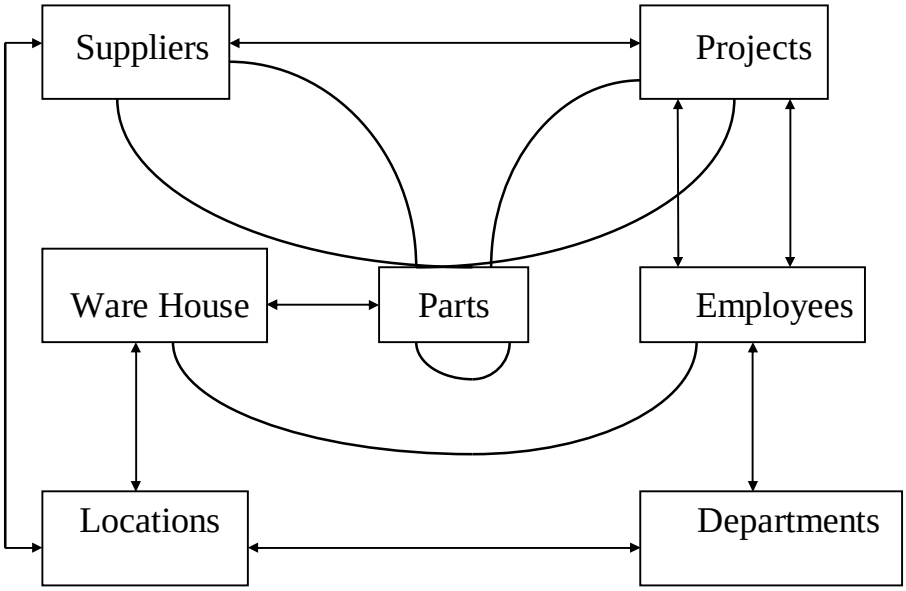
Entity Relationship

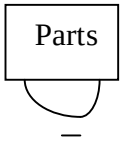
The term entity is widely used in database systems to mean any distinguishable object that is to be presented in the database. Entity is a thing that exists and is distinguishable. A group of all similar entities forms an entity. In general these will be associations or relationships linking the basic entities together.



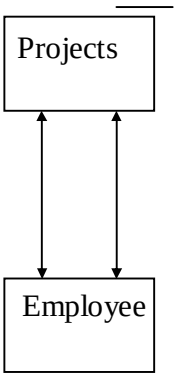
An example of operational data

In the above figure relationships are all bidirectional that is, they may all be traversed in either direction. So two queries can be answered by this :

a) Given an employee, find the corresponding departments
b) Given a department, find all corresponding employees



means a parts within a parts – a screw is a component of a hinge assembly, which is also considered as a parts.



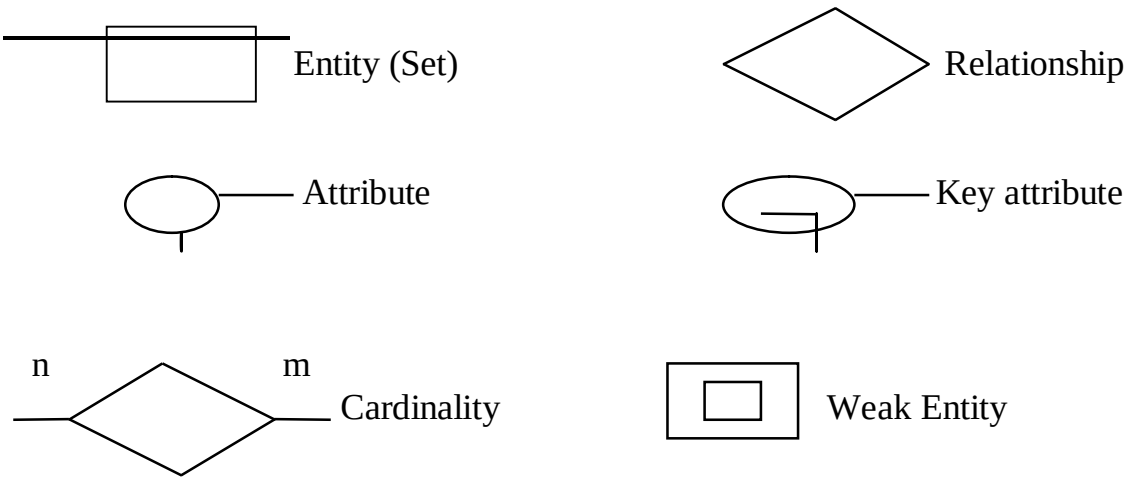
Projects and employees are linked in two relationships. One might represent the relationship "Works On" means the employee works on the project. The other relationship is "manager of" i.e. the employee is the manager of the project.

