

Unit 2

Relational Model And Entity-Relationship Model

-SUSHMA VANKHEDE

Contents

Relational data model: Structure of relational databases, Domains, Relations,

Relational algebra : fundamental operators and syntax, relational algebra queries.

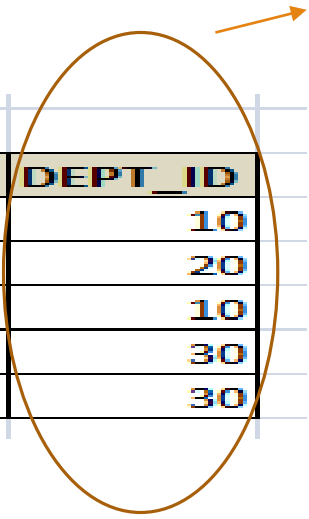
Entity- Relationship model: Basic concepts, Design process, constraints, Keys, Design issues, E-R diagrams, weak entity sets, extended E-R features – generalization, specialization, aggregation, reduction to E-R database schema

Relational data model

- It represents data and relationships among data by collection of tables.
- Each table has number of rows and columns.
- Columns are Attributes
- Rows are Records or tuple.

EMPLOYEE			
EMP_ID	EMP_NAME	ADDRESS	DEPT_ID
100	Joseph	Clinton Town	10
101	Rose	Fraser Town	20
102	Mathew	Lakeside Village	10
103	Stewart	Troy	30
104	William	Holland	30

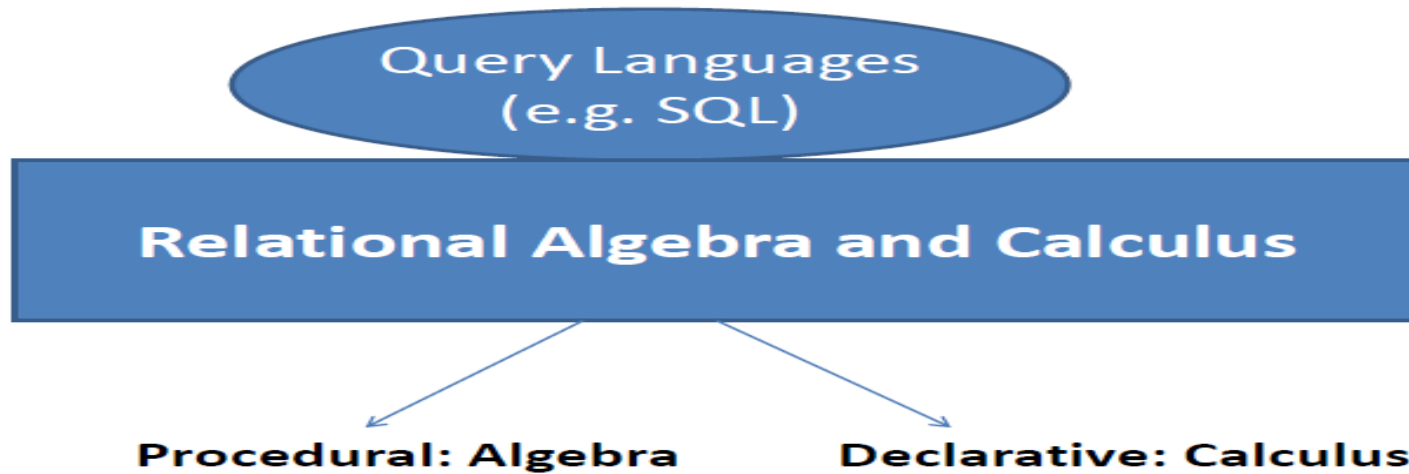
Domain: Set of
all permissible
values



Introduction to Relational Algebra

It is a Procedural language.

A collection of operations each acting on one or two relations and producing one relation as result



Introduction to Relational Algebra

The Fundamental Operations are:

- 1) Selection (σ)
- 2) Projection (Π)
- 3) Cartesian Product(X)
- 4) Union(U)
- 5) Set Difference(-)
- 6) Rename (ρ)

