Unit -2

Data Modeling using the Entity-Relationship(ER) model:

Using High-Level Conceptual Data Models for Database Design, A sample Database Application, Entity types, Entity Sets, Attributes, and Keys, Relationship Types, Relationship Sets, Roles and Structural Constraints, Weak Entity types, Refining the ER Design, ER Diagrams, Naming Conventions and Design Issues, Relationship Types of Degree Higher than two, Relational Database Design using ER-to-Relational Mapping.

Text Book:

1. Database systems Models, Languages, Design and Application Programming, RamezElmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.

2. Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill

Prepared by,
Dr.Aruna M G
Associate Professor
Department of AI & ML
DSCE
Bangalore

Entity-Relationship Model

Using High-Level Conceptual Data Models for Database Design or Database Design Process

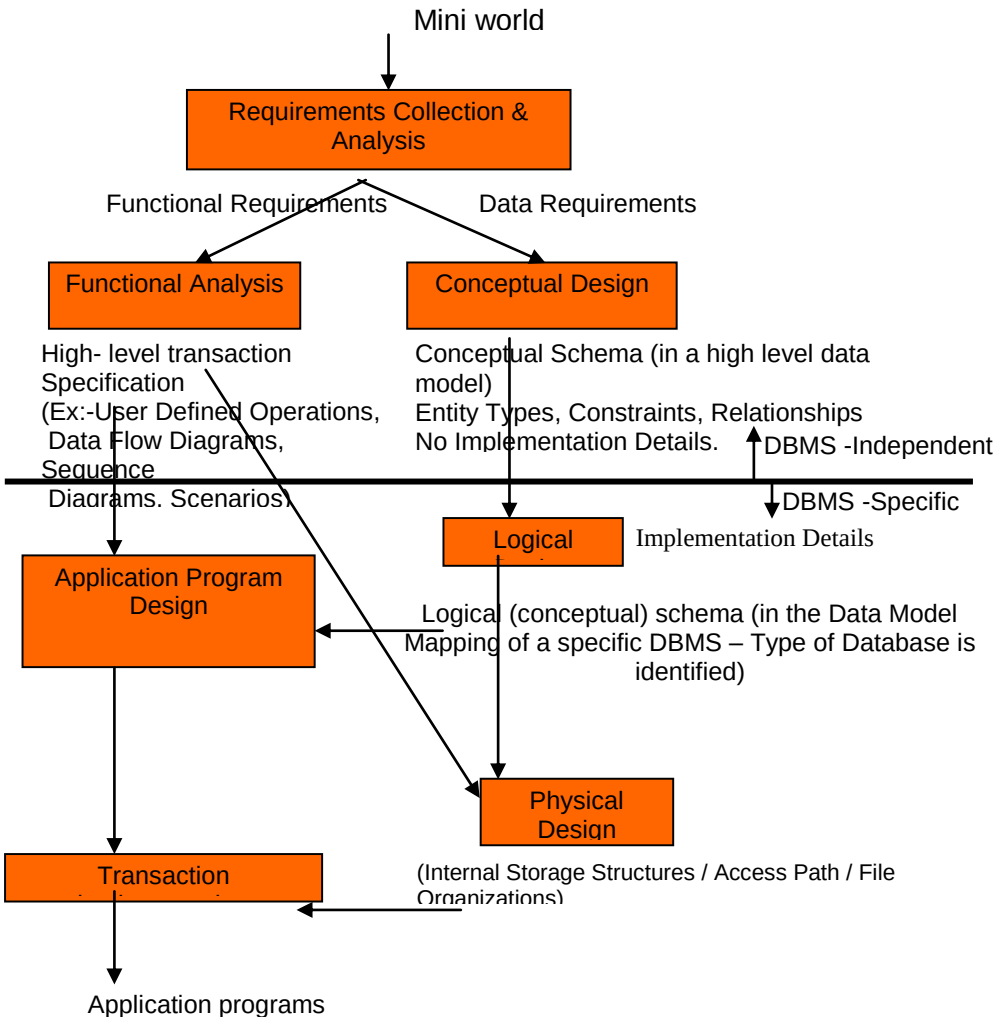


Figure2.1 A simplified diagram of database design phases

Figure shows a simplified description of the Database design process using high level conceptual Data Model

The first phase is **requirement collection and analysis**. During this step, the DB designer is responsible of DB users to understand and document their data requirement. The result of this is written in set of users requirements. The requirement should be specified in detailed and complete a form as possible.

In parallel with requirement, it is also necessary to specify **functional requirements of the application**. These consist the user-defined operation (transaction) that will apply to DB;

they include both retrievals and updates. Functional requirement is specified by using data flow diagrams, sequence diagrams, scenarios and other techniques.

Once all the requirement have been collected and analyzed.

The next step is to create a **conceptual schema** for the DB, using a high-level conceptual data model. This step is called **conceptual DB design**. This schema will describe the structures of the whole DB for community of users. It hides the details of physical storage structures and includes detailed descriptions of data types, relationships and constraints; these are expressed using the concepts provided by the high-level data model. These concepts do not include implementation details; they are easier to understand and can be used to communicate with nontechnical users.

High level conceptual schema is used as a reference, to ensure all user's data requirements are met and that the requirements do not conflicts. This provides the DB designer to concentrate on specifying the properties of the data without concerning with storage details.

After the conceptual schema has been designed, the basic data model operation can be used to specify high level user operations identified during **functional analysis**.

The next step in DB design is actual implementation of the DB, using a commercial DBMS.

Conceptual schema is transformed from the high level data model into the implementation data model. This step is called **logical DB design or data model mapping.**

The last step is the **physical DB design** phase, during which the internal storage structures, indexes, access paths, and file organization for the Db are specified.

In parallel with these activities, application programs are designed and implemented as DB transactions corresponding to the high level transaction specifications.

An example Database Application:

- Requirements of the Company (oversimplified for illustrative purposes)

–The company is organized into DEPARTMENTs. Each department has a unique name, unique number and an employee who manages the department. To keep track of the start date of the employee began to managing department. A department may have several locations.

–Each department controls a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.

–To store each EMPLOYEE's social security number, address, salary, sex, and birthdate. Each employee works for one department but may work on several projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the direct supervisor of each employee.

–Each employee may have a number of DEPENDENTs. For each dependent, to keep track of their name, sex, birth date, and relationship to employee.

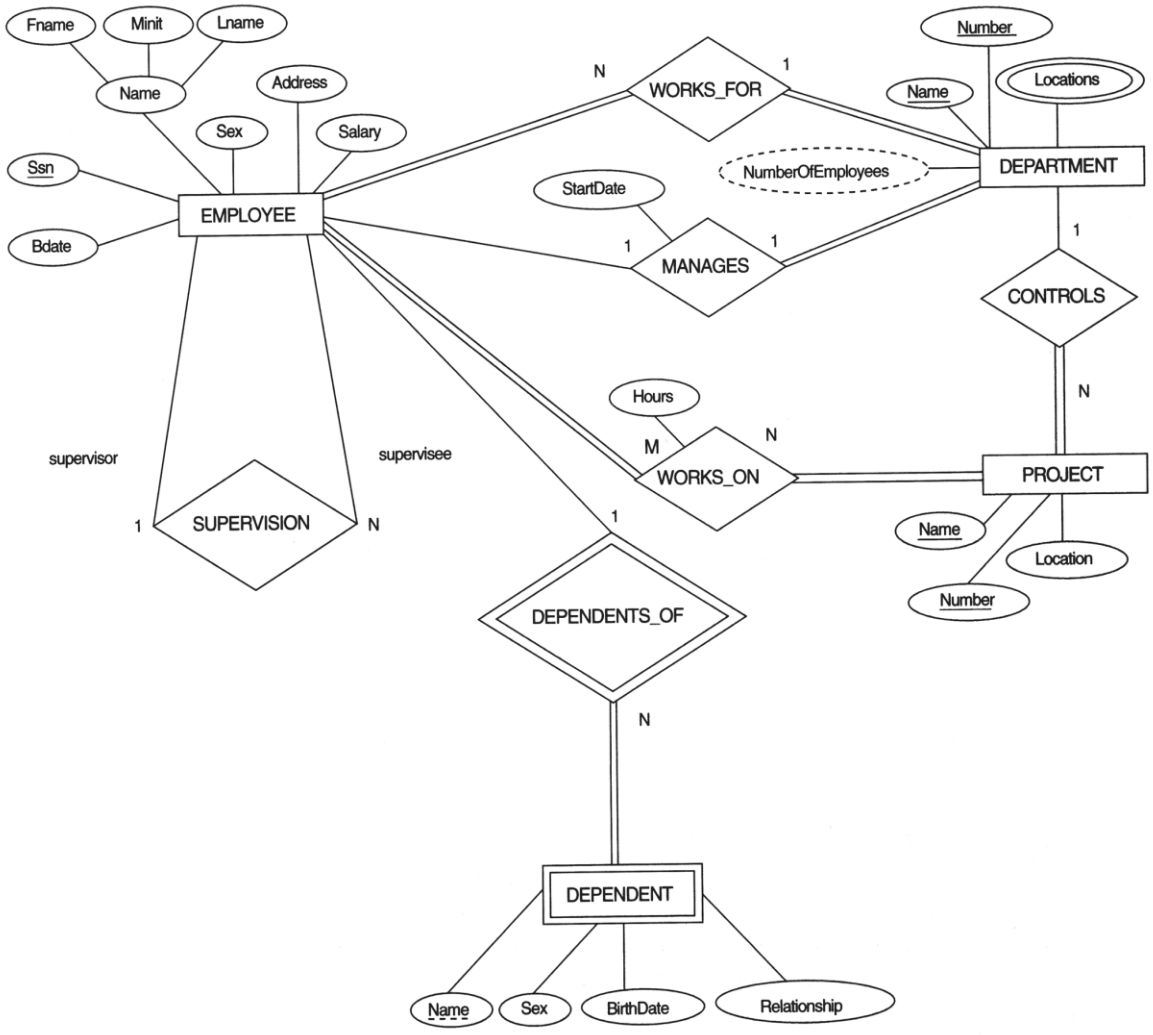
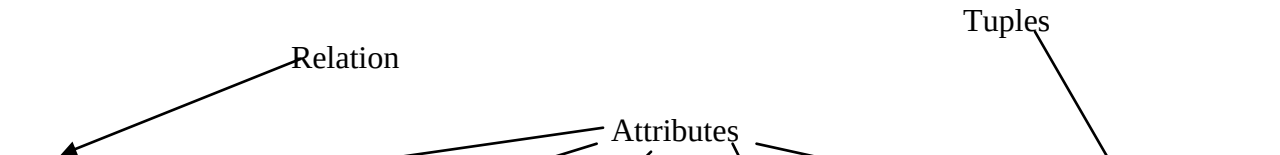


Figure2.2 An ER schema diagram for the COMPANY database

Relation

A relational database is made up of a collection of tables or relation.