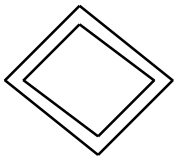
Though entities and relationships are two dissimilar types of objects, an association between entities may itself be considered as an entity .

The Figure also illustrates a point as follows :
Although most of the relationships in the fig. associates two types of entities, this is by no means always the case. In the example, there is an arrow connecting three types of entities – Suppliers – Parts – Projects.

"Supplier S2 supplies part P4 to project J3" tells us more than the combination "Supplier S2 supplies part P4" and "Part P4 is used in Project J3. We can't deduce the first of these three association knowing only the 2nd & 3rd.

Symbols used in E-R diagram

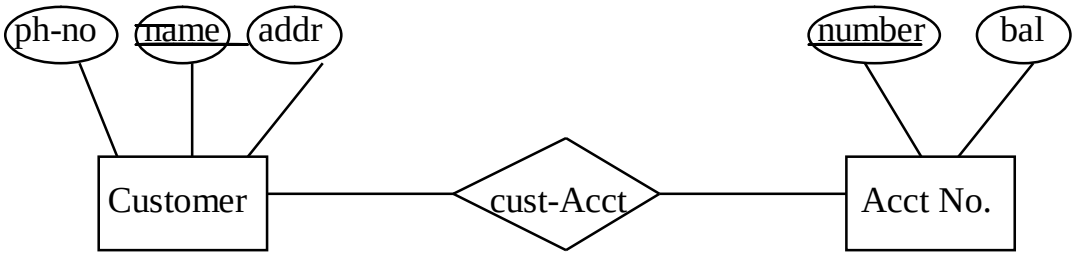




Strong Weak Relationship

The overall logical structure of a database can be expressed graphically by a E-R diagram.

Example



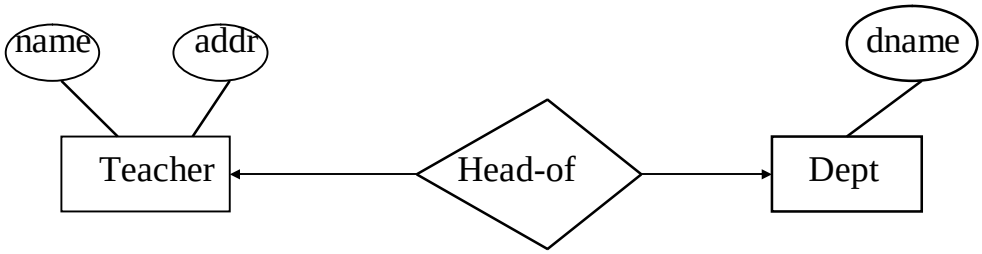
E-R diagram for a part of the database of a banking system

The set of all persons having an account at a bank can be defined as the entity set customer. Similarly the entity set account might represent the set of all accounts in a particular branch of a particular bank.

Mapping Constraints

One important constraint is mapping cardinalities which express the number of entities to which another entity can be associated via a relationship.

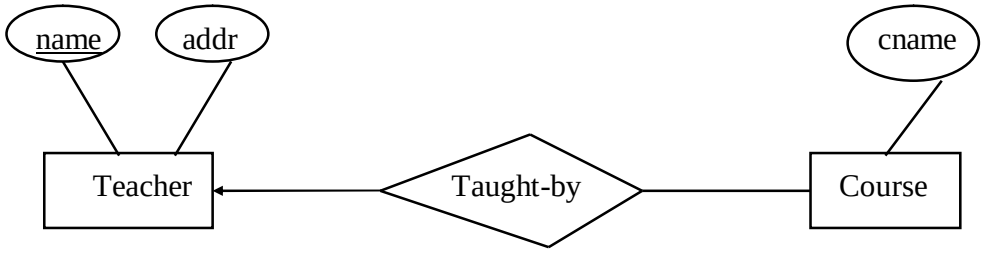
One -to-one :



One-to-one relationship

In the above figure there is a relationship Head-of connects two entities Teacher and Dept. If we assume that each department can have only one head and one teacher cannot be the head of more than one department.

