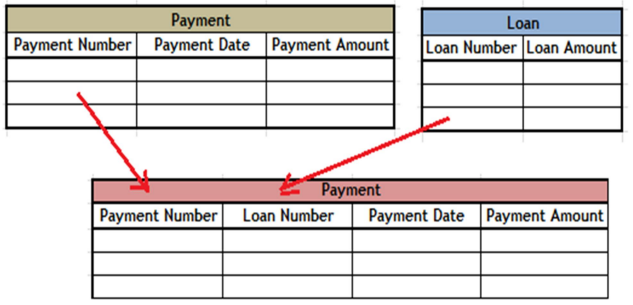


E-R diagram with a ternary relationship.

Weak Entity Set



E-R diagram with a weak entity set.



An entity set **may not have** sufficient attributes to form a primary key. Such an entity set is termed a **weak entity set**. For a weak entity set to be meaningful, it must be associated with another entity set, called the **identifying or owner entity set**.

- Every weak entity must be associated with an identifying entity i.e., the weak entity set is said to be existence dependent on the identifying entity set.
- The identifying entity set is said to own the weak entity set that it identifies.
- The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.

Note: Identifying Relationship set

It should have no descriptive attributes, since any required attributes can be associated with the weak entity set

It is many to one from the weak entity set to the identifying entity set, and

The participation of the weak entity set in the relationship is total.

Although a weak entity set does not have a primary key, we nevertheless need a means of distinguishing among all those entities in the weak entity set that depend on one particular strong entity. The discriminator of a weak entity set is a set of attributes that allows this distinction to be made.

Aggregation

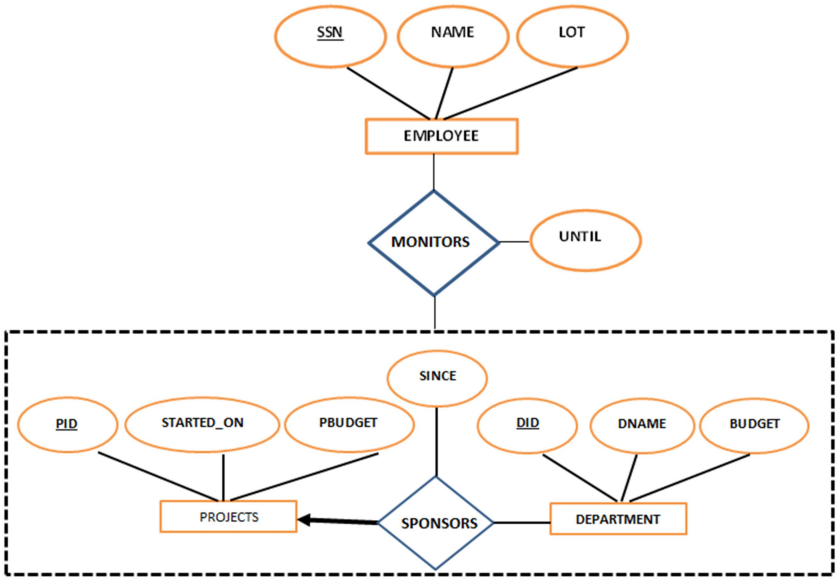
Aggregation is an abstraction through which relationships are treated as higher-level entities or simply a relationship participates in another relationship.

Sometimes, we have to model a relationship between a collection of entities and relationships. Aggregation allows us to indicate that a relationship set (identified through a dashed box) participates in another relationship set.

Example:

- Suppose that we have an entity set called Projects and that each Projects entity is sponsored by one or more departments. The Sponsors relationship set captures this information.
- A department that sponsors a project might assign employees to monitor the sponsorship.
- Intuitively, Monitors should be a relationship set that associates a Sponsors relationship (rather than a Projects or Departments entity) with an Employees entity. However, we have defined relationships to associate two or more entities.

With a dashed box around Sponsors (and its participating entity sets) used to denote aggregation. This effectively allows us to treat Sponsors as an entity set for purposes of defining the Monitors relationship set.



The Monitors relationship has an attribute until that records the date until when the employee is appointed as the sponsorship monitor.

Compare this attribute with the attribute since of Sponsors, which is the date when the sponsorship took effect. The use of aggregation versus a ternary relationship may also be guided by certain integrity constraints

Design Issues

Conceptual Design with the ER Model

1. Entity versus Attribute
2. Entity versus Relationship
3. Binary versus Ternary Relationships
4. Aggregation versus Ternary Relationships

Entity Vs Attribute

While identifying the attributes of an entity set, it is sometimes not clear whether a property should be modeled as an attribute or as an entity set (and related to the first entity set using a relationship set).