Common terms	DBMS	RDBMS
Database	Table	Database
Table	Table	Relation
Column	Field	Attribute
Row	record	tuple



EMPLOYEE

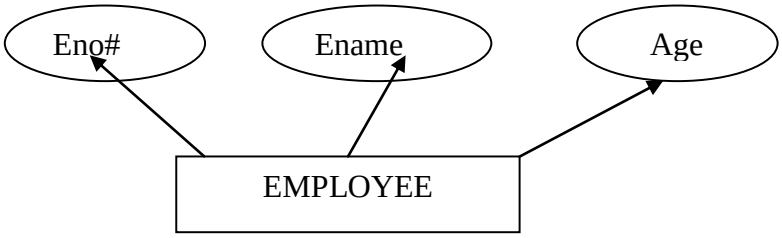
SSN	**Name**	**Age**	**Address**	**Salary**
123467	Aruna	25	#45,Bangalore	20000
123469	Thunga	26	#35,hassan	12000
123468	Nimi	27	#23,Bangalore	15000
123567	Sur	28	Null	20000

Figure2.3Employee Relation

ER Model (Entity Relationship Model)
The ER model describes data as entities, relationships, and attributes.

Entities
–Entities are specific objects or things in the mini-world that are represented in the database.
Or
- An entity is a real world object with a physical existence.
Example: a particular person, car, house, Employee etc.
Represented in ER diagram by rectangle

Attributes
Attributes are particular properties or characteristics used to describe an entity.
For example an EMPLOYEE entity may have a Name, SSN, Address, Sex, BirthDate etc.
–Each attribute has a value set (or data type) associated with it – e.g. integer, string, subrange, enumerated type,
Represented in ER diagram by oval



Attribute Type:
It is the property of entity type instance is called attribute type

Attribute Instance:
It is the property of entity instance is called attribute instance.

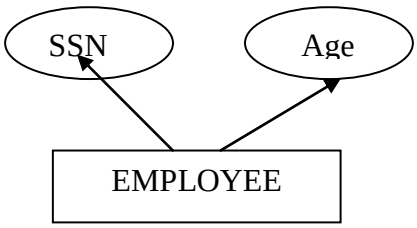
–A specific entity will have a value for each of its attributes
For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55‘

Types of Attributes

1.Simple Vs composite attribute

Simple or Atomic Attribute
–Each entity has a single atomic value for the attribute i.e means the attribute cannot be divided into simpler components further.
Or
The attribute that is not divisible into subparts.

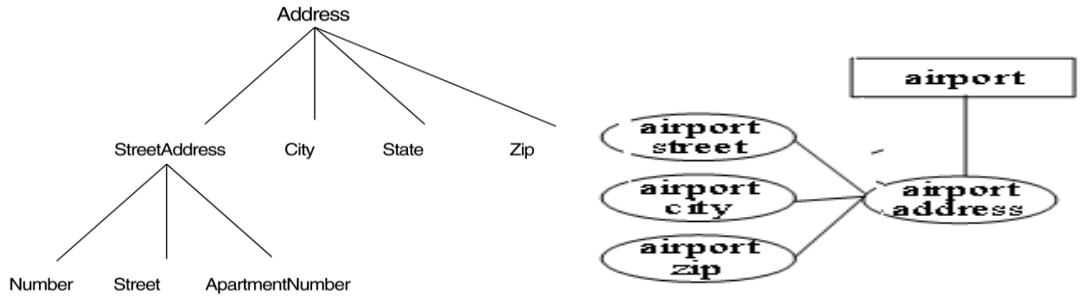
For example: SSN, Sex, Age of an employee



Composite
–The attribute may be composed of several components which mean attribute can be further divided into smaller subparts.
Or
The attribute that can be divided into smaller subparts.

- Composite attribute is denoted by **parenthesis ( )**

For example: Address (Apt#, House#, Street, City, State, ZipCode, Country) ,
Name (FirstName, MiddleName, LastName)
Date of joining of the employee.
» Can be split into day, month and year



2.Single Vs Multi-valued Attributes

Multi-valued

An entity may have multiple values for that attribute.

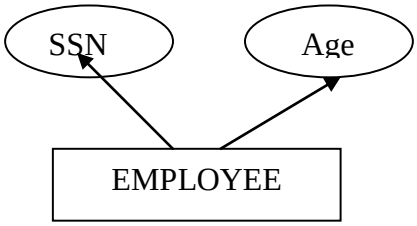
- Multi valued attribute is denoted by **flower braces { }** and ER diagram by **double oval**.

For example, Color of a CAR or PreviousDegrees of a STUDENT. Denoted as {Color} or {PreviousDegrees}.

Single-valued

An entity has a single value for the attribute.

For example, Age , SSN, Sex



3 Stored Vs Derived attribute

Stored

The attribute values cannot be derived from related entities. Or The Attribute cannot be derived from any other attribute.

Example: Birth date, Date of joining attribute of employee.

Derived

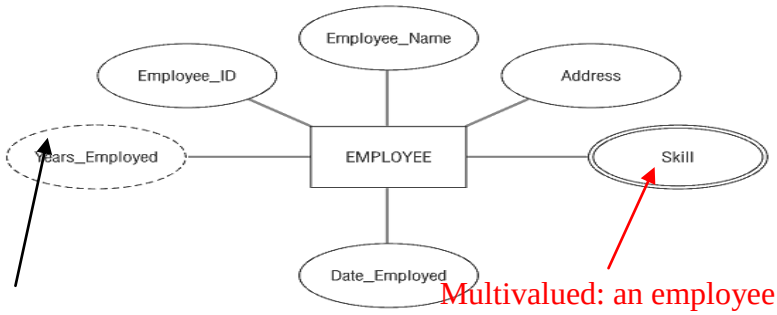
Attribute that can be calculated based on other attributes.

Example: years of service of employee can be calculated from date of joining and current date.

Or

The attribute values can be derived from related entities.

Example: Age attribute. If we know the birthdate it is easy to find age of person.



Derived
From date employed and current date

Multivalued: an employee can have more than one skill

4 Complex

The composite and multi-valued attributes may be nested arbitrarily to any number of levels although this is rare.

Or

The combination of composite and multivalued attribute is also called complex attribute.

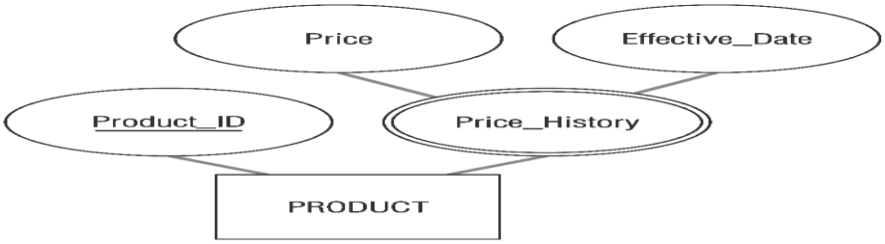
Composite attribute between denoted by parentheses ( ) and multivalued attributes between braces { }.

Example1: PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}.

Example2:

{AddressPhone({Phone(AreaCode,PhoneNumber)},
Address(StreetAddress(Number,Street,ApartmentNumber),
City,State,Zip)) }

Example3: **an attribute that is both multivalued and composite**



Null Attirbute

An entity may not have any applicable value for an attribute. A special value is used in such situation called NULL.

Example: PhoneNo or Degree attribute of EMPLOYEE entity.

The unknown category of Null is classified into two cases:

- The first case arises when it is know that the attribute value exists but is missing.
- The second case, when it is not known whether the attribute value exist or not.

Entity Types

An Entity type defines a collection (or set) of entities that have the same attributes. Each entity in the DB is described by a entity name and list of attributes.

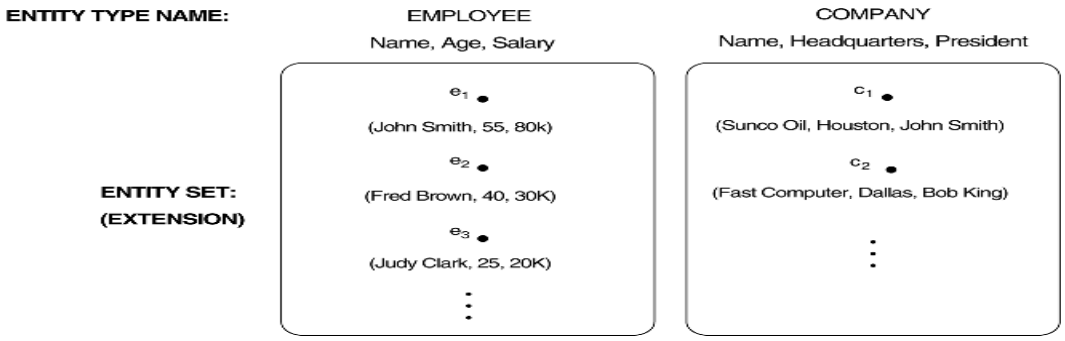
For example, the EMPLOYEE entity type/entity name or the PROJECT entity type.

Entity Set

A collection of all entities of a particular entity type.

For example: "All people having an account at a bank".

ENTITY SET corresponding to the ENTITY TYPE CAR



Key Attributes

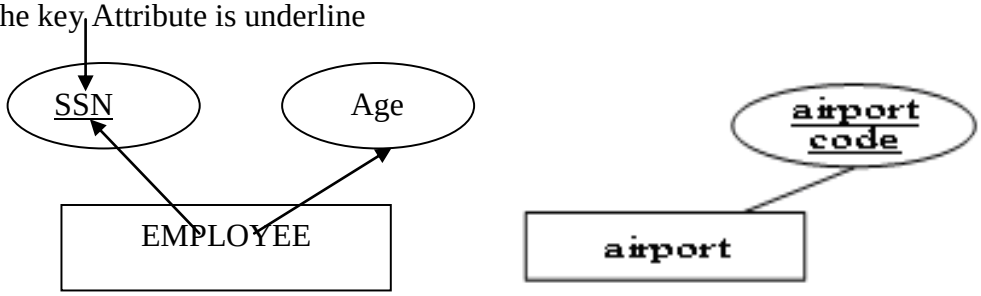
An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type.