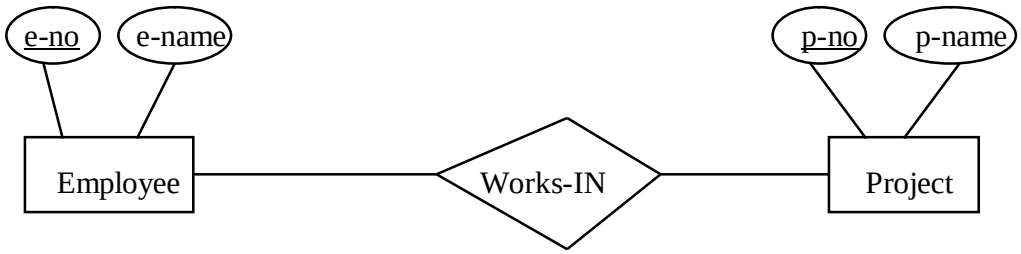
One-to-many A teacher can teach many courses but each course is taught by one teacher



One-to-many relationship

In the above example if we assume that each course is taught by one teacher only then the relationship is one to many between Teacher & Course. Similarly the same relationship is many-to-one from Course & Teacher.

Many-to-many relationship



Many-to-many relationship

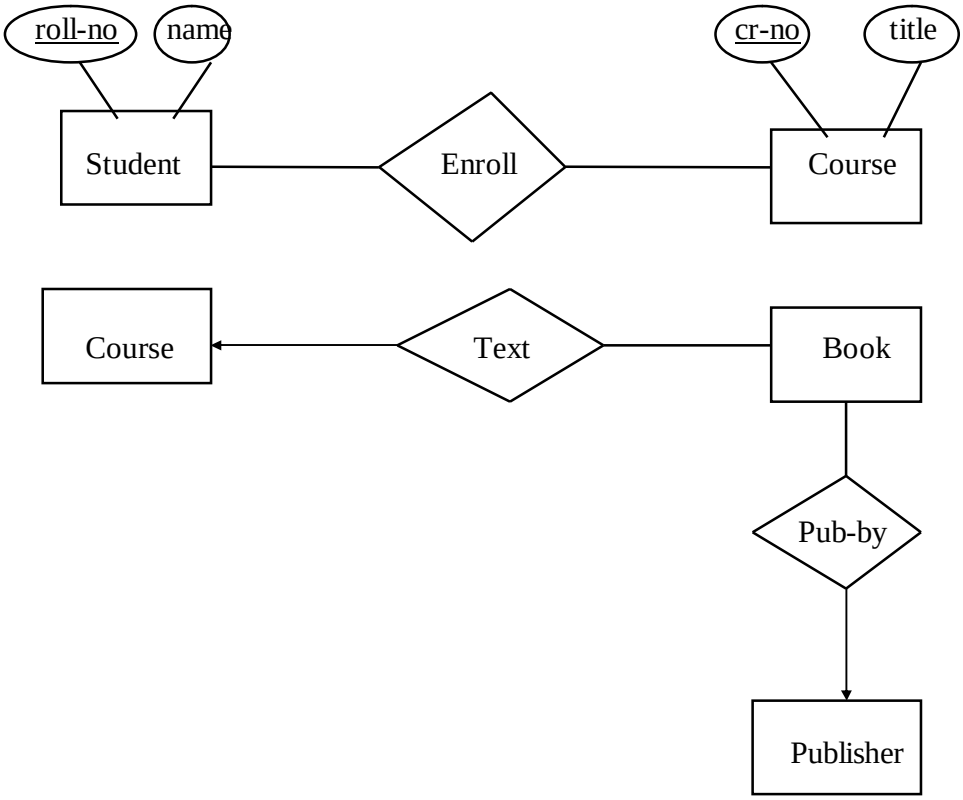
In this example one employee may work in several projects and one project may have several employees.

