ENTITY–RELATIONSHIP MODELING

Concepts of the ER Model

- Entity types
- Relationship types
- Attributes

Entity Type

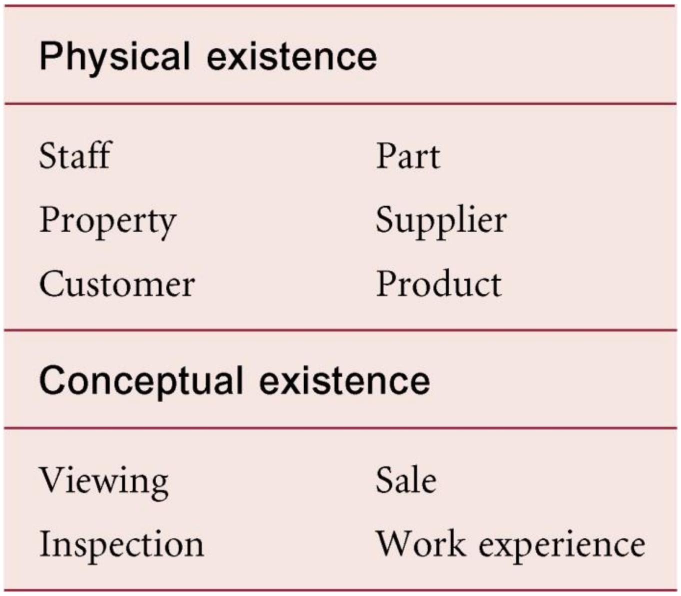
- Entity type

○ Group of objects with same properties, identified by enterprise as having an independent existence.

- Entity occurrence

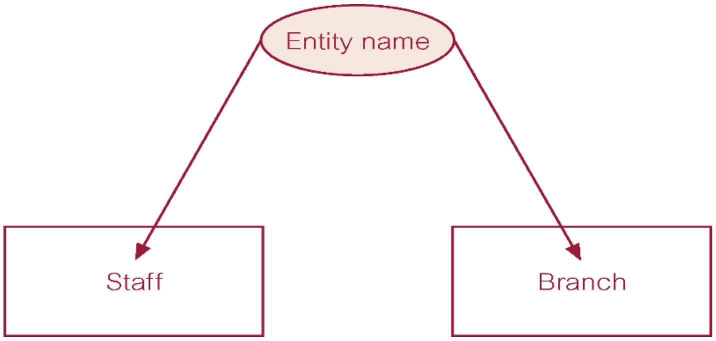
Uniquely identifiable object of an entity type.

Examples of Entity Types



Physical existence	
Staff	Part
Property	Supplier
Customer	Product
Conceptual existence	
Viewing	Sale
Inspection	Work experience

ER diagram of Staff and Branch entity types



Relationship Types

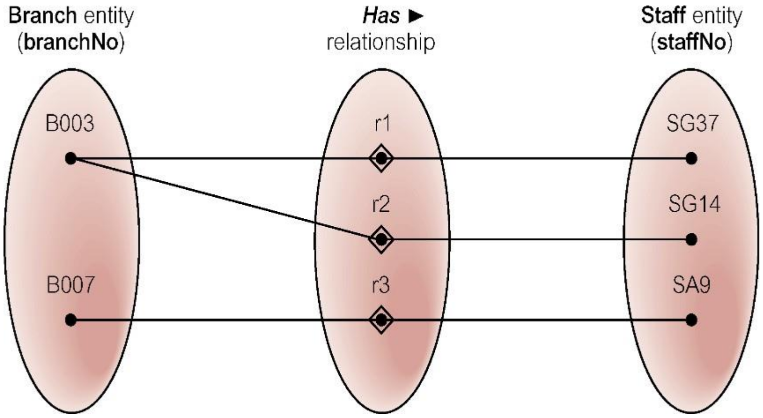
- ➢ Relationship type

– Set of meaningful associations among entity types.