Or

An entity type has an attribute whose values can be used to identify each entity uniquely is called a key attribute.

For example: SSN of EMPLOYEE. Airport _code in AIRPORT

The key Attribute is underline



Super Key

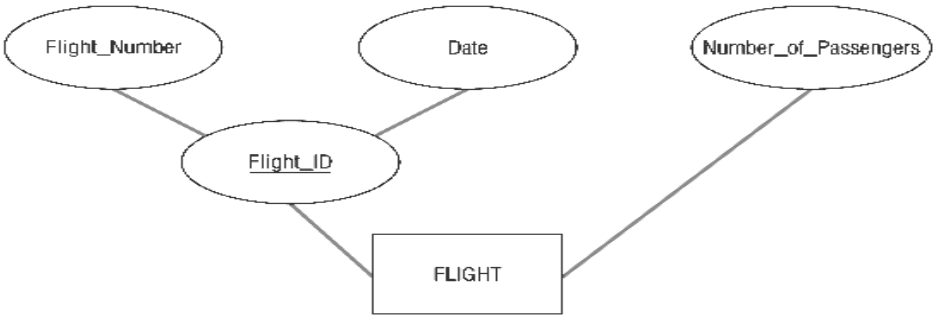
- The set of attributes that can uniquely identify a tuple is known as Super Key. For Example, STUD_NO, (STUD_NO, STUD_NAME),
- A super key is a group of single or multiple keys that identifies rows in a table.
- It supports NULL values.
- Adding zero or more attributes to the candidate key generates the super key.
- A candidate key is a super key but vice versa is not true.
- Super Key values may also be NULL.

Composite Key

A key attribute may be composite. Such attribute is composite key.

For example, VehicleTagNumber is a key of the CAR entity type with components (Number, State).

Example2: Flight table Flight_ID is composite key

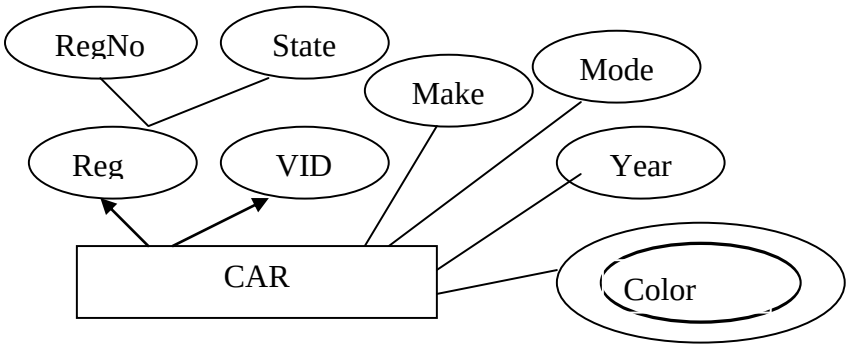


Candidate key

An entity type may have more than one key called candidate key.
Or
The minimal set of attributes that can uniquely identify a tuple is known as a candidate key

For example, the CAR entity type may have two keys:
–VehicleIdentificationNumber (popularly called VIN) and
–VehicleTagNumber (Number, State), also known as license_plate number.

Example: The CAR entity type with two key attributes, Registration and VehicleID
CAR
Registration(RegistrationNumber, State), VehicleID, Make, Model, Year, (Color)
car1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 1999, (red, black))
car2
((ABC 123, NEW YORK), WP9872, Nissan 300ZX, 2-door, 2002, (blue))
car3
((VSY 720, TEXAS), TD729, Buick LeSabre, 4-door, 2003, (white, blue))

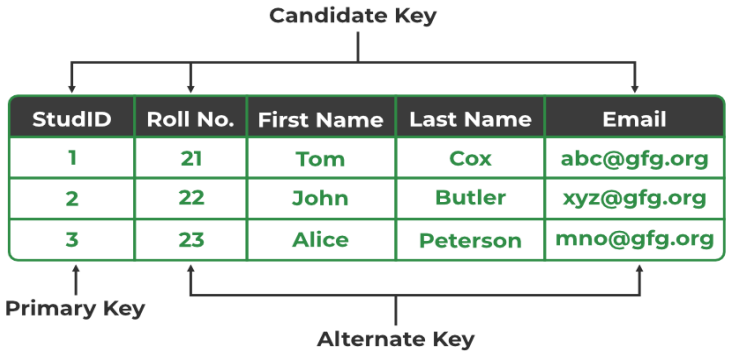


STUD_NO, as well as STUD_PHONE, are candidate keys for relation STUDENT but STUD_NO can be chosen as the primary key (only one out of many candidate keys).
STUDENT table -> Student(STUD_NO, SNAME,
ADDRESS, PHONE) , STUD_NO is a primary key

Table STUDENT

STUD_NO	**SNAME**	**ADDRESS**	**PHONE**

1	**Shyam**	**Delhi**	**123456789**
2	**Rakesh**	**Kolkata**	**223365796**
3	**Suraj**	**Delhi**	**175468965**



Keys in RDBMS

- In EasyShop retail system, how is uniqueness maintained among customers?
- How do you establish the link/association between two relations?

Candidate key

- A *minimal* set of columns/attributes that can be used to *uniquely* identify a single row/tuple in a given relation, is called candidate key
- Identified during **design time**

Primary key

- Database designer selects one of the candidate keys as primary key for the purpose of identification of a row of table uniquely
- Implented during table creation

Foreign key

- A foreign key is a column or combination of columns that is used to establish and enforce a link between the data in two relations
- Implented during table creation

Value set (or Domain) of attributes

The domain of an attribute is the set of all possible values from which the attributes can take its values.

Value sets are not displayed in ER diagrams.It is specified during creating entity type.
Example: Season attribute have the possible values are Spring, Summer, Autumn, Winter.
Gender attribute can have Female or Male.

Mathematically Definition of value sets of attribute: An attribute A of entity type E whose vale set is V can be defined as a function from E to the power set P(V) of V;
A : E→ P(V)

Initial Conceptual Design of the Company

DEPARTMENT
Name,Number,{Location},Manager,ManagerStartDate

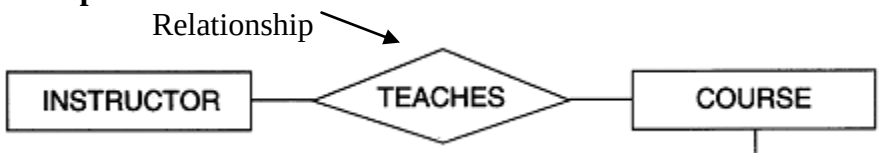
PROJECT
Name,Number,Location,contrllingDepartment

EMPLOYEE
Name(Fname,Initial,Lname),Eno,Sex,Address,Salary,DOB,Department,Supervisor, {Workon(Project,Hours)}

DEPENDENT
Employee,DependentName,Sex,DOB,Relationship

Relationship
An association among entities is called relationship.
In ER diagram relationship are displayed as **diamond shape boxes** which are connected by straight line to the rectangular box representing entity types.

Example:



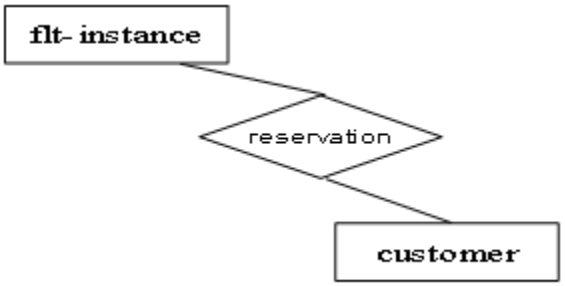
Relationship Type

A relationship type between two entity types defines the set of all association between these entity types. A relationship relates two or more distinct entities with a specific meaning.

or

A relationship type R among n entity types $E_1, E_2, \ldots\ldots, E_n$ defines a set of associations.
Example: EMPLOYEE John Smith works on the ProductX PROJECT
or EMPLOYEE Franklin Wong manages the Research DEPARTMENT.
$\{ (e_1, \ldots, e_n) \mid e_1 \square E_1, \ldots\ldots, e_n \square E_n \}$

Example:



Mathematically, the relationship set R is a set of relationship instance r_i , where each r_i associates n individual entities ($e_{1,....,}$ e_n) and each entity e_i in r_i is a member of entity type E_j ,1<=j<=n.

Where R be the relationship type. E_1,E_2,….E_n be the entity type. e_1 through e_n, are entity set and r_i is the relationship instances.

Relationship set

An instance of a relationship set is a set of relationship.

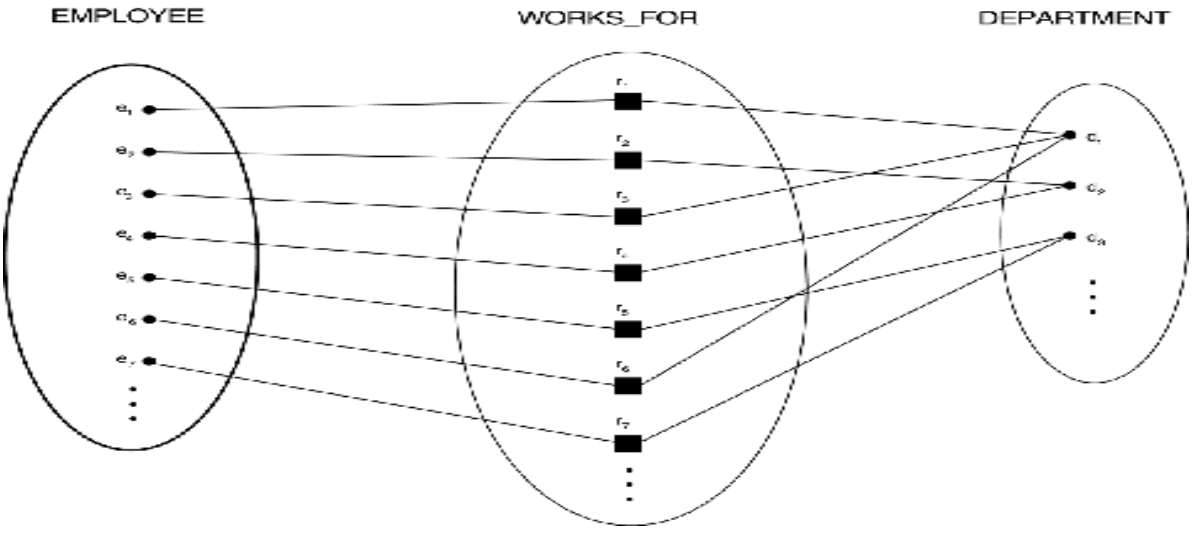
Relationship instance

Each instance of the relationship between members of these entity types is called a relationship instance.

r_i = ($e_{1,....,}$ e_n)

Participate Relationship Type:

Each of the entity type E_1,E_2,….E_n is said to Participate in the relationship type R.