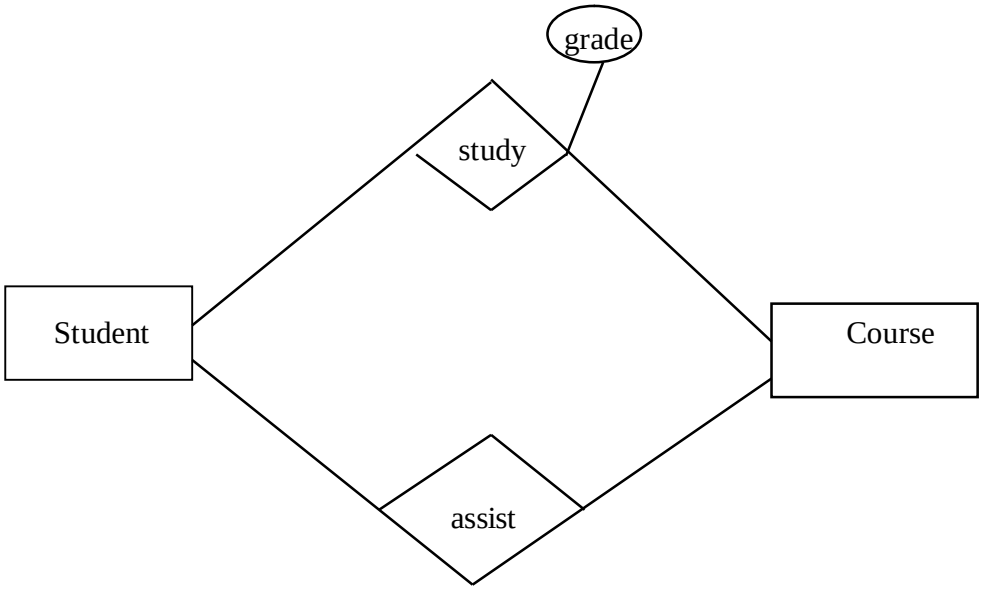
Key

A **super key** for an entity set is a set of one or more attributes whose value uniquely identifies the entities in the entity set. For a student {roll-no, name, address} can be

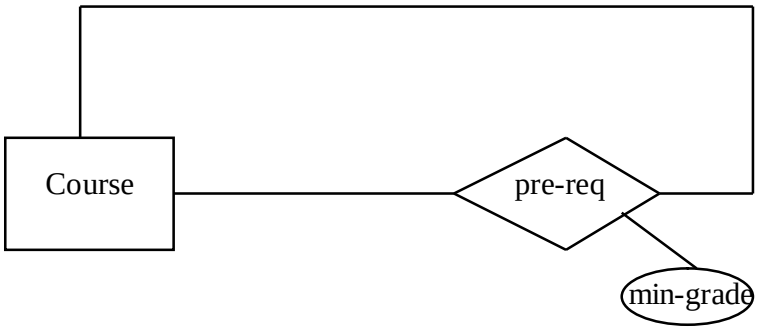
considered to be a superkey. A **candidate key** is a minimal superkey. A **primary key** is a candidate key that is chosen by the database designer.