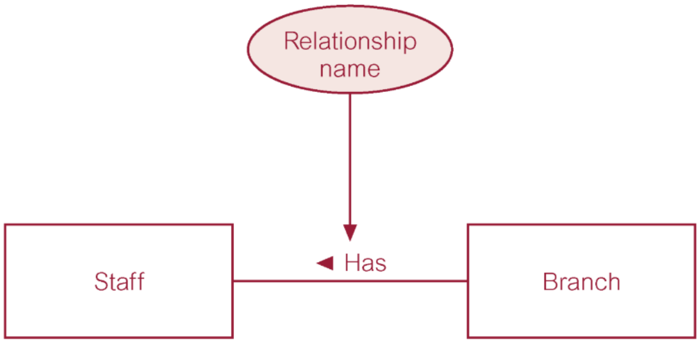
- ➢ Relationship occurrence

– Uniquely identifiable association, which includes one occurrence from each participating entity type.

Semantic net of *Has* relationship type



ER diagram of Branch *Has* Staff relationship



'Branch has staff'

Degree of a Relationship

– Number of participating entities in relationship.

– Relationship of degree :

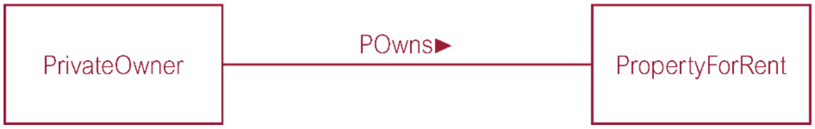
✓ two is binary

✓ three is ternary

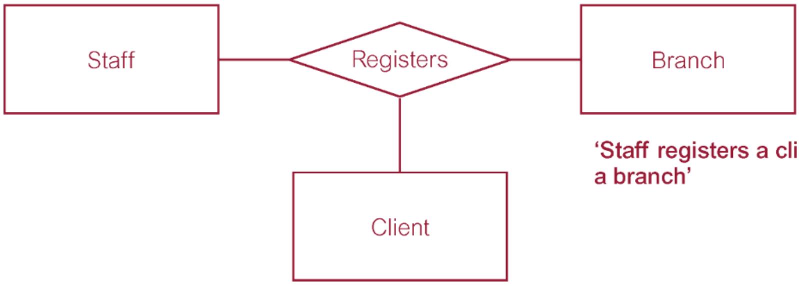
✓ four is quaternary.

Binary relationship called *POwns*

'Private owner owns property for rent'

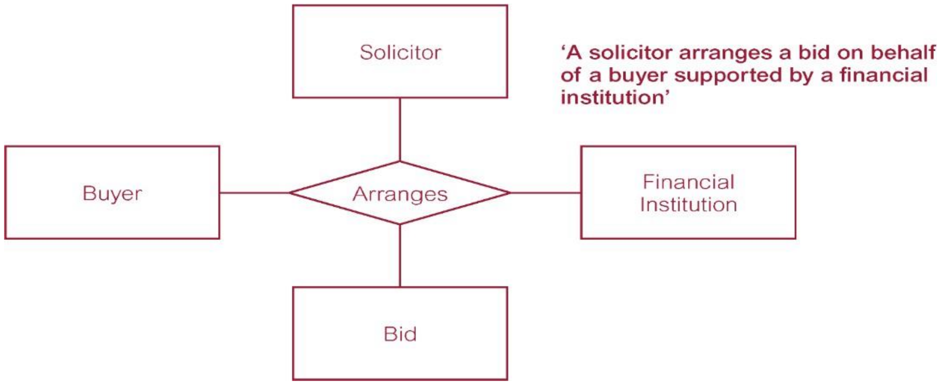


Ternary relationship called *Registers*



'Staff registers a client at a branch'

Quaternary relationship called *Arranges*



'A solicitor arranges a bid on behalf of a buyer supported by a financial institution'