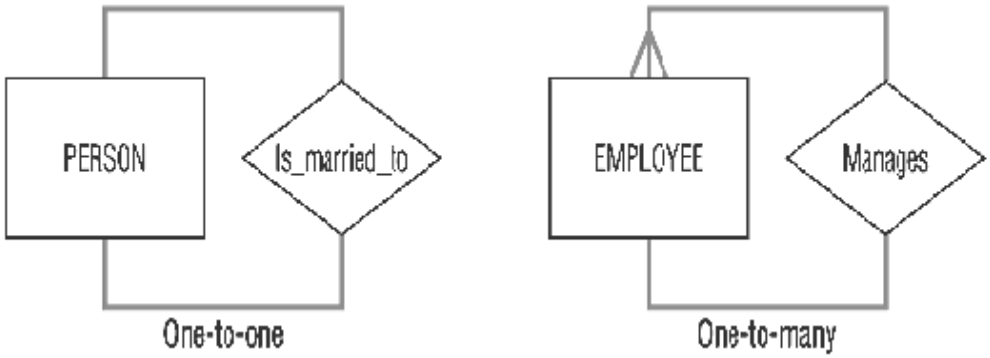
Degree of a relationship types

- The degree of a relationship type is the number of participating entity types.

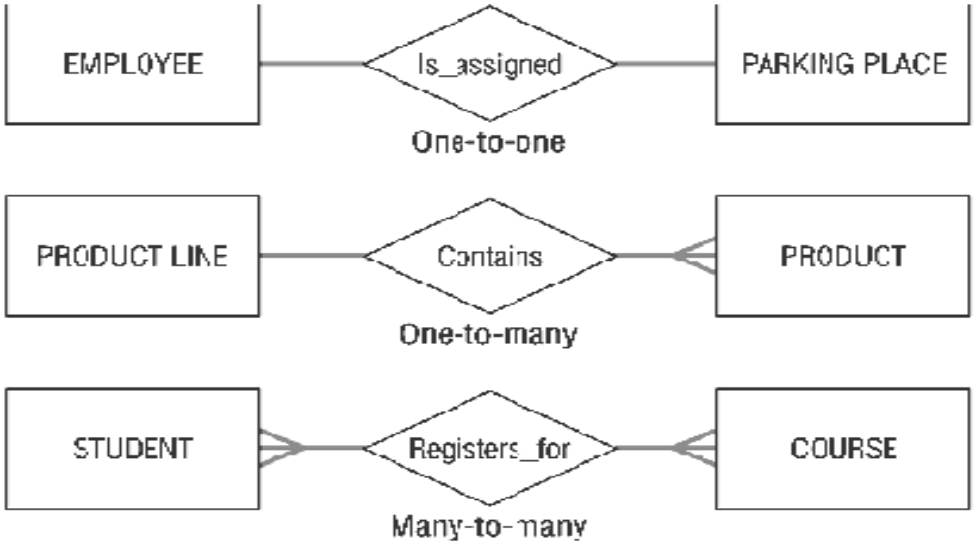
Both MANAGES and WORKS_ON are binary relationships.

- **Unary Relationship** : The number of participating entity type is one in relationship type called unary relationship.
  Example1-SUPERVISION

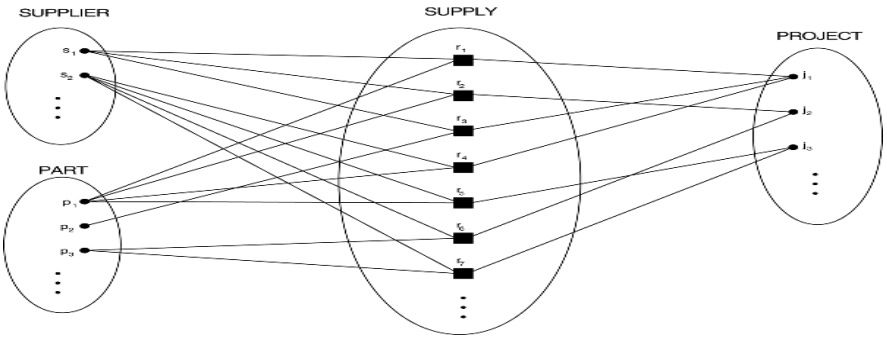
Examples 2:



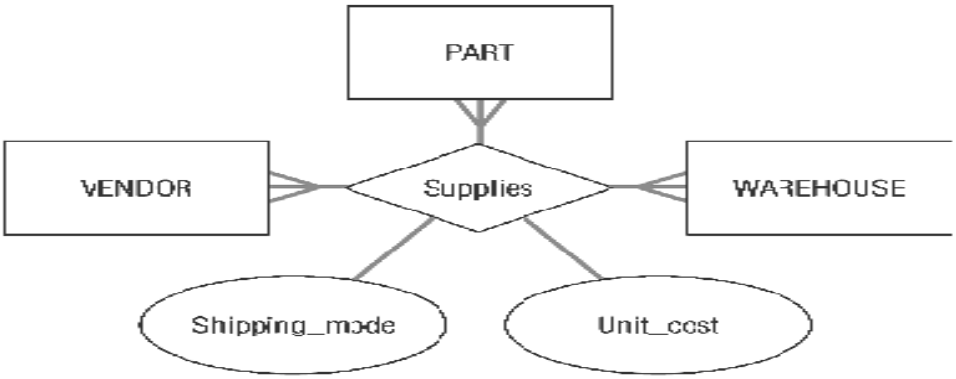
- **Binary Relationship**: The number of participating entity type is two in relationship type called binary relationship.
  Example1- MANAGES ,CONTROLS , WORKSON AND WORKS-FOR .
  Example2:



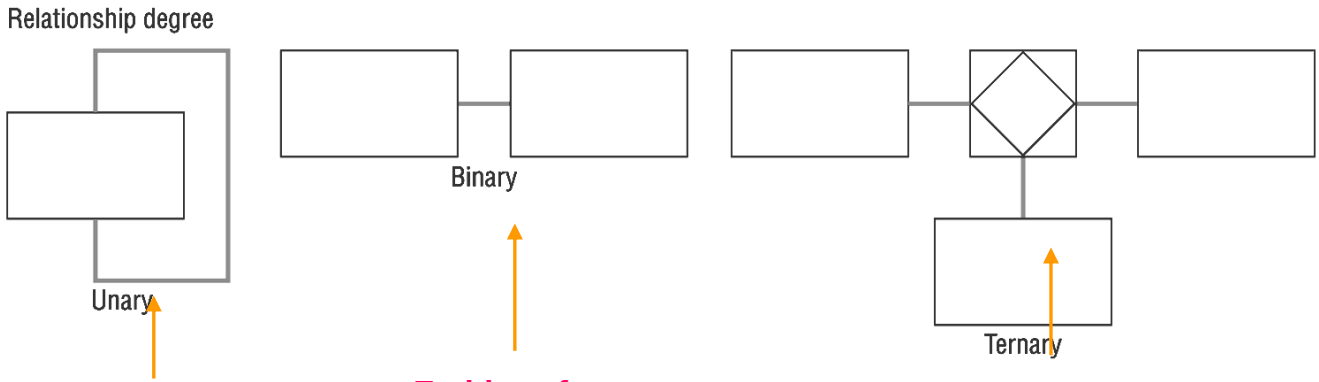
- **Ternary Relationship**; The number of participating entity type is three in relationship type called ternary relationship.



Example-SUPPLIER



Degree of relationships –



One entity related to another of the same entity type

Entities of two different types related to each other

Entities of three different types related to each other

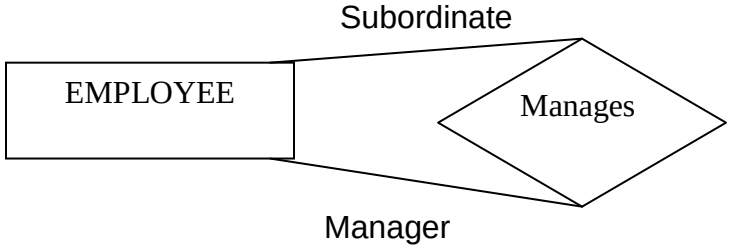
Role names

Each entity type that participates in a relationship type plays a particular role in the relationship.

Role names may be added to make the meaning more explicit.
Ex: WORKFOR relationship type, Employee plays the role of employee or worker and DEPARTMENT plays the role of department or employer.

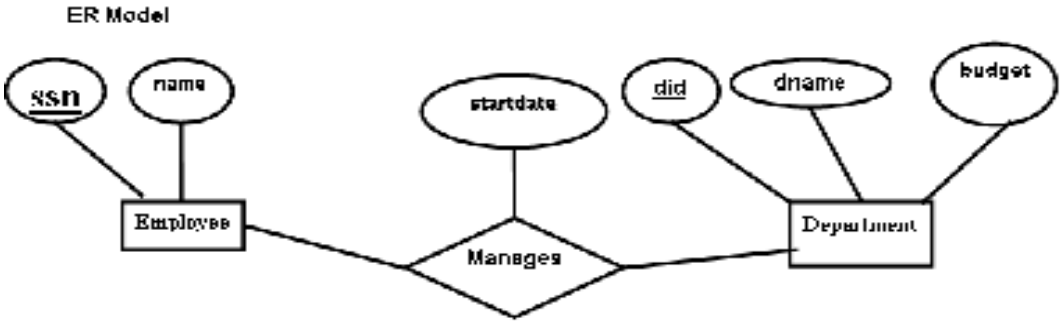
When to use role name?
Roles names are not technically necessary in r/n type where all the participating entity types are distinct, since each participating entity type name can be used as the role name. But in some cases the same entity type participates more than once in an r/n type in different roles. In such cases the role name becomes essential for distinguishing the meaning of each participation. Such r/n types are called recursive relationship.