Example: Consider adding "Address" information to the Employees entity set.

1. Use an attribute address.

   Note: This option is appropriate if we need to record only one address per employee, and it suffices to think of an address as a string.

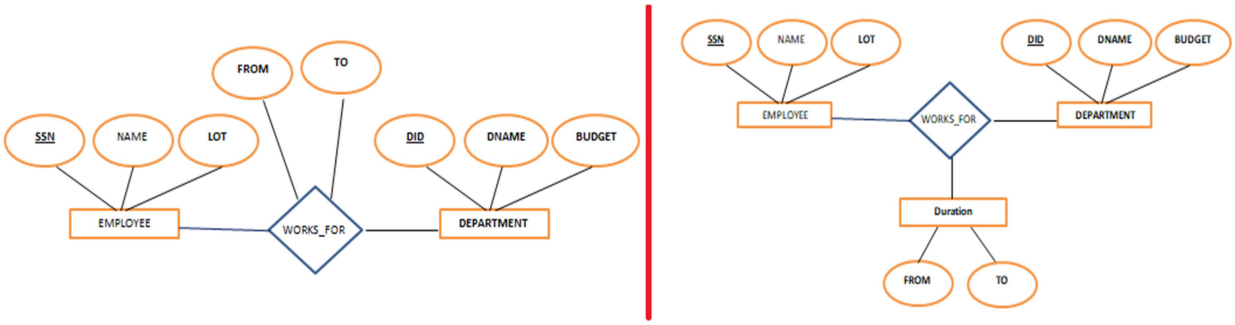
2. Create an entity set called ADDRESS

   [and to record associations between employees and addresses using a relationship (say, HAS_ADDRESS)]

   This more complex alternative is necessary in two situations:

   1. We have to record more than one address for an employee.
   2. We want to capture the structure of an address in our ER diagram.
      - For example, we might break down an address into city, state, country, and Zip code, (in addition to a string for street information).
      - By representing an address as an entity with these attributes, we can support queries such as

        "Find all employees with an address in Madison, WI."



Entity Vs Relationship

Consider the relationship set called "Manages".

Example:

Suppose that each department manager is given a "Discretionary Budget (DBUDGET)", as shown below. Given a department, we know the manager, as well the manager's starting date and budget for that department. (if we assume that a manager receives a separate discretionary budget for each department that he or she manages.)

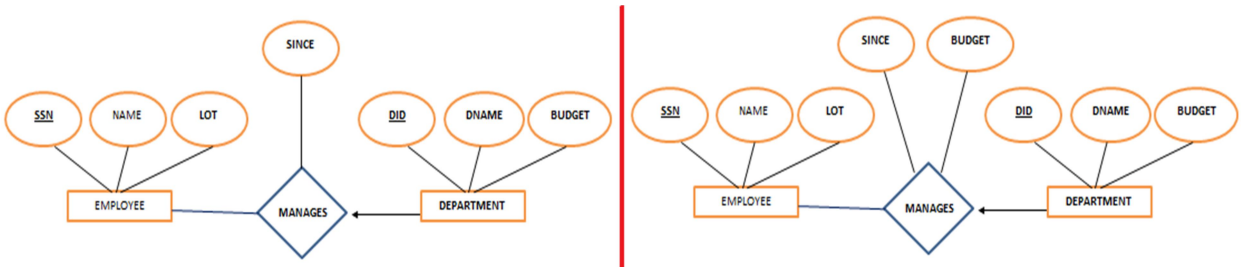
Problem: But what if the **"Discretionary Budget (DBUDGET)** is a sum that **covers all departments** managed by that employee?

a. In this case, each "Manages" relationship that involves a given employee will have the same value in the "DBUDGET" field, leading to redundant storage of the same information.

b. Another problem with this design is that it is misleading; it suggests that the budget is associated with the relationship, when it is actually associated with the manager.

Solution: By introducing a new entity set called Managers (which can be placed below Employees in an ISA hierarchy, to show that every manager is also an employee).

- The attributes "SINCE" and "DBUDGET" now describe a manager entity, as intended. As a variation, while every manager has a budget, each manager may have a different starting date (as manager) for each department.
- In this case "DBUDGET" is an attribute of Managers, but since is an attribute of the relationship set between managers and departments



Binary Vs Ternary Relationship

It models a situation in which an employee can own several policies, each policy can be owned by several employees, and each dependent can be covered by several policies.