More Examples

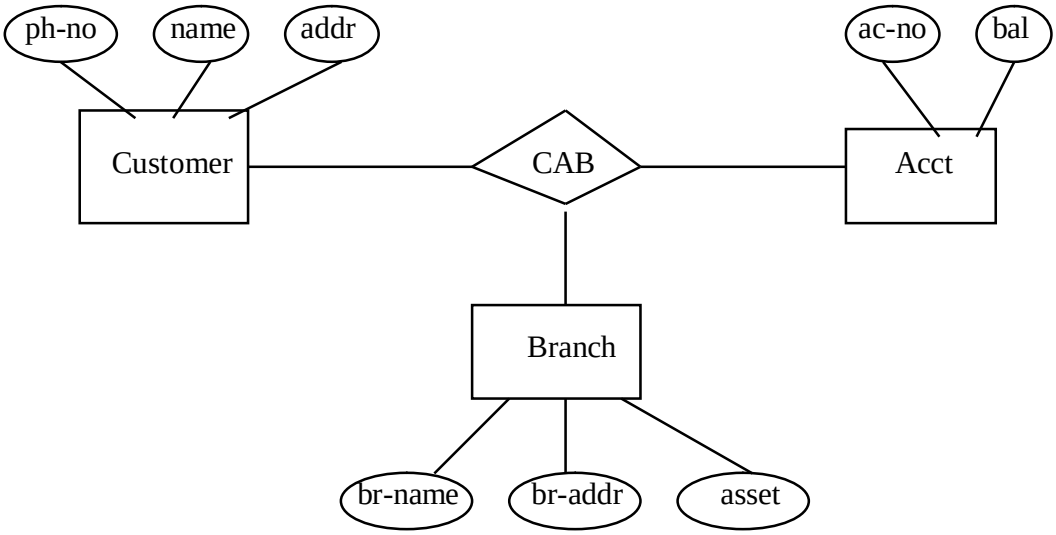




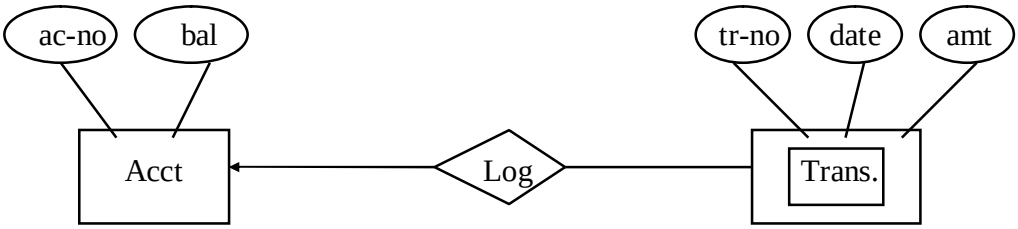
Multiple relationships



Self-relationship

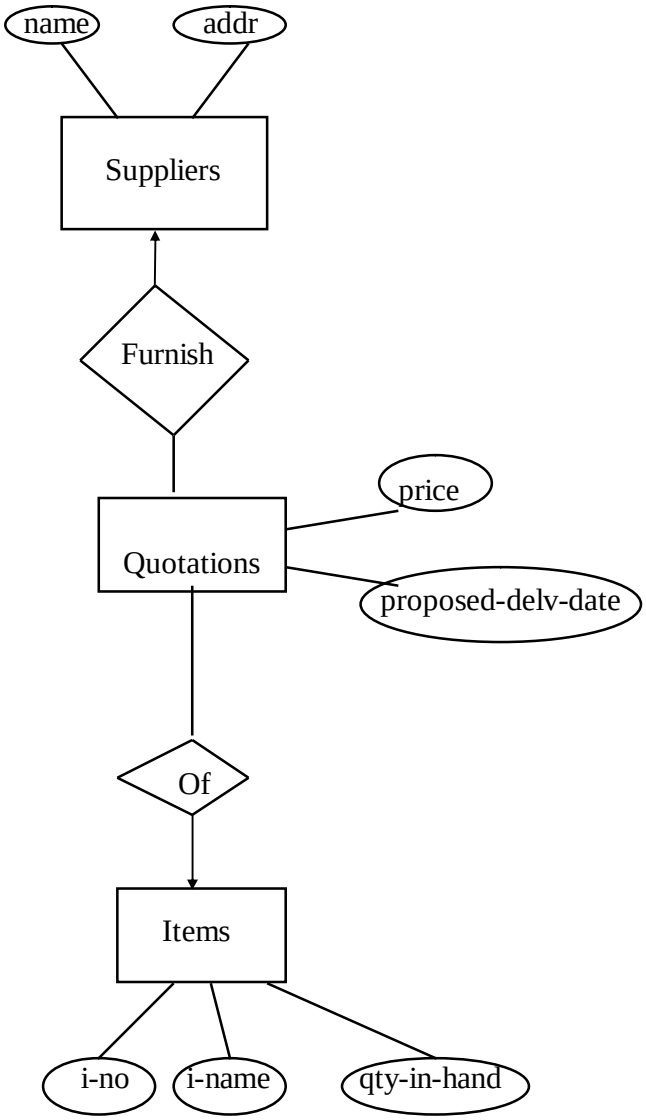