Introduction to Relational Algebra

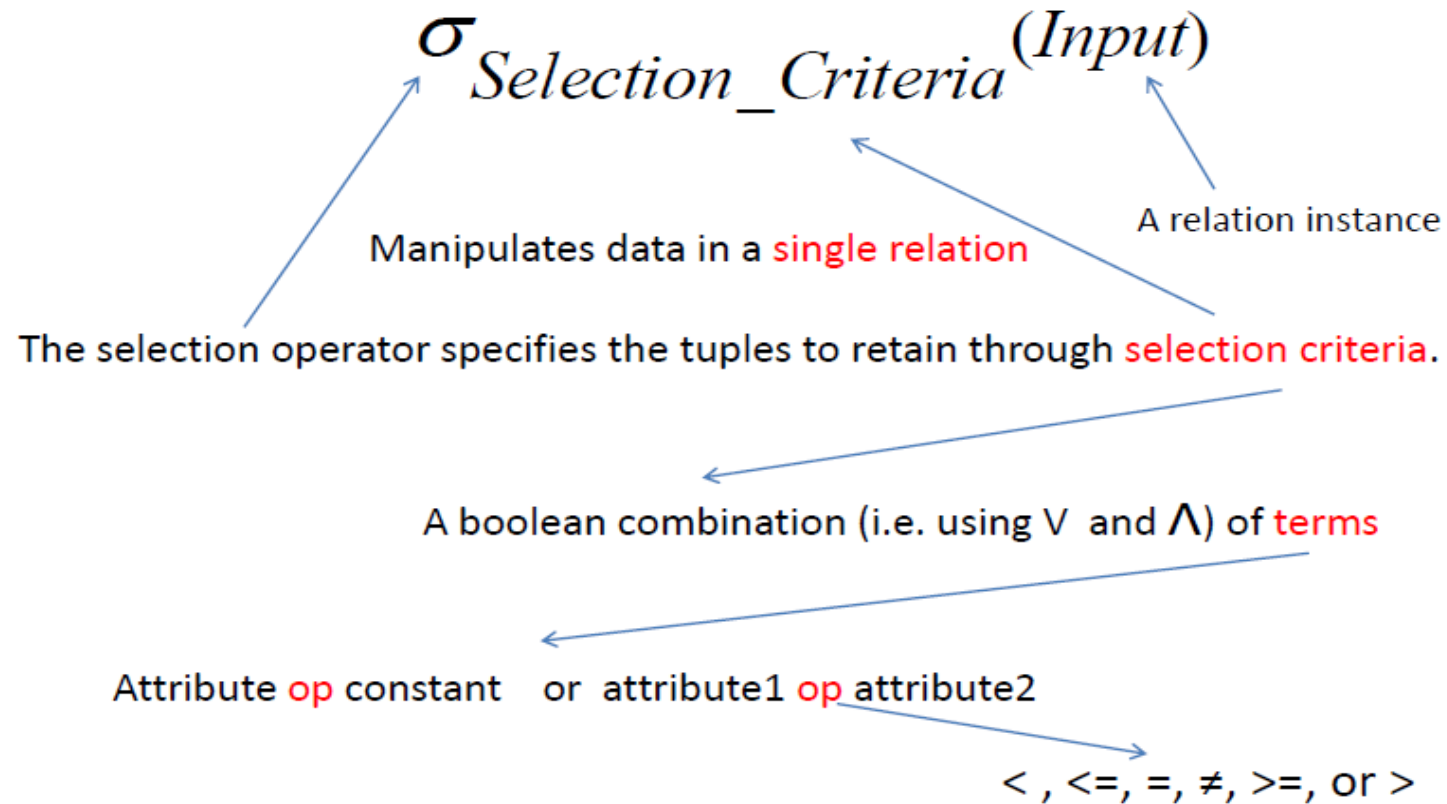
Unary Operations

- Operates on Single table or relation
- Unary Operators are : Selection, Projection and Rename

Binary Operations

- Operates on Minimum two or more relation.
- Binary Operators are : Cartesian Product, Union, Set Difference.


Selection Operation



Selection Operation

S2

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

 $\sigma_{rating > 8}(S2)$

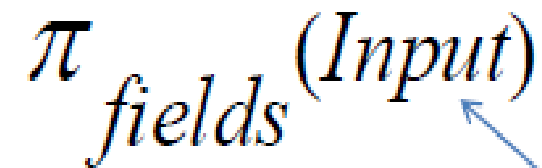


sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

Projection Operation

The Project Operation allow projecting the selected attributes from relations.

$\pi_{fields}(Input)$

The diagram shows the notation for the projection operation, $\pi_{fields}(Input)$. Two blue arrows originate from the text 'Allows us to extract columns from a relation' below. One arrow points to the Greek letter π , and the other points to the word 'Input' in parentheses.

Allows us to extract **columns** from a **relation**

Projection Operation

Example:

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0



$\pi_{age}(S2)$



age
35.0
55.5

Projection with Selection Operation

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0



$\pi_{sname, rating}(\sigma_{rating > 8}(S2))$



sname	rating
yuppy	9
rusty	10

Binary Operations

Takes as input two relation instances

Four standard operations

- Union
- Set-difference
- Cross-product

Union, set-difference require the two input set to be union compatible.

- They have the same number of fields
- corresponding fields, taken in order from left to right, have the same domains.

Union Operation

R U S returns relation instance containing all tuples that occur in either relation instance R or S, or both.

R and S must be union compatible.

Schema of the result is defined to be that of R.

Union Operation

S1

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

S1 U S2

Set-Difference

$R - S$: returns a **relation instance** containing all tuples that occur in R but not in S .

R and S must be union-compatible.

Scheme of the result is the schema of R .

Set-Difference

S1

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S1 – S2

sid	sname	rating	age
22	dustin	7	45.0

Cross-Product

$R \times S$: Returns a relation instance whose schema contains:

- All the fields of R (in the same order as they appear in R)
- All the fields of S (in the same order as they appear in S)

The result contains one tuple $\langle r, s \rangle$ for each pair with $r \in R$ and $s \in S$

Basically, it is the Cartesian product.

Fields of the same name are unnamed.