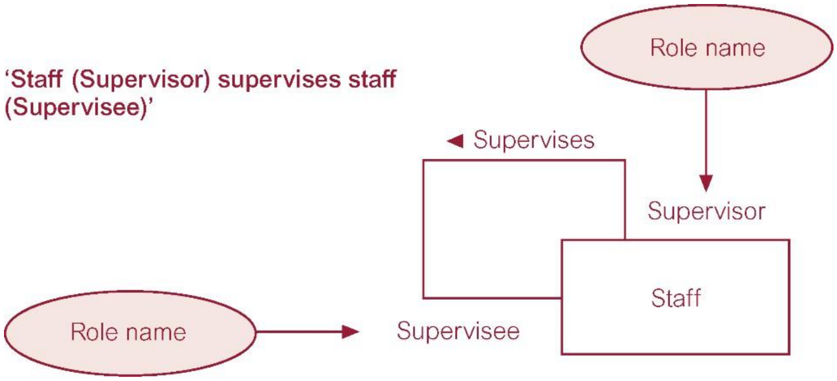
Recursive Relationship

– Relationship type where *same* entity type participates more than once in *different roles*.

– Relationships may be given role names to indicate purpose that each participating entity type plays in a relationship.

Recursive relationship called *Supervises* with role names

'Staff (Supervisor) supervises staff (Supervisee)'



Attributes

- ➢ Attribute
- ○ Property of an entity or a relationship type.
- ➢ Attribute Domain
- ○ Set of allowable values for one or more attributes.

Attribute types

An attribute can be classified into various types.

(i) ***Simple attribute*****:** An attribute that cannot be divided into further subparts.

Example: Customer-id of customer entity.

(ii) ***Composite attribute*****:** An attribute that can be divided into a set of subparts.

Example: In a customer entity, the attribute name can further be divided into first-name, middle-name, last-name.

(iii) ***Single value attribute*****:** An attribute having only one value in a particular entity.

Example: In a customer entity, name, id, street are single valued attributes.

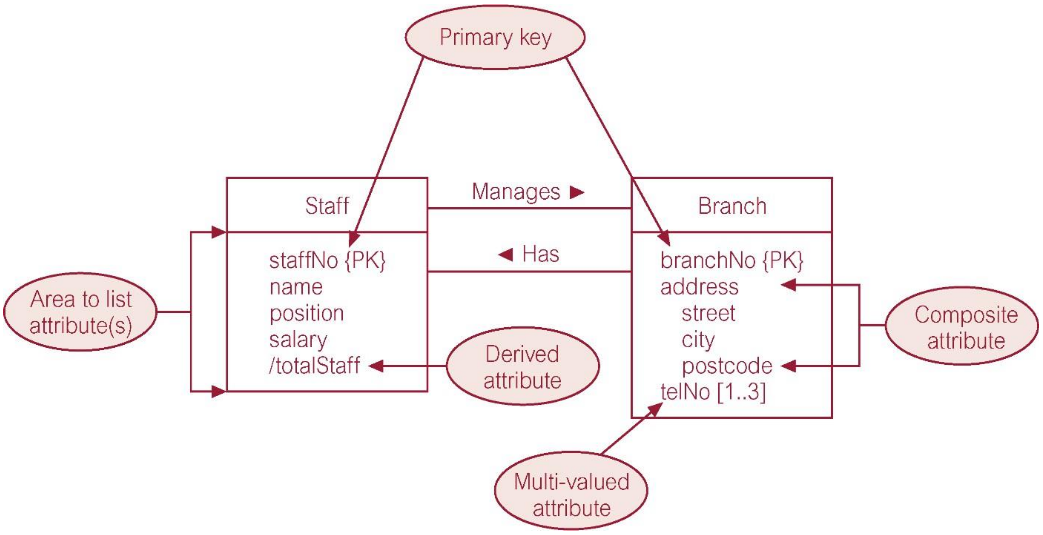
(iv) ***Multi-valued attribute*****:** An attribute having more than one value for a particular entity.

Example: Consider the customer entity set with the attribute phone no. A customer may have zero, one or several phone nos and different customers may have different numbers of phone.

(v) ***Derived attribute*****:** An attribute that is derived from other related attributes or entities.

For example, the age of a customer entity set is derived from the attribute date-of-birth of a customer.