Recursive relationship
The same entity participates more than once in a relationship type in different roles.
Ex: MANAGES relationship type, employee plays the role of subordinate and manager.

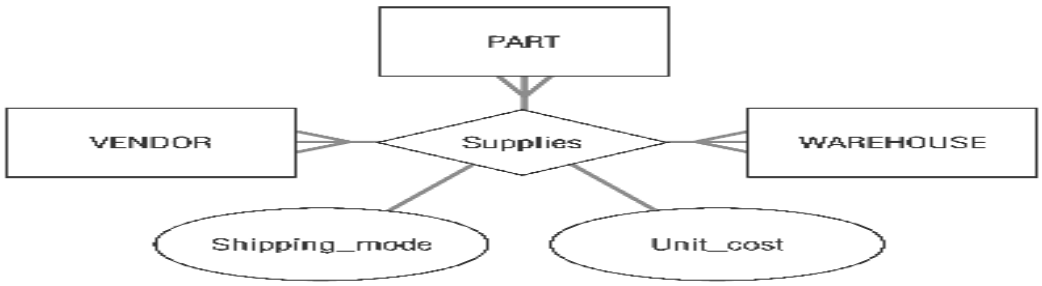


A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

Attributes of relationship types
Relationship types can also have attribute similar to those of entity type.
Example1: To record the number of hours per week that an employee works on a project, we include an attribute hours for the WORKSON r/n type as shown in figure2.2.
Example2:To record the startdate of employee who manages the department is placed on MANAGES r/n type.



Example3:



Note: Placing the attribute in relationship type i.e 1:1, 1:N and M:M . Representing this attribute in one of the entity type.

- The attributes of 1:1 or 1:N relationship(or r/n) types can be migrated to one of the participating entity types.
  1:1 r/n type

Ex: StartDate attribute for the MANAGES r/n can be an attribute of either EMPLOYEE or DEPARTMENT, although conceptually it belongs to MANAGES. This is because MANAGES is 1:1 r/n, so every department or employee entity participates in ***at most one*** r/n instance. Hence, the value of the Startdate can be determined separately, either by the participating department entity or by the participating employee(manager) entity.

2) 1:N r/n type, a r/n attribute can be migrated only to the entity type on the N-side of the r/n.
Ex: if workfor r/n also ha an attribute startdate that indicates when an employee started working for a department, this attribute can be included as an attribute of employee. *This is because each employee works for only one department, and hence participates in* ***at most one*** *r/n instance in workfor.*

Placing the attribute is determined by schema designer.

3) M:M r/n types, some attributes any be determined by the combination of participating entities in a r/n instance, not by any single entity. Such attributes must be specified as r/n attributes.
EX: Hours attribute of the M:M WORKON r/n type, the number of hours the employee works on project is determined by an (separate/new relation as) employee_project combination and not separately by either entity.

Constraints on Relationships Types (constraints means restriction or limitation)
Relationship types have certain constraints that limit the possible combination of entities that may participate in the corresponding relationship set.
Example: a company has a rule that each employee must work for exactly one department.

There two main types of relationship constraints

- Cardinality ratio (Maximum Cardinality constraint)

- Participation (also called Minimum Cardinality or participation constraint or existence dependency constraints)

Constraints ratio for binary Relationship Types

(Also known as ratio constraints)

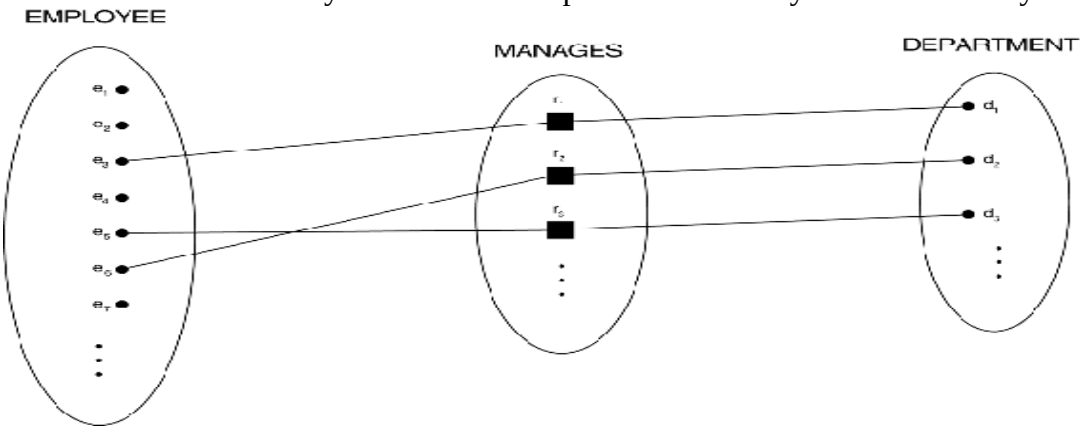
Cardinality ratio (Maximum Cardinality) for a binary relationship specifies the maximum number of relationship instances that an entity can participate in.

Or

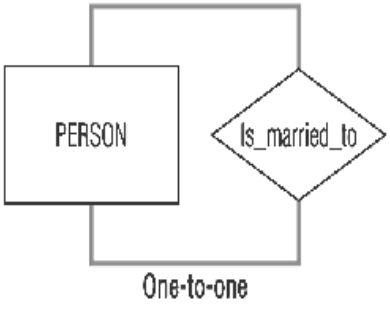
Cardinality Constraints - the number of instances of one entity that can or must be associated with each instance of another entity.

- One-to-one (1:1)
- One-to-many (1:N)
- Many-to-one (N:1)
- Many-to-many (M:N)

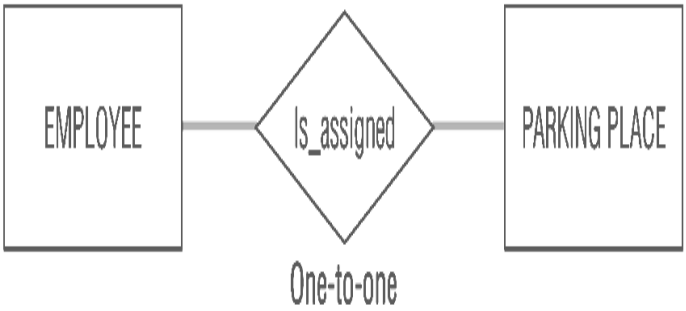
- **One – to – One**
  - Each entity in the relationship will have exactly one related entity.



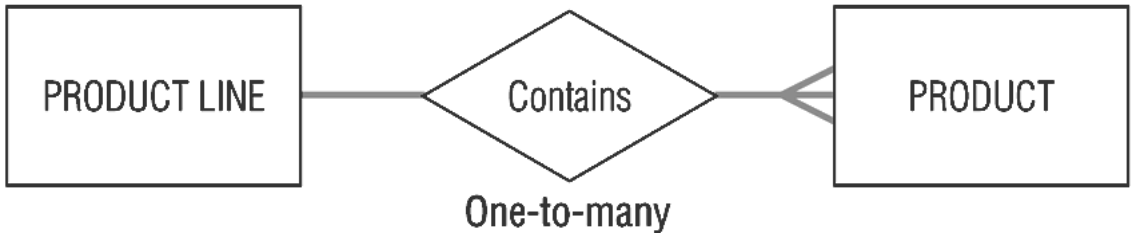
Example2:



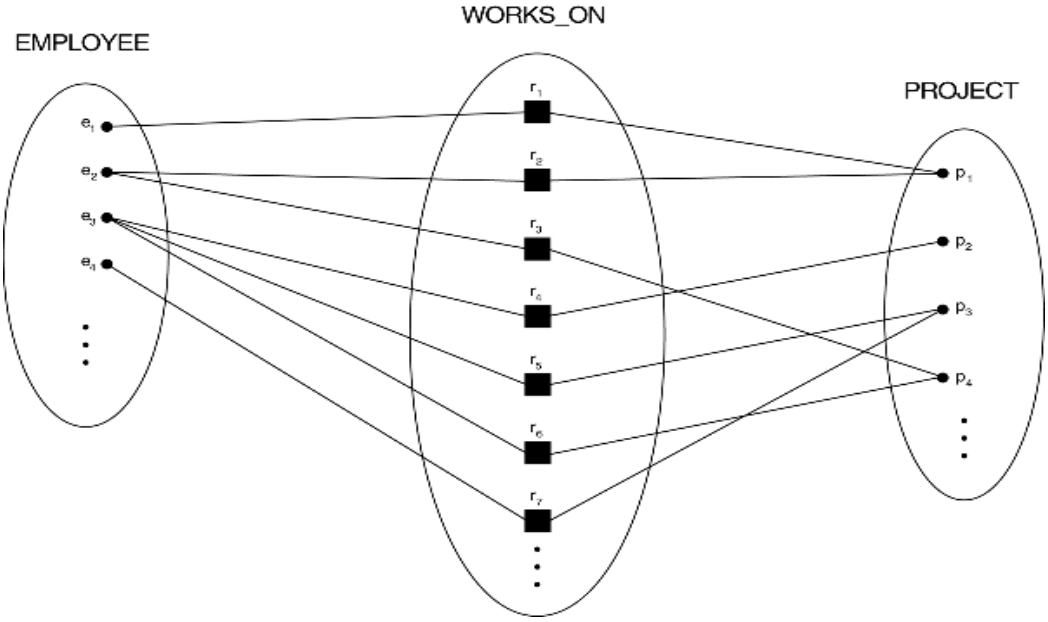
Example3:

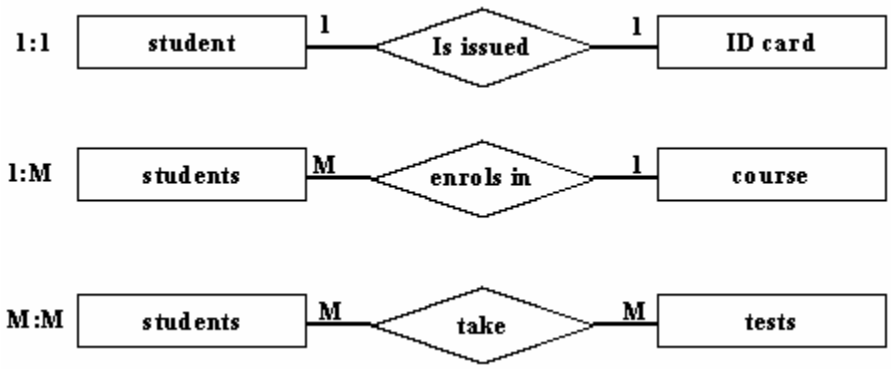


- **One – to – Many**
  - An entity on one side of the relationship can have many related entities, but an entity on the other side will have a maximum of one related entity.