Cross-Product

S1

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

<u>sid</u>	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

S1 x R1

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Renaming

Name conflict can arise in some situations

It is convenient to be able to give names to the fields of a relation instance defined by a relational algebra expression.

$$\rho(R(\overline{F}), E)$$

- Take arbitrary relational expression E
- Returns an instance of a new relation R
- R is the result of E except that some fields are renamed
- **Renaming list** has the form (oldname \rightarrow newname or position \rightarrow newname)

Rename Operation

$$\rho(C(1 \rightarrow sid1, 5 \rightarrow sid2), S1 \times R1)$$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

sid1	sname	rating	age	sid2	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96



Entity Relationship Model

Overview of Database Design Process

Two main activities:

- Database design
- Applications design

Database design

- To design the conceptual schema for a database application

Applications design focuses on the programs and interfaces that access the database

- Generally considered part of software engineering

Overview of Database Design Process

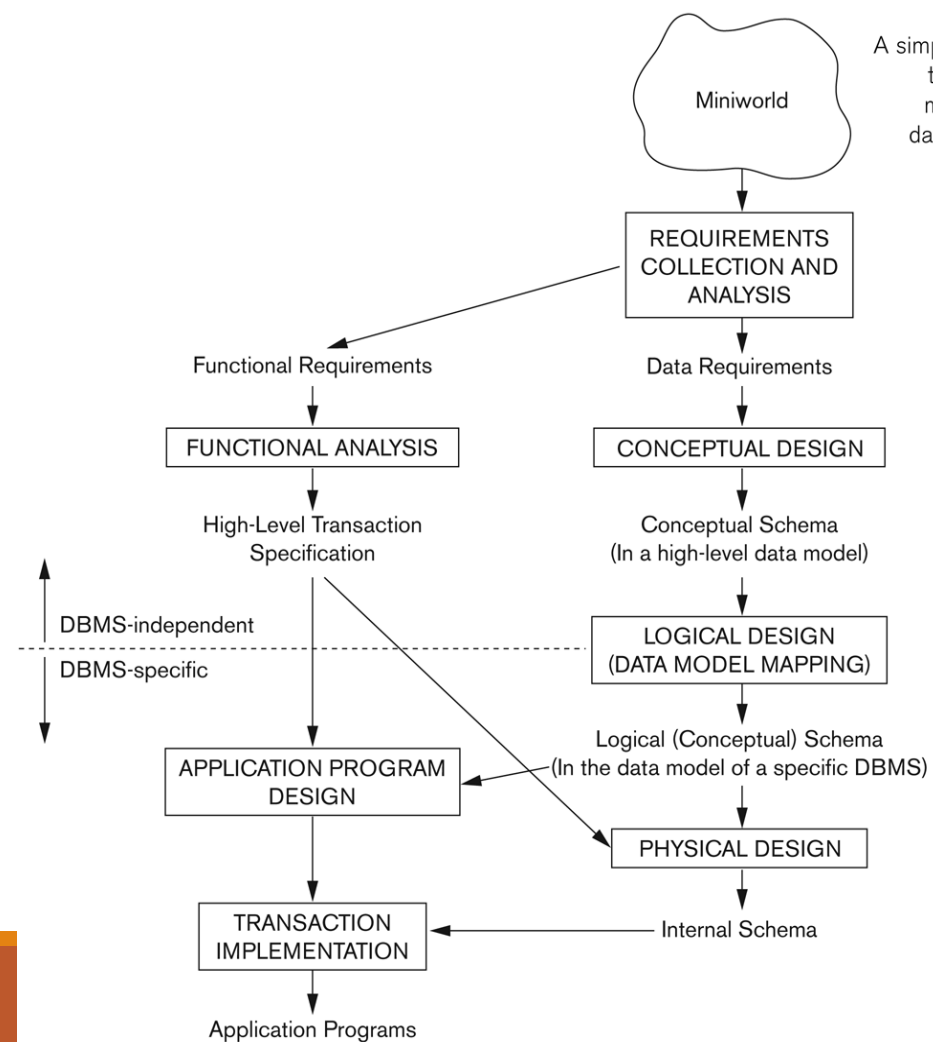


Figure 3.1

A simplified diagram to illustrate the main phases of database design.

Example COMPANY Database

We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:

- The company is organized into DEPARTMENTS. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager. 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.

Example COMPANY Database (Contd.)

- We 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, we keep track of their name, sex, birthdate, and relationship to the employee.

ER Model

Peter Pin-Shan Chen is a Taiwanese American computer scientist. He is a Distinguished Career Scientist and faculty member at Carnegie Mellon University, who is known for the development of the entity-relationship model in 1976.



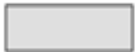

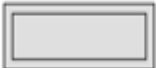
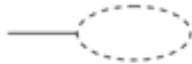








ER Model

The entity-relationship model is Graphical representation of entities and their relationships in a database structure.

ER models are normally represented in an entity relationship diagram(ERD), which uses graphical representations to model database components.

Entity Relationship Diagram (ERD) A detailed, logical representation of the entities, associations and data elements for an organization or business