ER diagram of Staff and Branch entities and their attributes



Keys

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

Different types of keys are:

1. Super key

2. Candidate key

3. Primary key

4. Foreign key

1. **Super key:** A **super key** is a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation. For example, consider the student relation.
Student (Rollno, Name, Age) is a *Super key.*

2. **Candidate key:** A table which have more than one attribute that uniquely identify an instance of an entity set. These attributes are called *Candidate keys*.
For example, Consider the car relation,
Car (license_no, engine_serial_no, make, model, year) In this relation, we can find the two
Candidate keys:
license_no and engine_serial_no.

3. **Primary key:** An attribute which is unique and not null, can identify an instance of the entity set is termed as *Primary key*.
For example, Consider the employee relation,
Employee (eno, ename, sex, doj, dob, sal, job, dno) in this eno is the *Primary key.*

4. **Foreign key:** An attribute in one relation whose value matches the primary key in some other relation is called a *Foreign key*.
For example, Consider the two relations dept and employee,

Dept (dno, dname, dloc)
Employee (eno, ename, sex, doj, dob, sal, job, dno)
In the above relations, for dept relation dno is the *primary key*, for employee

relation eno is the
primary key and here we can find that the employee relation - dno matches with the dept relation
- dno, so that employee relation dno is known as *Foreign key*.

Strong and Weak Entity Types

- **Strong Entity Type**

  Entity types that have key attributes of their own are called strong entity types.

  Entity type that is *not* existence-dependent on some other entity type.

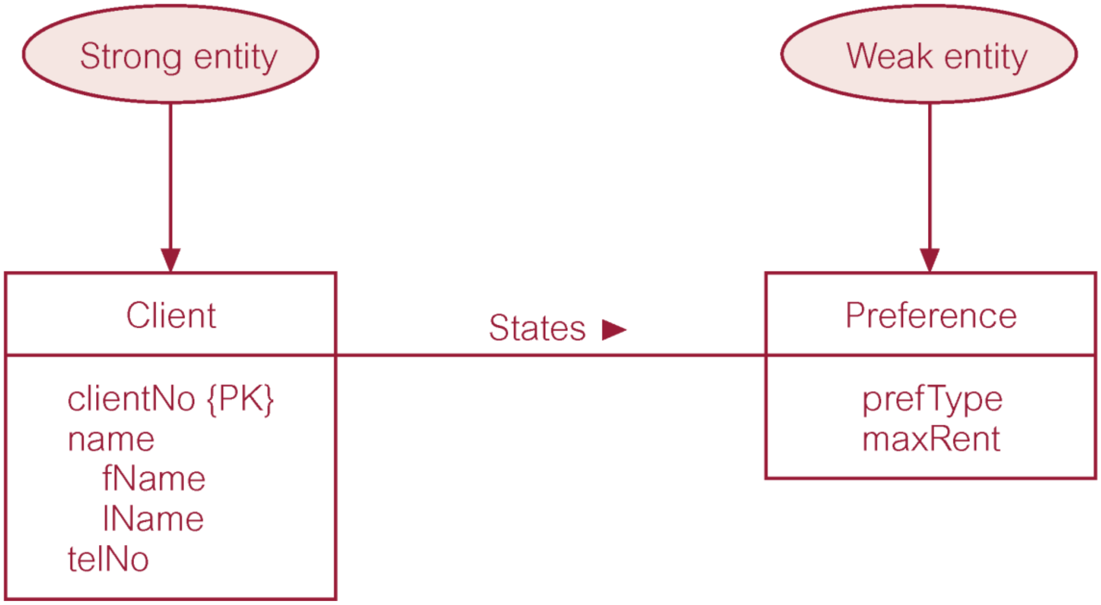
- ## Weak Entity Type

  Entity types that do not have key attributes of their own are called weak entity types.

– Entity type that is existence-dependent on some

other entity type. Strong entity type called Client and weak entity

type called Preference



Structural Constraints

- Main type of constraint on relationships is called multiplicity.

- Multiplicity - number (or range) of possible occurrences of an entity type that may relate to a single occurrence of an associated entity type through a particular relationship.

- Represents policies (called *business rules*) established by user or company.