- **Many – to – Many**
  - Entities on both sides of the relationship can have many related entities on the other side.





Participation constraint

It specifies whether the existence of an entity depends on its being related to another entity the relationship types.

This constraint specifies the minimum number of relationship instances that each entity can participate in. This type is called **Minimum Cardinality constraint**.

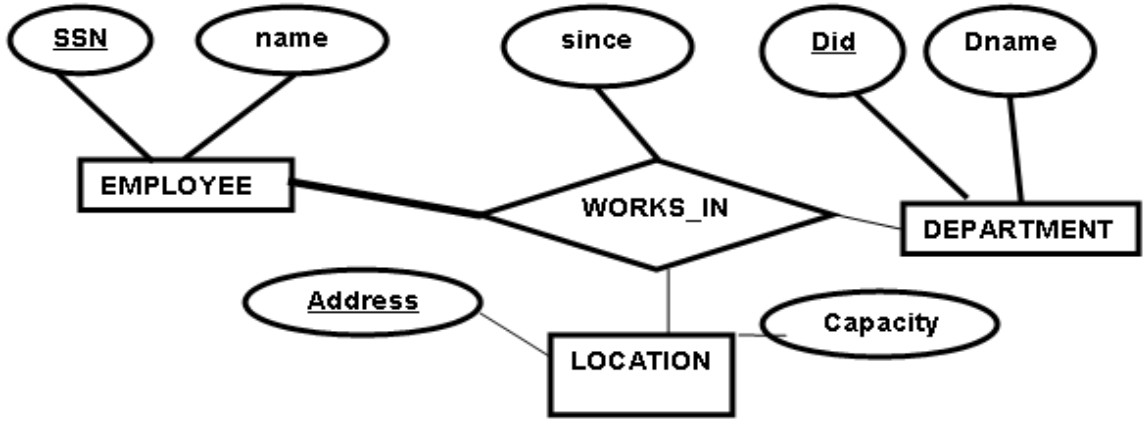
There are two types of Participation constraint

1.Total Participation constraint (or existence dependency):

It means that every entity participates in 'the total set "of the relationship types

Represented in **double line** in ER diagram.

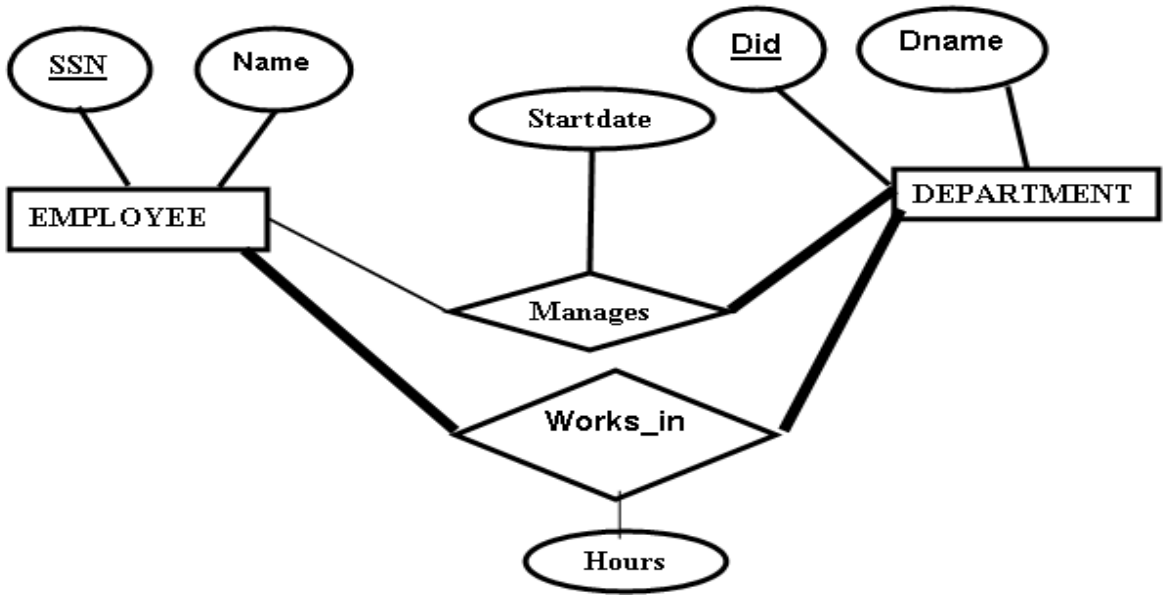
Ex: WORKS_IN thick line representing total participation



2.Partial Participation constraint:

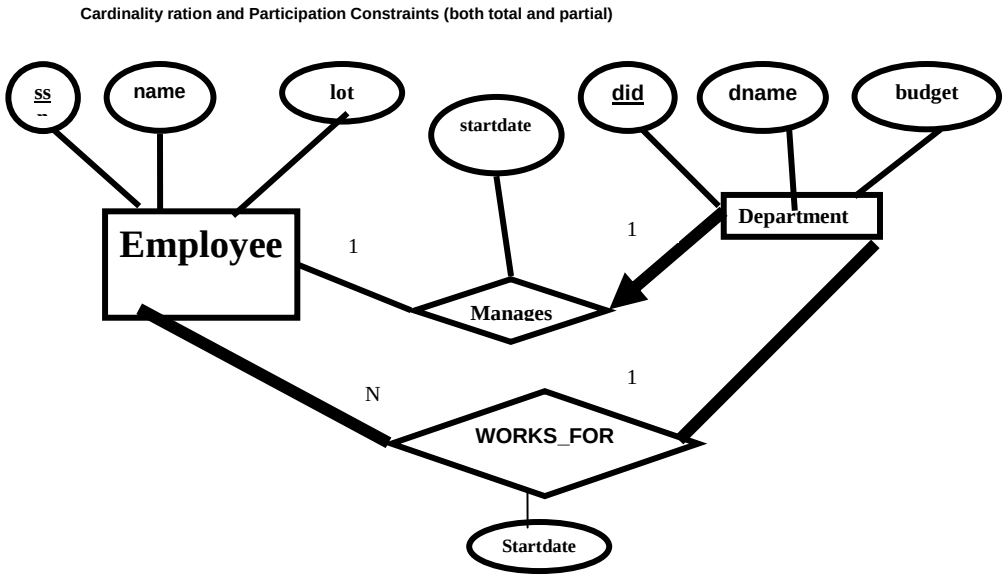
It means that some or "part of the set of" entity participates in a relationship types. Represented in **single line** in ER diagram.

Ex: MANAGES thin line representing partial participation.



Structural constraints

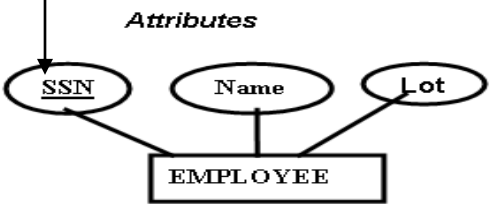
The combination of cardinality ratio and participation constraints taken together is called a **Structural constraints of a r/n type.**



Regular entity or Strong entities ***or identifying or owner entity type or parent entity type or dominant entity type***

Entity that has its own key attribute.
It exist independently of other types of entities
Eg: Emloyee, Student etc

KEY ATTRIBUTE



Weak Entity Types o*r child entity type or subordinate entity type*

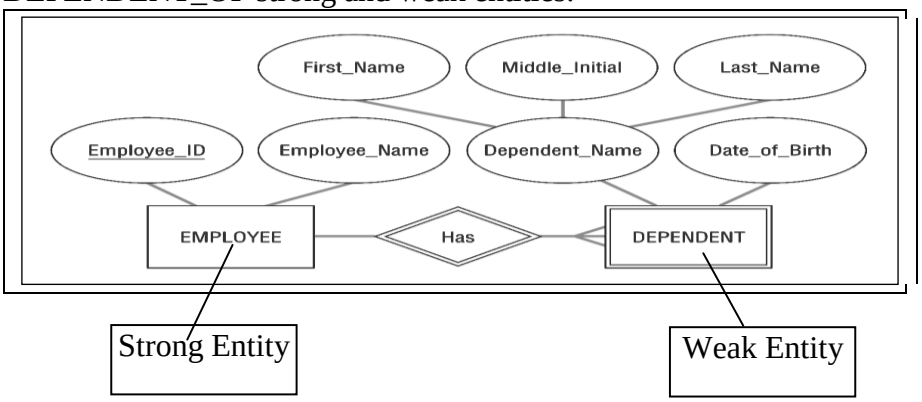
- An entity that depends on other entity for its existence and doesn't have a key attribute of its own.

Or
Weak entity type dependent on a strong entity type and cannot exist on its own.
Does not have a unique identifier.

- A weak entity must participate in an identifying relationship type with an owner or identifying entity type.
- Entities are identified by the combination of:
  - A partial key of the weak entity type.
  - The particular entity they are related to in the identifying entity type.

Example:

Suppose that a DEPENDENT entity is identified by the dependent's first name and birthdates, and the specific EMPLOYEE that the dependent is related to. DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type DEPENDENT_OF strong and weak entities.



Identifying relationship

Identifying relationship links strong entities to weak entities.

Why weak entity type is always total participation constraint?

Because weak entity type cannot be identified without an owner entity type.

Partial key or discriminator:

A weak entity type has partial key, which is the set of attributes that can uniquely identify weak entity type that are related to the same owner entity type.

Note:

- a composite attribute of all the weak entity attributes will be the partial key.
- weak entity type can have more than one identifying entity type and an identifying relationship type .
- weak entity type can be complex attribute (composite, multivalued) sometimes.