Suppose that we have the following additional requirements:

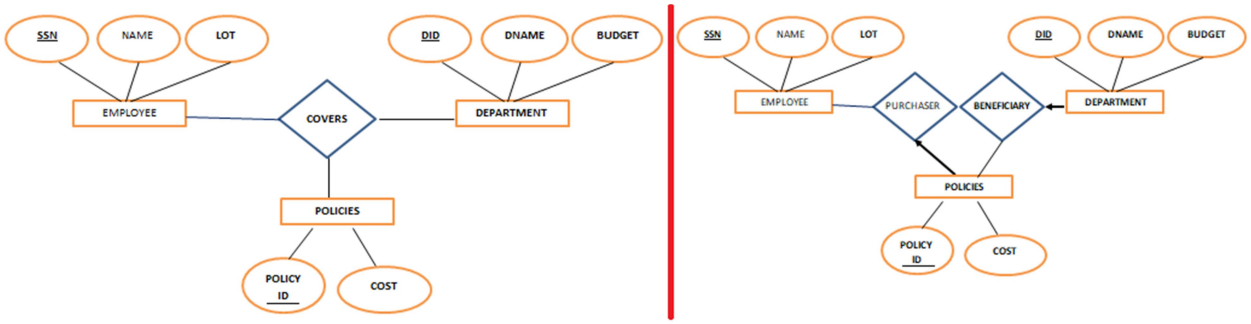
1. A policy cannot be owned jointly by two or more employees.
2. Every policy must be owned by some employee.
3. Dependent is a weak entity set, and each dependent entity is uniquely identified by taking PNAME in conjunction with the POLICYID of a policy entity (which, intuitively, covers the given dependent).

The **first requirement** suggests that we impose a key constraint on POLICIES with respect to COVERS, but this constraint has the unintended side effect that a policy can cover only one dependent.

The **second requirement** suggests that we impose a total participation constraint on POLICIES. This solution is acceptable if each policy covers at least one dependent.

The **third requirement** forces us to introduce an identifying relationship that is binary (in our version of ER diagrams, although there are versions in which this is not the case).

Even ignoring the third requirement, the best way to model this situation is to use two binary relationships, This example really has two relationships involving Policies, and our attempt to use a single ternary relationship is inappropriate. There are situations, however, where a relationship inherently associates more than two entities.



Aggregation Vs Ternary Relationships

The choice between using aggregation or a ternary relationship is mainly determined by the existence of a relationship that relates a relationship set to an entity set (or second relationship set). The choice may also be guided by certain integrity constraints that we want to express.

According to this diagram, a project can be sponsored by any number of departments, a department can sponsor one or more projects, and each sponsorship is monitored by one or more employees.

NOTE: If we don't need to record the UNTIL attribute of Monitors, then we might reasonably use a ternary relationship

Reason for Aggregation rather than Ternary Relationship Set:
Consider the constraint, that each sponsorship (of a project by a department) **be monitored by at most one employee**. We cannot express this constraint in terms of the SPONSORS relationship set. Thus, the presence of such a constraint serves as another reason for using aggregation rather than a ternary relationship set.

Class Hierarchies

1. **Specialization:**
   The process of designating subgroupings within an entity set is called specialization.

2. **Generalization:**
   The refinement from an initial entity set into successive levels of entity subgroupings represents a **top-down design** process in which distinctions are made explicit.
   The design process may also proceed in a **bottom-up manner**, in which multiple entity sets are synthesized into a higher-level entity set on the basis of common features.

   Higher- and lower-level entity sets also may be designated by the terms **superclass** and **subclass**, respectively.

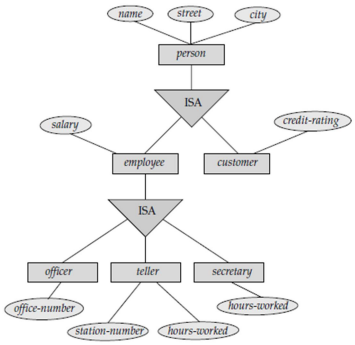
   Example: Person entity set is the **superclass of the customer**
   Employee entity set is the **subclass of the customer**

3. **Higher and Lower-Level Entity Sets:**
   A crucial property of the higher- and lower-level entities created by specialization and generalization is attribute inheritance.