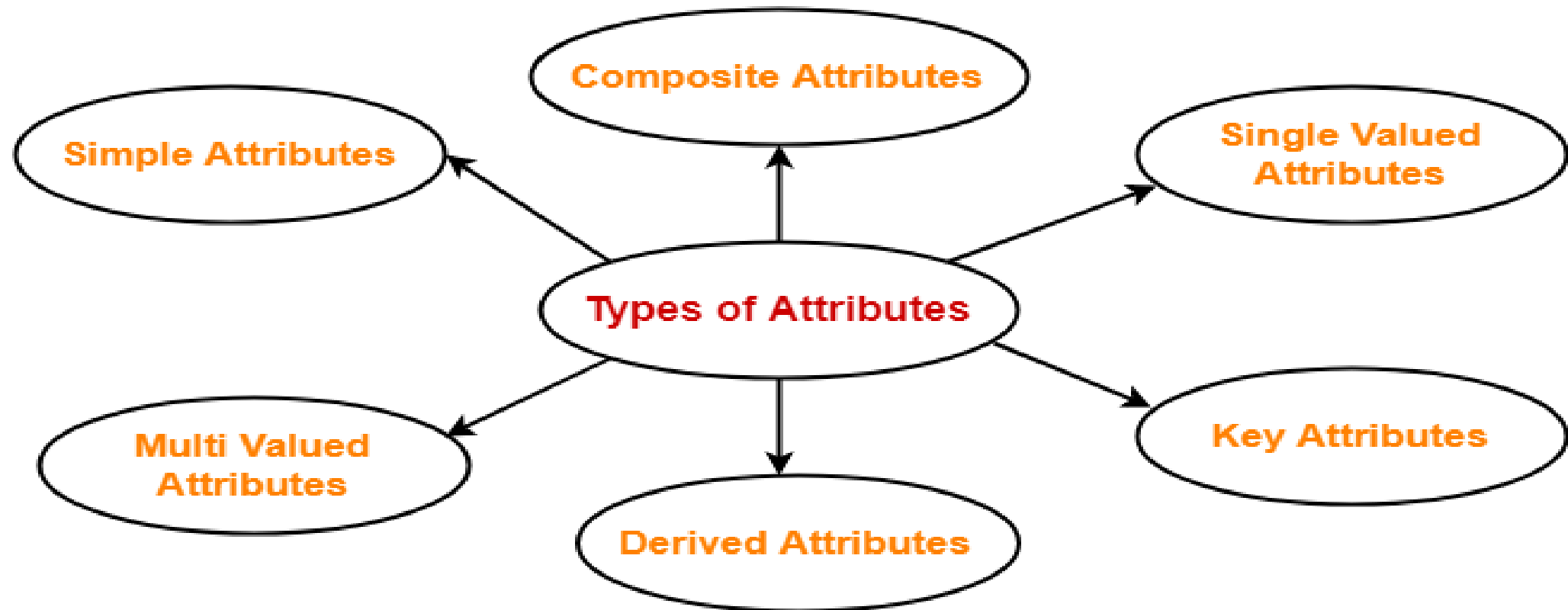
ER Model Components

Symbol	Meaning		
	Entity		Composite Attribute
	Weak Entity		Derived Attribute
	Relationship		Total Participation of E_2 in R
	Identifying Relationship		Cardinality Ratio 1: N for $E_1:E_2$ in R
	Attribute		Structural Constraint (min, max) on Participation of E in R
	Key Attribute		
	Multivalued Attribute		

ER Model Components

- **Entities** are specific objects or things in the real-world that are represented in the database.
- The name of the entity, a noun, is written in the center of the rectangle.
 - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT
- **Attributes** are properties used to describe an entity.
 - For example an EMPLOYEE entity may have the attributes Name, SSN, Address, Sex, BirthDate

Types of Attributes



Types of Attributes

Single Valued

- Each entity has a single atomic value for the attribute. For example: SSN or Sex.

Simple

- It means it can not be divided further. For example: SSN.

Composite

- The attribute may be composed of several components. For example:
 - Address(Apt#, House#, Street, City, State, ZipCode, Country), or
 - Name(FirstName, MiddleName, LastName).
 - Composition may form a hierarchy where some components are themselves composite.

Multi-valued

- An entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT.
 - Denoted as {Color} or {PreviousDegrees}.

Types of Attributes...Cont

Derived

- Attribute whose existence is dependent on some other attribute is called as has a single atomic value for the attribute. For example, SSN or Sex.