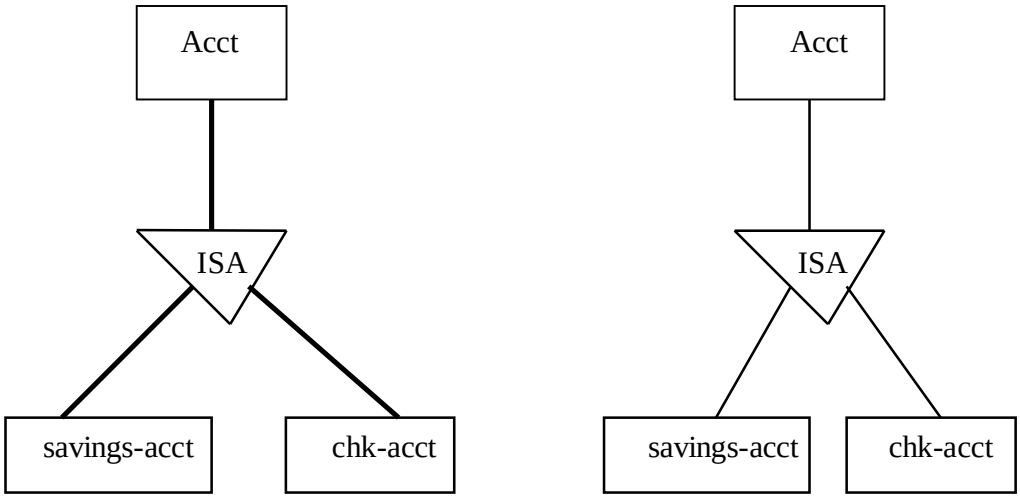
Ternary relationship



Weak entity

Although each transaction entity is distinct, transaction on different accounts may share the same tr-no. Here the key is {ac-no, tr-no}