4. **Attribute Inheritance**
   The attributes of the higher-level entity sets are said to be inherited by the lower-level entity sets.

   Note:
   - A higher-level entity set with attributes and relationships that apply to all of its lower-level entity sets
   - Lower-level entity sets with distinctive features that apply only within a particular lower-level entity set

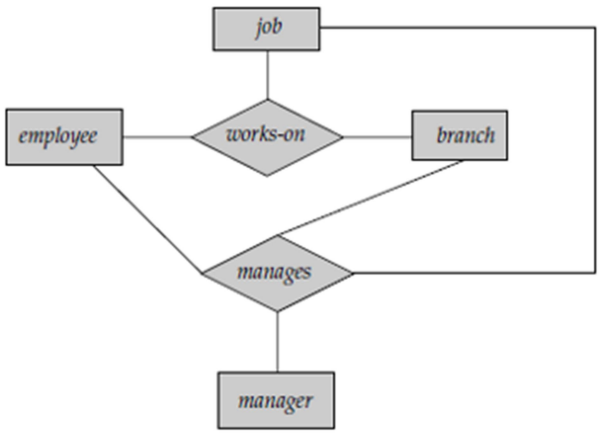
5. **Aggregation**
   Aggregation is an abstraction through which relationships are treated as higher-level entities or simply a relationship participates in another relationship.



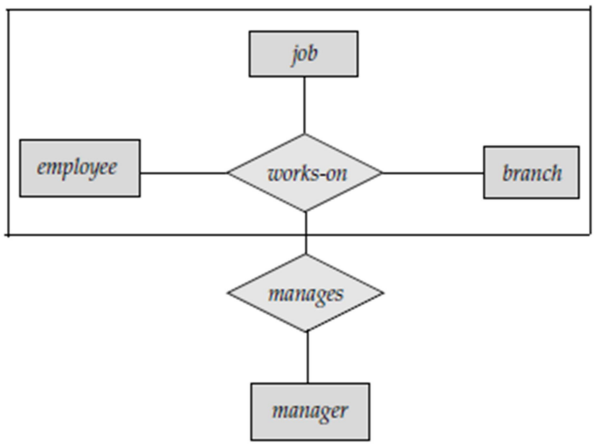
Specialization and generalization.

Redundant Relationships Vs Aggregation

Aggregation is an abstraction through which relationships are treated as higher level entities or simply a relationship participates in another relationship.



E-R diagram with redundant relationships.



E-R diagram with aggregation.

Relationship set works-on (relating the entity sets employee, branch, and job) as a higher-level entity set called works-on.

Such an entity set is treated in the same manner as is any other entity set. We can then create a binary relationship manages between works-on and manager to represent who manages what tasks.

Constraints on Generalization

1. Constraint involves determining which entities can be members of a given lower-level entity set. Such membership may be one of the following:

    a. **Condition-defined:** In condition-defined lower-level entity sets, membership is evaluated on the basis of whether or not an entity satisfies an explicit condition or predicate. For example, assume that the higher-level entity set account has the attribute account-type.
    b. **Attribute-defined:** All the lower-level entities are evaluated on the basis of the same attribute (in this case, on account-type), this type of generalization is said to be attribute-defined.
    c. **User-defined:** User-defined lower-level entity sets are not constrained by a membership condition; rather, the database user assigns entities to a given entity set.

2. Constraint relates to whether or not entities may belong to more than one lower-level entity set within a single generalization. The lower-level entity sets may be one of the following:

    a. **Disjoint:** A disjointness constraint requires that an entity belong to no more than one lower-level entity set. In our example, an account entity can satisfy only one condition for the account-type attribute; an entity can be either a savings account or a checking account, but cannot be both.
    b. **Overlapping:** In overlapping generalizations, the same entity may belong to more than one lower-level entity set within a single generalization

        Example: Suppose generalization applied to entity sets customer and employee leads to a higher-level entity set person. The generalization is overlapping if an employee can also be a customer.

3. The completeness constraint on a generalization or specialization, specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within the generalization/specialization. This constraint may be one of the following:

    - **Total generalization or specialization:** Each higher-level entity must belong to a lower-level entity set.
    - **Partial generalization or specialization:** Some higher-level entities may not belong to any lower-level entity set.

        Partial generalization is the default. We can specify total generalization in an E-R diagram by using a double line to connect the box representing the higher-level entity set to the triangle symbol. (This notation is similar to the notation for total participation in a relationship.)