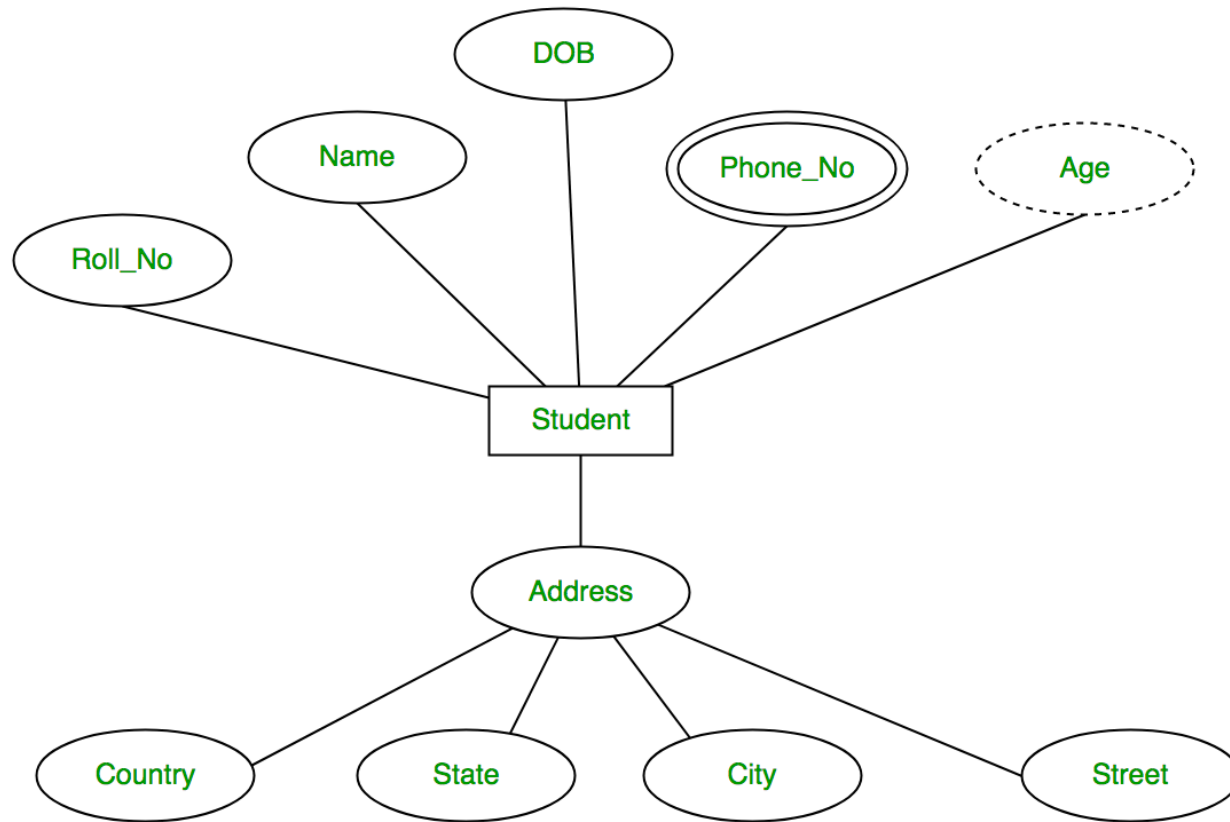
Key Attribute

- . Key attributes are those attributes which can identify an entity uniquely in an entity set.

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

- For example, PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}
- Multiple PreviousDegrees values can exist
- Each has four subcomponent attributes:
 - College, Year, Degree, Field

Example of types of attribute



Entity Type CAR with two keys and a corresponding Entity Set

(a)

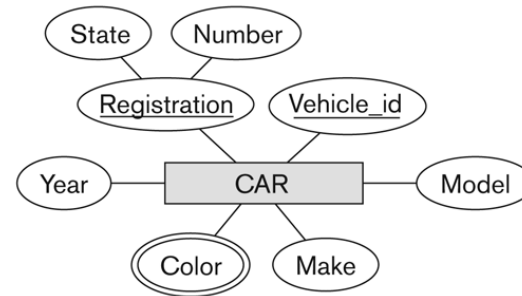


Figure 3.7

The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

(b)

CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR₁
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR₂
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR₃
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

Entity Set

Each entity type will have a collection of entities stored in the database

- Called the **entity set**

Previous slide shows three CAR entity instances in the entity set for CAR

Same name (CAR) used to refer to both the entity type and the entity set

Entity set is the current *state* of the entities of that type that are stored in the database

Initial Design of Entity Types for the COMPANY Database Schema

Based on the requirements, we can identify four initial entity types in the COMPANY database:

- DEPARTMENT
- PROJECT
- EMPLOYEE
- DEPENDENT

Their initial design is shown on the following slide

The initial attributes shown are derived from the requirements description

Initial Design of Entity Types:

EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

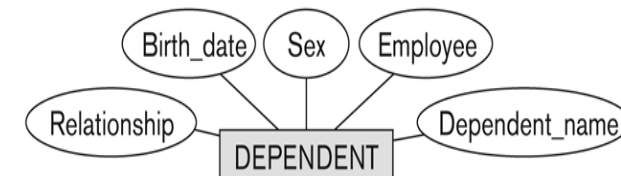
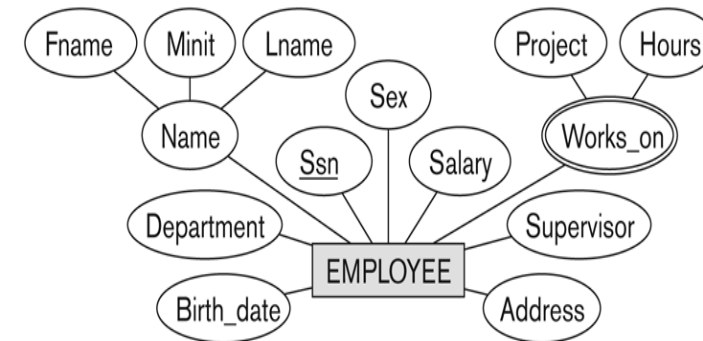
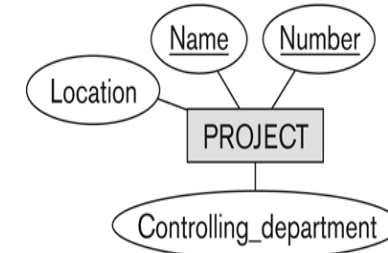
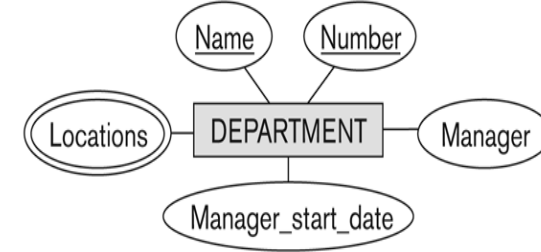


Figure 3.8

Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Refining the initial design by introducing **relationships**