PROPER NAMING OF SCHEMA CONSTRUCTS

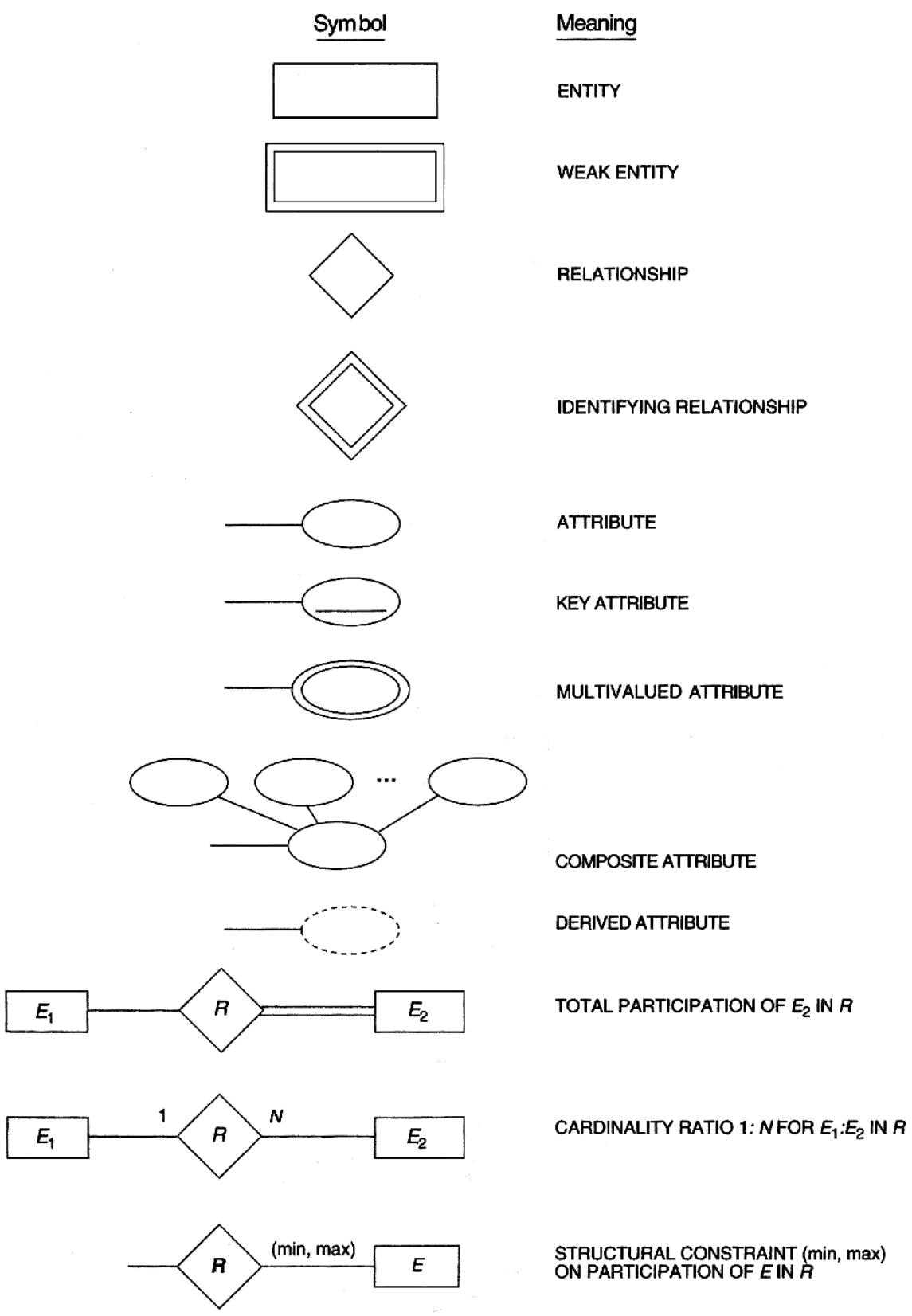
- Singular names for entity types.
- Entity type and r/n type names are in uppercase letter.
- Attributes names are capitalized.
- Role name are in lower case letter.
- ER diagram of the schema readable from left to right and from top to bottom.

- In general practice, nouns tend to indicates names of entity types and the verbs ten d to indicate names of relationship types. Attributes names generally from additional nouns that describe the nouns corresponding to entity types.

Ex: DEPENDENTS_OF r/n types reads from bottom-to-top. To change this to read from top to bottom, rename the r/n type to HAS-DEPENDENTs. These read as follows an employee entity(top entity type) has-dependents(r/n type) of type dependent (bottom entity type).

Notations for ER diagram

A graphical represents the entities, attributes and relationship.


Symbol
Meaning
ENTITY
WEAK ENTITY
RELATIONSHIP
IDENTIFYING RELATIONSHIP
ATTRIBUTE
KEY ATTRIBUTE
MULTIVALUED ATTRIBUTE
COMPOSITE ATTRIBUTE
DERIVED ATTRIBUTE
E1
R
E2
TOTAL PARTICIPATION OF E2 IN R
1
N
CARDINALITY RATIO 1: N FOR E1:E2 IN R
(min, max)
STRUCTURAL CONSTRAINT (min, max) ON PARTICIPATION OF E IN R

Alternative (min, max) notation for relationship structural constraints:

One alternative ER notation for specifying structural constraints on r/n. This involves associating a pair of integer numbers (min,max) with each participation of an entity type E in a r/n type R, where 0<=min<=max and max>=1.

The numbers mean that for each entity e in E, e must participate in **at least min** and **at most max** r/n instances in R at any point in time.

Min=0 implies partial participation, whereas min>0 implies total participation.

Default (no constraint): min=0, max=n.

- Must have min≤max, min≥0, max≥1.
- Derived from the mini-world constraints.

Examples:

(a) A department has *exactly one* manager and an employee can manage *at most one* department.

- Specify (1,1) for participation of EMPLOYEE in MANAGES
- Specify (0,1) for participation of EMPLOYEE in MANAGES

(b) An employee can work for *exactly one* department but a department can have *any number of* employees.

- Specify (1,1) for participation of EMPLOYEE in WORKS_FOR
- Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

ER-to-Relational Mapping Algorithm

Step 1: Mapping of Regular Entity Types
Step 2: Mapping of Weak Entity Types
Step 3: Mapping of Binary 1:1 Relation Types
Step 4: Mapping of Binary 1:N Relationship Types.
Step 5: Mapping of Binary M:N Relationship Types.
Step 6: Mapping of Multivalued attributes.
Step 7: Mapping of N-ary Relationship Types.

ER-to-Relational Mapping Algorithm

- *Step 1: Mapping of Regular Entity Types.*
  - For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
  - Choose one of the key attributes of E as the primary key for R. If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

Example: We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram. SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown.

Step 2: Mapping of Weak Entity Types

- For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R.
- In addition, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

Example: Create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT. Include the primary key SSN of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to ESSN).

The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT.

Step 3: Mapping of Binary 1:1 Relation Types

For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. There are three possible approaches:

(1) Foreign Key approach: Choose one of the relations-S, say-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S.

Example: 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.

(2) Merged relation option: An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.

(3) Cross-reference or relationship relation option: The third alternative is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

Step 4: Mapping of Binary 1:N Relationship Types.

- For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
  - Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
  - Include any simple attributes of the 1:N relation type as attributes of S.

Example: 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION in the figure. For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.

Step 5: Mapping of Binary M:N Relationship Types.

- For each regular binary M:N relationship type R, create a new relation S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.
  - Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

Example: The M:N relationship type WORKS_ON from the ER diagram is mapped by creating a relation WORKS_ON in the relational database schema. The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS_ON and renamed PNO and ESSN, respectively.

Attribute HOURS in WORKS_ON represents the HOURS attribute of the relation type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

Step 6: Mapping of Multivalued attributes

- For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.
  - The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

Example: The relation DEPT_LOCATIONS is created. The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key-represents the primary key of the DEPARTMENT relation. The primary key of R is the combination of {DNUMBER, DLOCATION}.

Step 7: Mapping of N-ary Relationship Types

- For each n-ary relationship type R, where n>2, create a new relationship S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
  - Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.

Example: The relationship type SUPPY in the ER below. This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNO, PROJNAME}

FIGURE 4.11
Ternary relationship types. (a) The SUPPLY relationship.

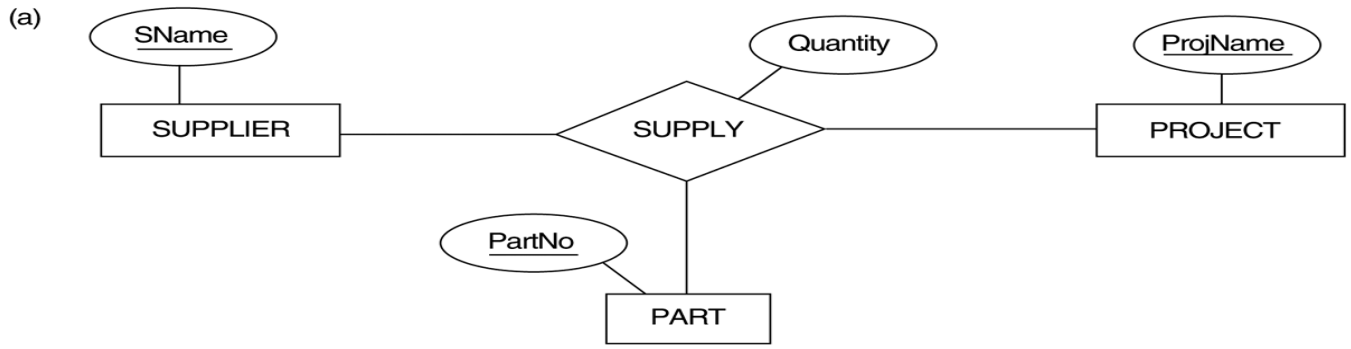


FIGURE 7.3
Mapping the *n*-ary relationship type SUPPLY from Figure 4.11a.

SUPPLIER

SNAME	. . .

PROJECT

PROJNAME	. . .

PART

PARTNO	. . .