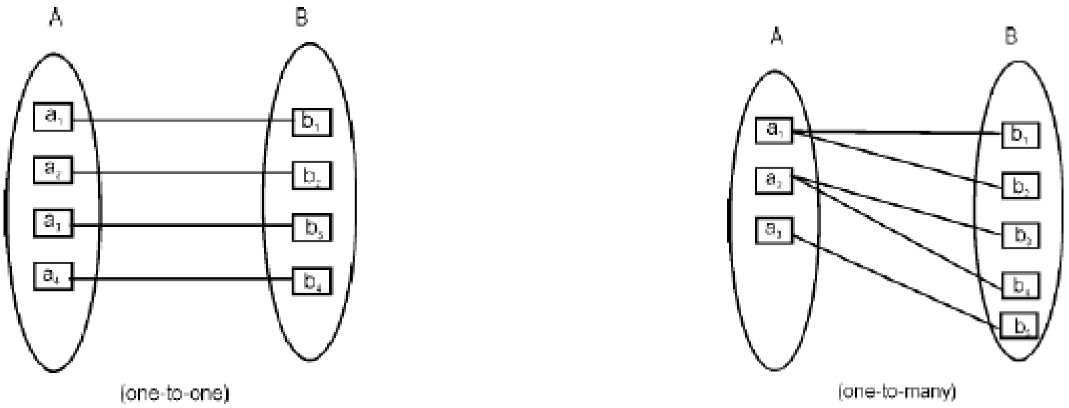
Mapping Cardinalities

Mapping cardinalities or Cardinality ratio is defined as the number entities to which another entity can be associated via a relationship set.

For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following

(i) One-to-one (1 : 1)

An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



(a)

(b)

Fig. 2.4 One-to-One & One-to-Many

(ii) One-to-many (1 : M)

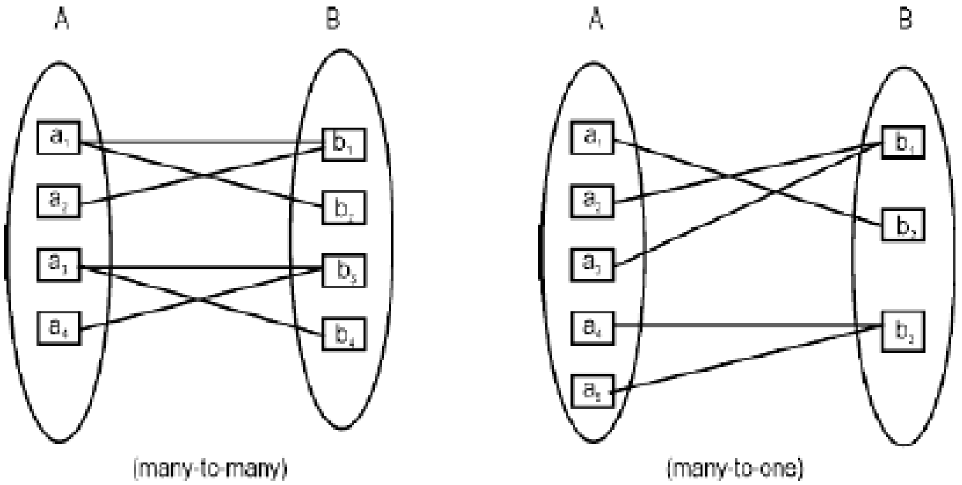
An entity in A is associated with any number of entities in B. An entity in B can be associated with at most one entity in A.

(iii) Many-to-Many (M : N)

An entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.

(iv) Many to one (M : 1)

An entity in A is associated with at most one entity in B. An entity in B can be associated with any number of entities in A.



(c) (d)

Fig. 2.5 Many-to-Many & Many-to-One

Problems with ER Models

- Problems may arise when designing a conceptual data model called connection traps.
- Often due to a misinterpretation of the meaning of certain relationships.
- Two main types of connection traps are called *fan traps* and *chasm traps*.
- Fan Trap

○ Where a model represents a relationship between entity types, but pathway between certain entity occurrences is ambiguous.

○ Chasm Trap

○ Where a model suggests the existence of a relationship between entity types, but pathway does not exist between certain entity occurrences.