The initial design is typically not complete

Some aspects in the requirements will be represented as **relationships**

ER model has three main concepts:

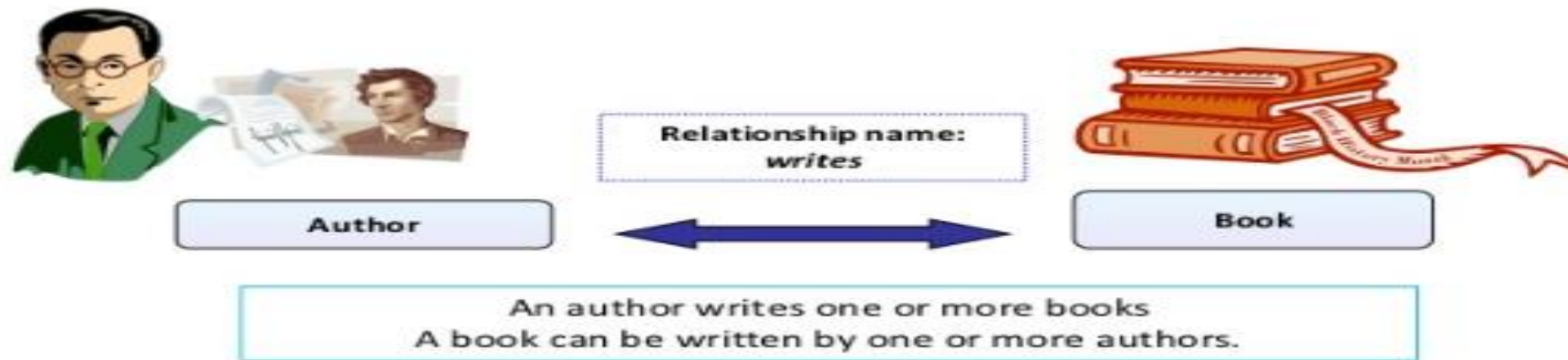
- Entities (and their entity types and entity sets)
- Attributes (simple, composite, multivalued)
- Relationships (and their relationship types and relationship sets)

We introduce relationship concepts next

Relationships and Relationship Types

A **relationship** relates two or more distinct entities with a specific meaning.

- For example, EMPLOYEE John Smith *works on* the ProductX PROJECT, or EMPLOYEE Franklin Wong *manages* the Research DEPARTMENT.



Relationships of the same type are grouped or typed into a **relationship type**.

- For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.

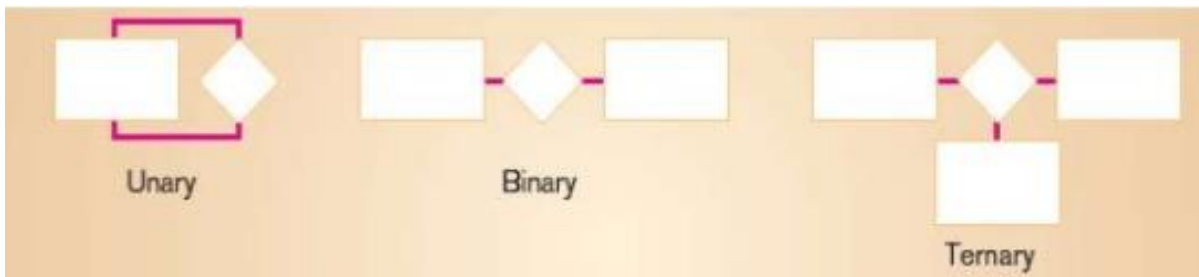
Degree of relationship

Degree: The number of entity types that participate in a relationship

For eg. Both MANAGES and WORKS_ON are *binary* relationships.

Types –

- Unary: between two instances of one entity type
- Binary: between the instances of two entity types
- Ternary: among the instances of three entity types



Cardinality

It defines relationships between entities by placing the relationship in the context of numbers. In an email system, for example, one account can have multiple contacts.



Cardinality in ER diagrams using UML notation

Relationship instances of the WORKS_FOR N:1 relationship between EMPLOYEE and DEPARTMENT