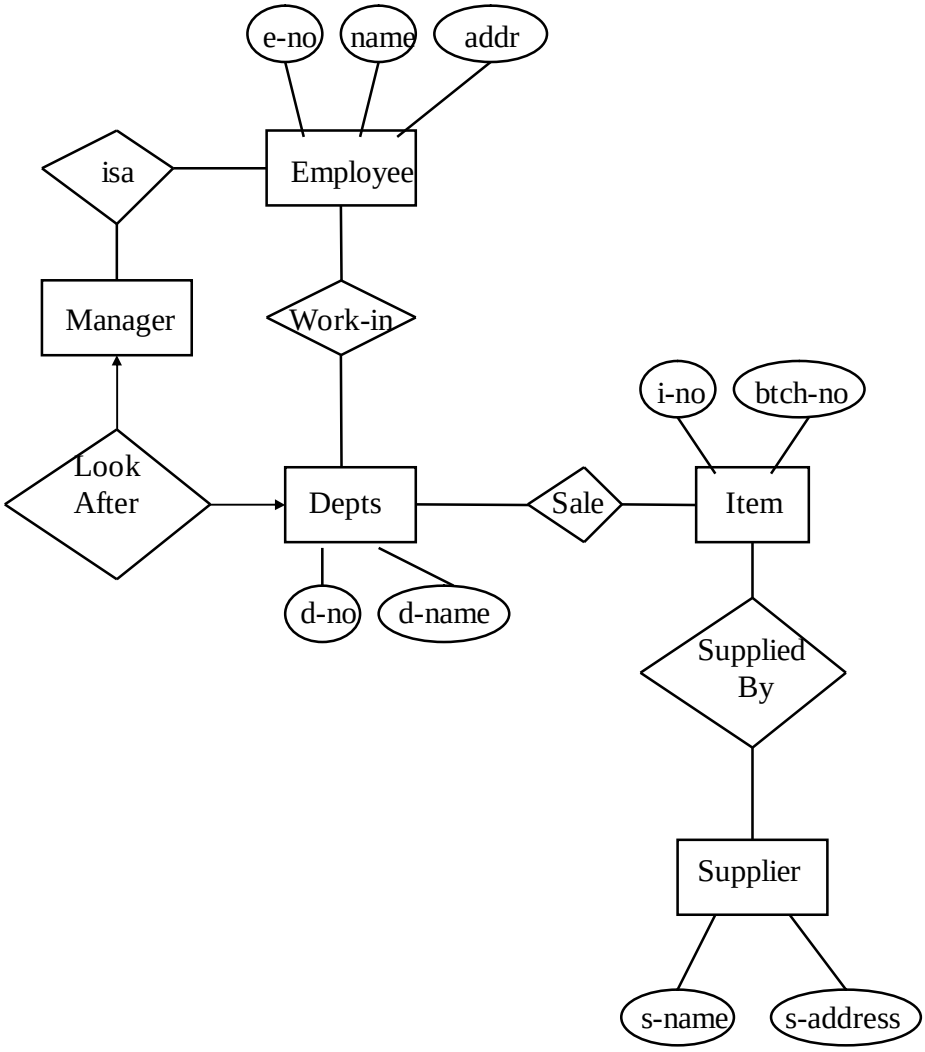




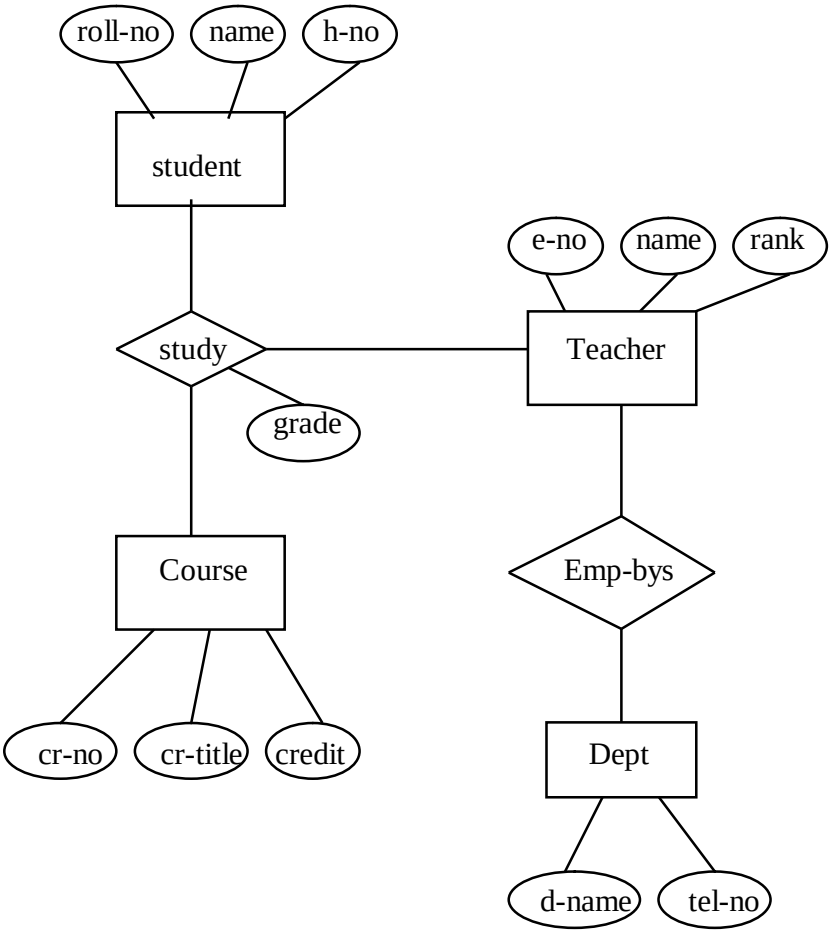
a) **Generalization** and b) **Specialization relationship**

It works on the principle of bottom up approach. In Generalization lower level functions are combined to form higher level function which is called as entities. This process is repeated further to make advanced level entities.

We can say that Specialization is opposite of Generalization. In Specialization things are broken down into smaller things to simplify it further. We can also say that in Specialization a particular entity gets divided into sub entities and it's done on the basis of it's characteristics. Also in Specialization Inheritance takes place.