SUPPLY

SNAME	PROJNAME	PARTNO	QUANTITY

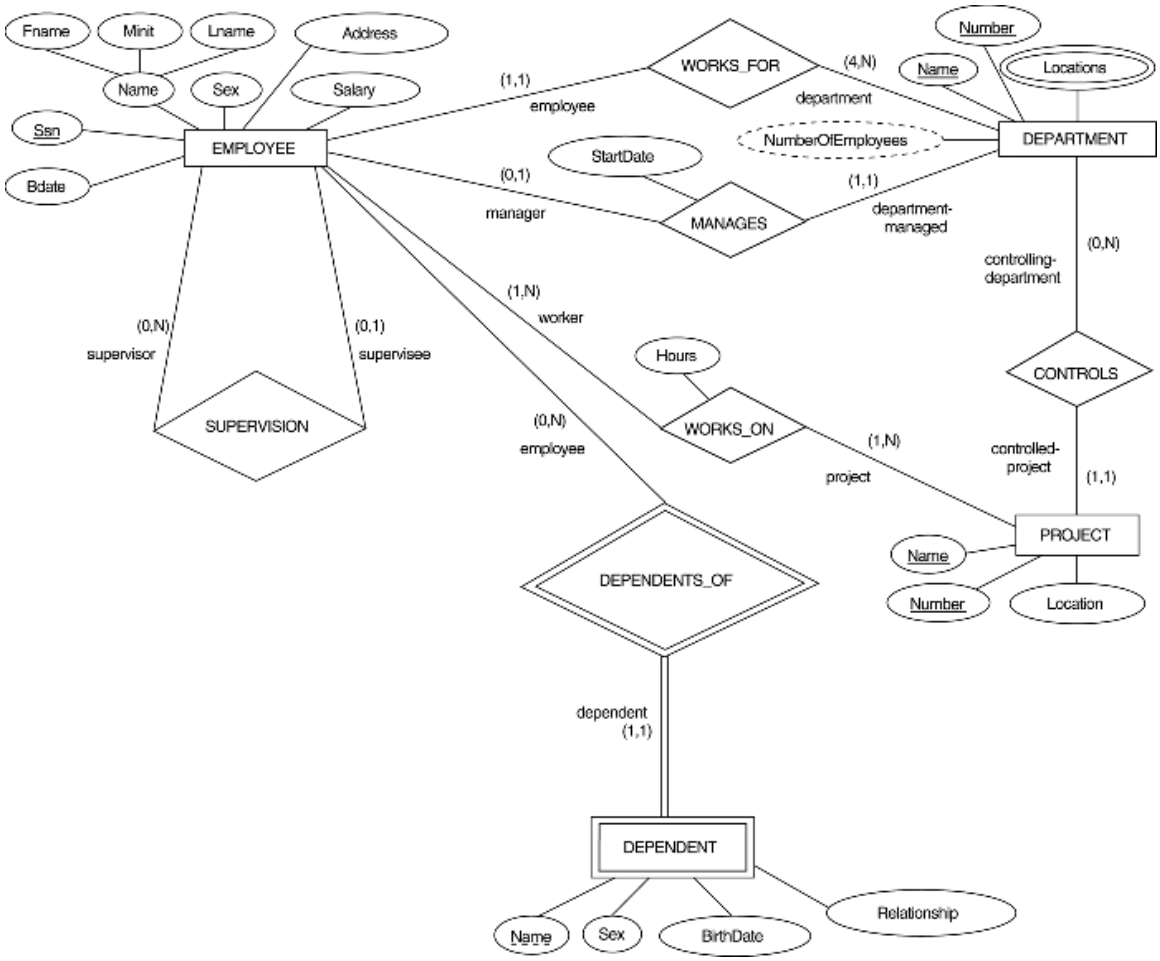
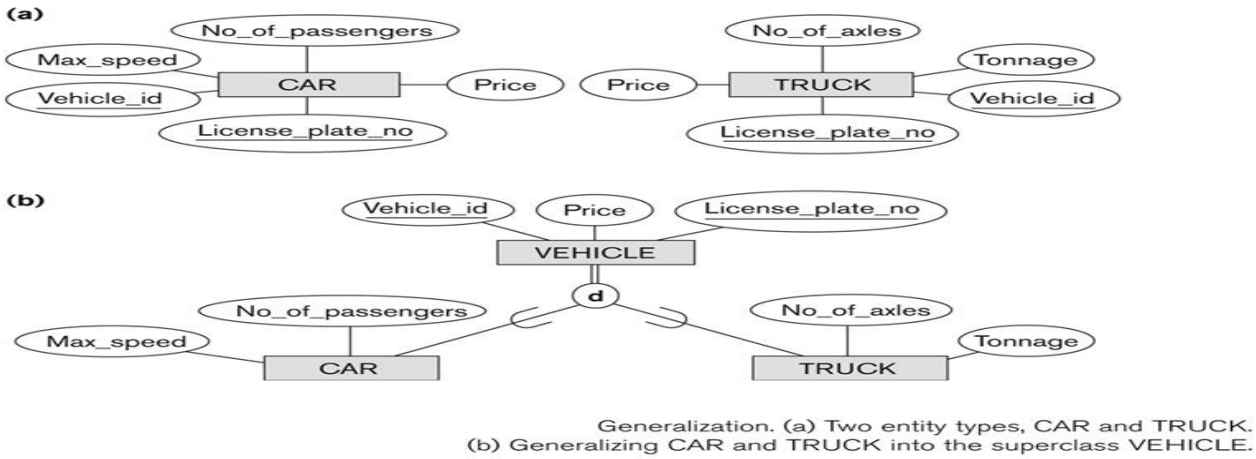


Figure:.ER diagram with structural constraints and all role names for the company DB schema.

Specialization and Generalization:

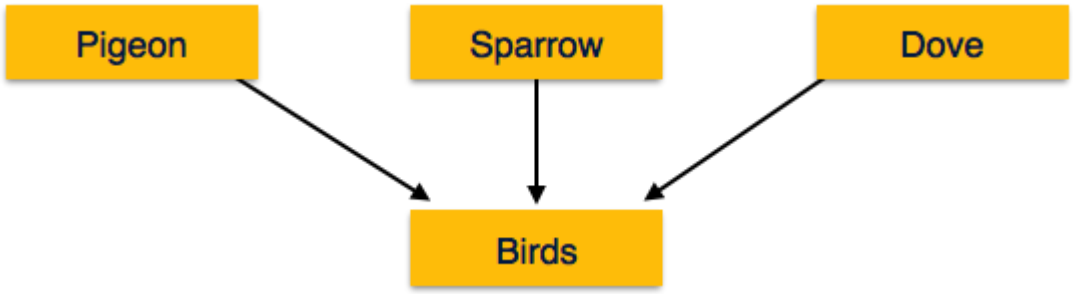
Generalization:

- Generalization is the reverse of the specialization process.
- Several classes with common features are generalized into a superclass;
  - original classes become its subclasses
- Example1: CAR, TRUCK generalized into VEHICLE;
  - both CAR, TRUCK become subclasses of the superclass VEHICLE.
  - We can view {CAR, TRUCK} as a specialization of VEHICLE
  - Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK.



Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

For example: pigeon, house sparrow, crow and dove can all be generalized as Birds.



Specialization:

Specialization is the process of defining a set of subclasses of a superclass.

The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass.

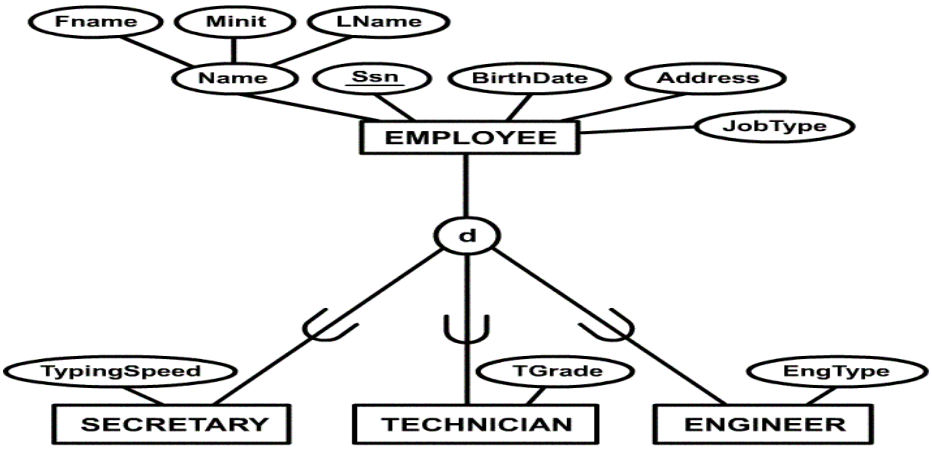
Or

Specialization is the opposite of generalization. In specialization, a group of entities is divided into sub-groups based on their characteristics.

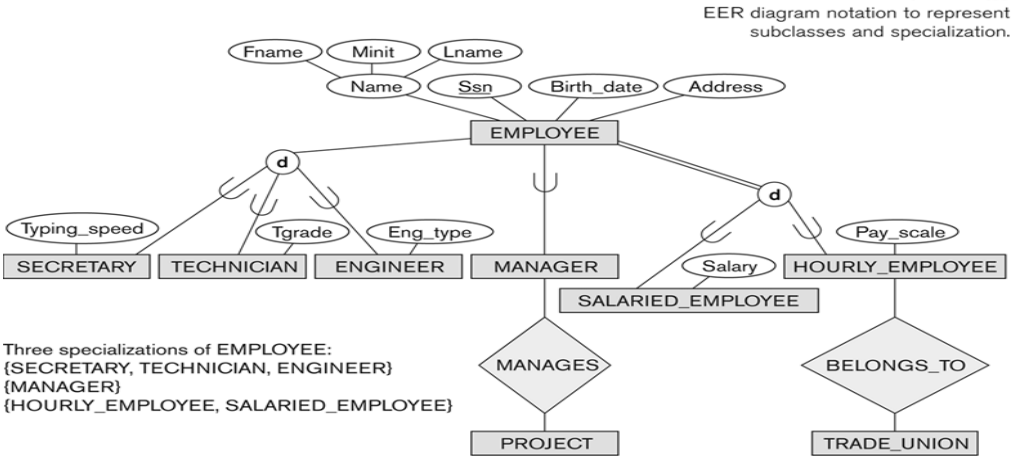
Take a group 'Person' for example. A person has name, date of birth, gender, etc. These properties are common in all persons, human beings.

But in a company, persons can be identified as employee, employer, customer, or vendor, based on what role they play in the company.

Example1: {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of EMPLOYEE based upon job type. May have several specializations of the same superclass.



Example2: Another specialization of EMPLOYEE based on method of pay is {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}.



Example3: Similarly, in a school database, persons can be specialized as teacher, student, or a staff, based on what role they play in school as entities.