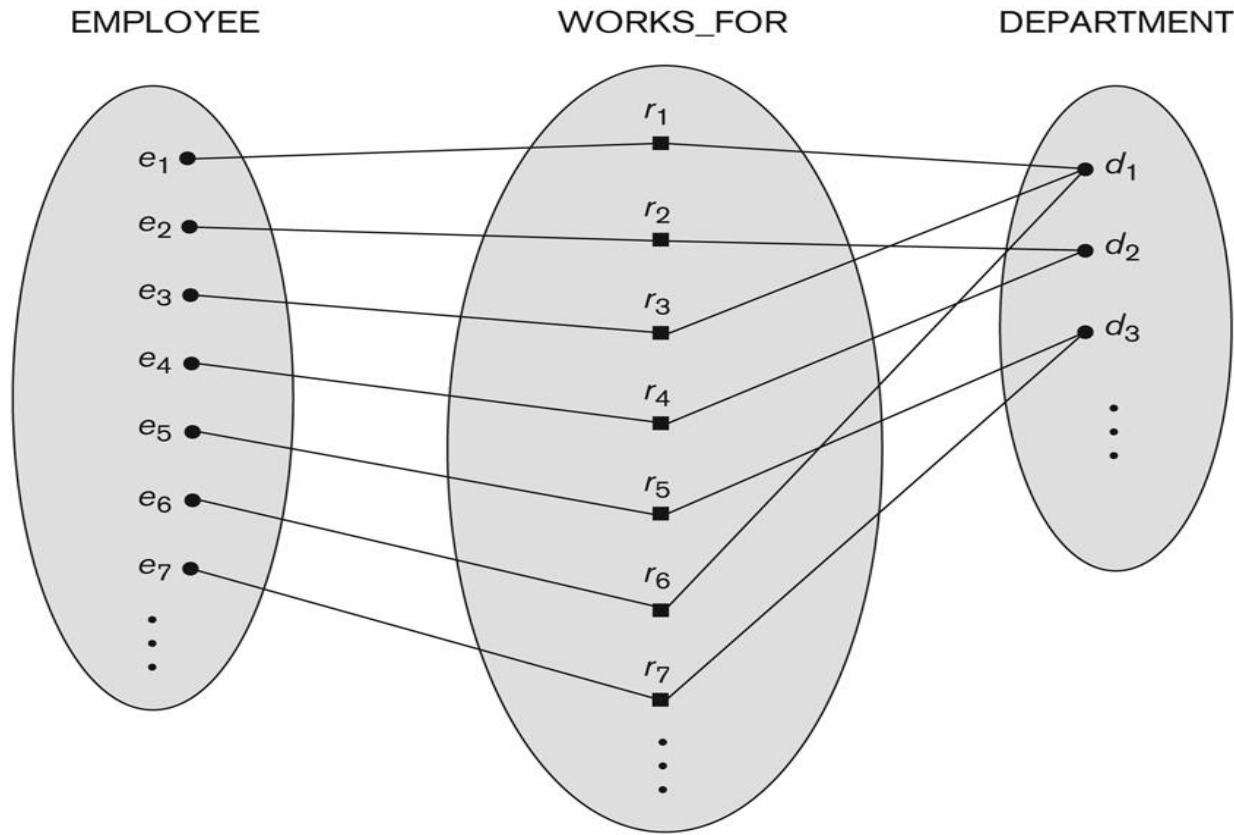


Figure 3.9

Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Relationship instances of the M:N WORKS_ON relationship between EMPLOYEE and PROJECT

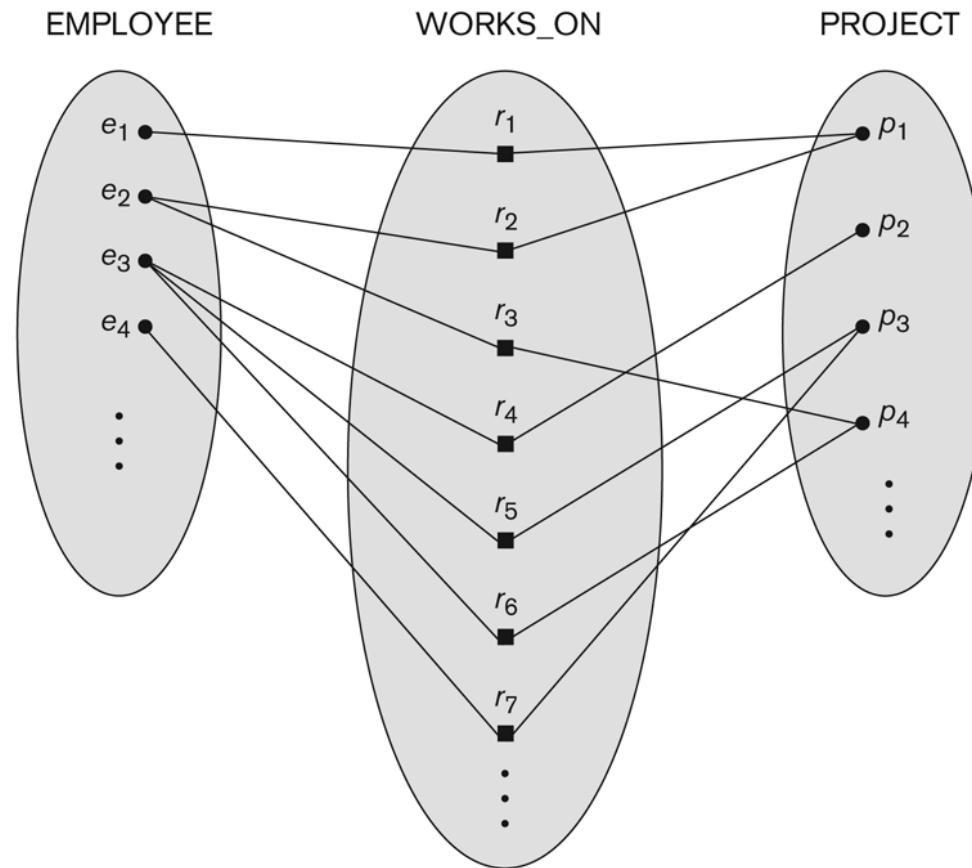


Figure 3.13
An M:N relationship,
WORKS_ON.