E-R diagram for the database of a departmental store


roll-no
name
h-no
student
e-no
name
rank
study
Teacher
grade
Course
Emp-bys
cr-no
cr-title
credit
Dept
d-name
tel-no

Converting E-R diagrams into Relations

represent each strong entity set as a table

- one column for each simple attribute
- one row for each entity

representation of weak entities

- define a table consisting of primary key of strong entity (on which it depends) and attributes of weak entity

representation of relationship sets

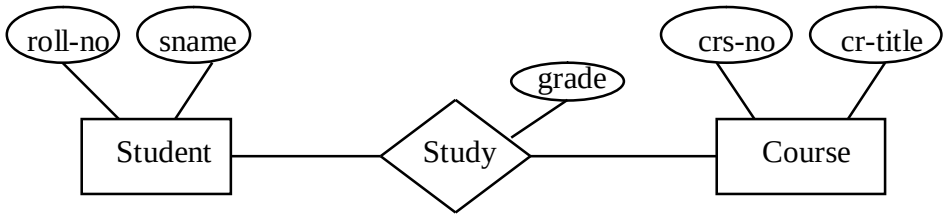
define a table consisting of

- primary keys of participating entity sets
- attributes of the relationship

for 1-to-1 and many-to-one binary relationships, separate tables need not be defined; the relationships can be included with tables for entities

for many-to-many binary relationships, separate tables are to be designed

Consider the figure below showing two entities and one relationship :



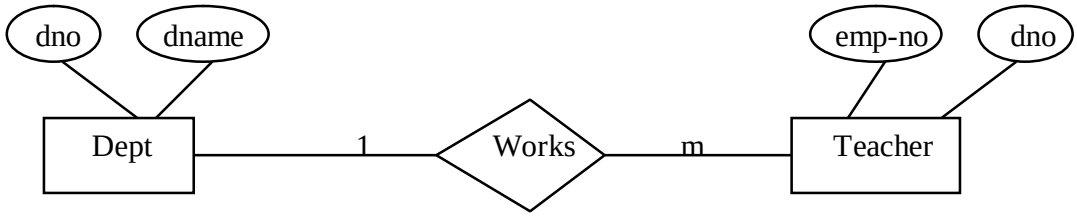
The schema design is as follows :

STUDENT (roll-no, sname, bdate, hno, dept)
Key : roll-no

COURSE (crs-no, cr-title, credit)
Key : crs-no

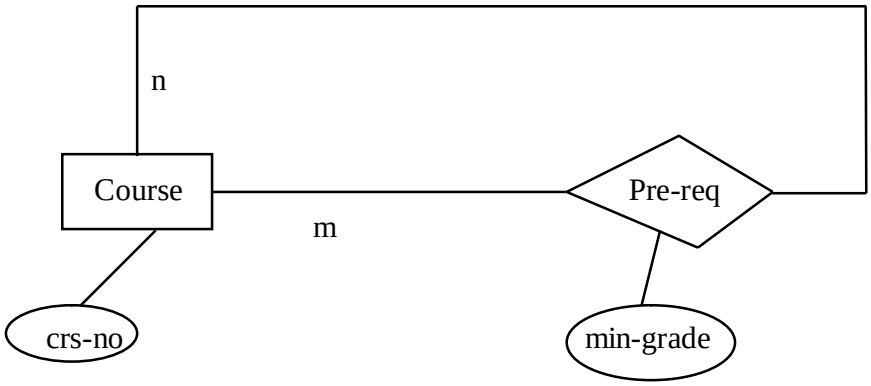
STUDY (roll-no, crs-no, grade)
Key : (roll-no, crs-no)

Consider the following diagram :



DEPT(dno, ….)
TEACHER (emp-no, …., dno)
dno is foreign key in TEACHER

Consider another diagram representing pre-requisite relationship :



COURSE (crs-no, crs-title, credit)
Key : crs-no

PREREQ (crs-no, pre-crs-no, min-grade)
Key : (crs-no, pre-crs-no)

Rules for Extended E-R model

EER model provides generalization and specialization (both are ISA type relationships), creating class/subclass hierarchies

- if entity E ISA (E1, E2, …..) , E possesses common attributes, and E1, E2 , ….. may have additional attributes; key is defined for E and is inherited by others

The generalization/specialization relationship can be represented in many ways :

- create relation for E as R_E(K, a1, a2, …)
  create relation for each Ei as R_Ei (K, attr(Ei))

- create relation for each Ei with inherited attributes
  R_Ei (K, a1, a2, ….., attr(Ei))

- create a single relation R which includes attributes of E, E1, E2, …. and type attribute T to indicate which Ei :
  R (K, a1, a2, …., attr (E1), attr (E2), …, T)
  (tuples may contain many null values)