Refining the COMPANY database schema by introducing relationships

By examining the requirements, six relationship types are identified

All are *binary* relationships(degree 2)

Listed below with their participating entity types:

- WORKS_FOR (between EMPLOYEE, DEPARTMENT)
- MANAGES (also between EMPLOYEE, DEPARTMENT)
- CONTROLS (between DEPARTMENT, PROJECT)
- WORKS_ON (between EMPLOYEE, PROJECT)
- SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
- DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)

ER DIAGRAM – Relationship Types are:

WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF

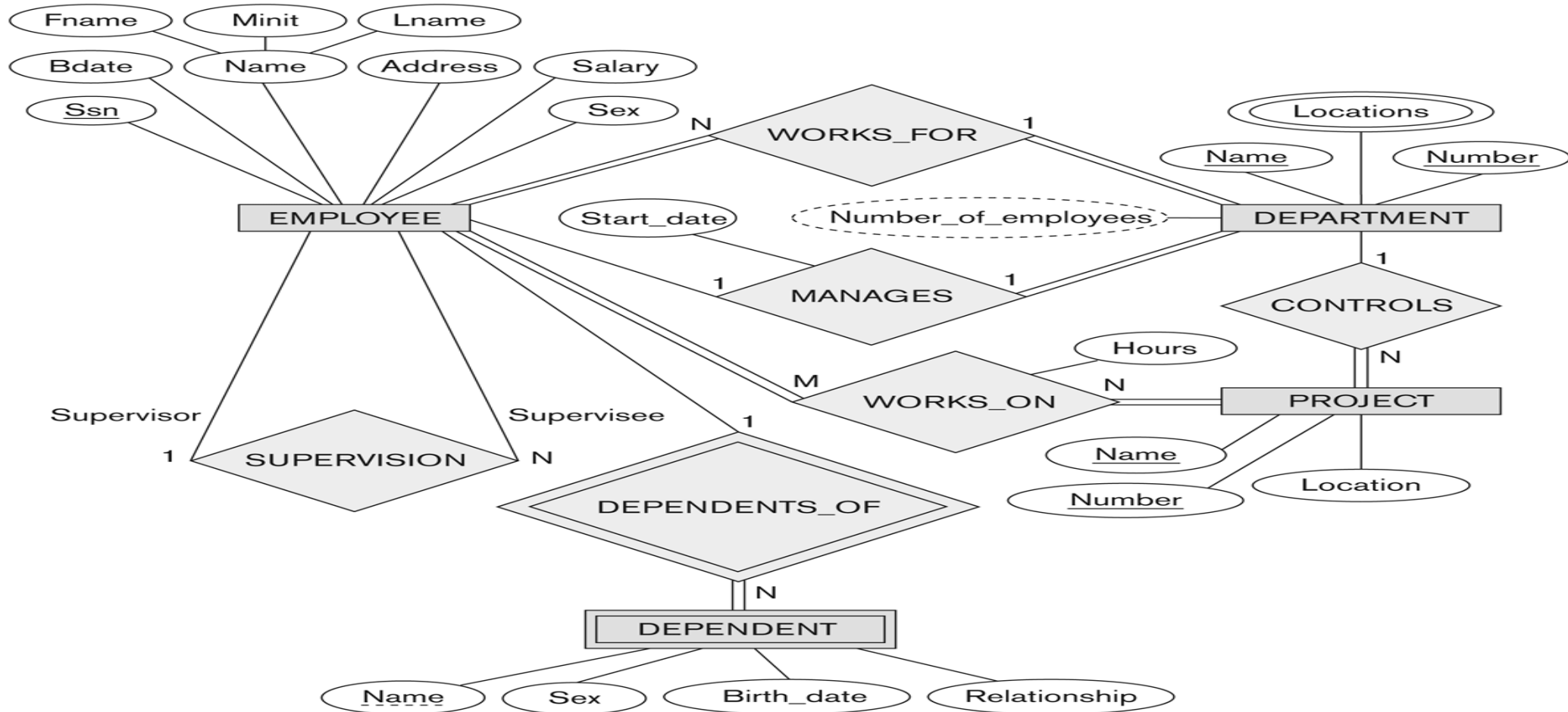


Figure 3.2

An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Discussion on Relationship Types

In the refined design, some attributes from the initial entity types are refined into relationships:

- Manager of DEPARTMENT -> MANAGES
- Works_on of EMPLOYEE -> WORKS_ON
- Department of EMPLOYEE -> WORKS_FOR
- etc

In general, more than one relationship type can exist between the same participating entity types

- MANAGES and WORKS_FOR are distinct relationship types between EMPLOYEE and DEPARTMENT
- Different meanings and different relationship instances.

Weak Entity Types

An entity that does not have a key attribute

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:

- A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
- Name of DEPENDENT is the *partial key*
- DEPENDENT is a *weak entity type*
- EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

Constraints on Relationships

Constraints on Relationship Types

- (Also known as ratio constraints)
- Cardinality Ratio (specifies *maximum* participation)
 - One-to-one (1:1)
 - One-to-many (1:N) or Many-to-one (N:1)
 - Many-to-many (M:N)
- Existence Dependency Constraint (specifies *minimum* participation) (also called participation constraint)
 - zero (optional participation, not existence-dependent)
 - one or more (mandatory participation, existence-dependent)

Notation for Constraints on Relationships

Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N

- Shown by placing appropriate numbers on the relationship edges.

Participation constraint (on each participating entity type): total (called existence dependency) or partial.

- Total shown by double line, partial by single line.

NOTE: These are easy to specify for Binary Relationship Types.

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

Specified on each participation of an entity type E in a relationship type R

Specifies that each entity e in E participates in at least *min* and at most *max* relationship instances in R

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

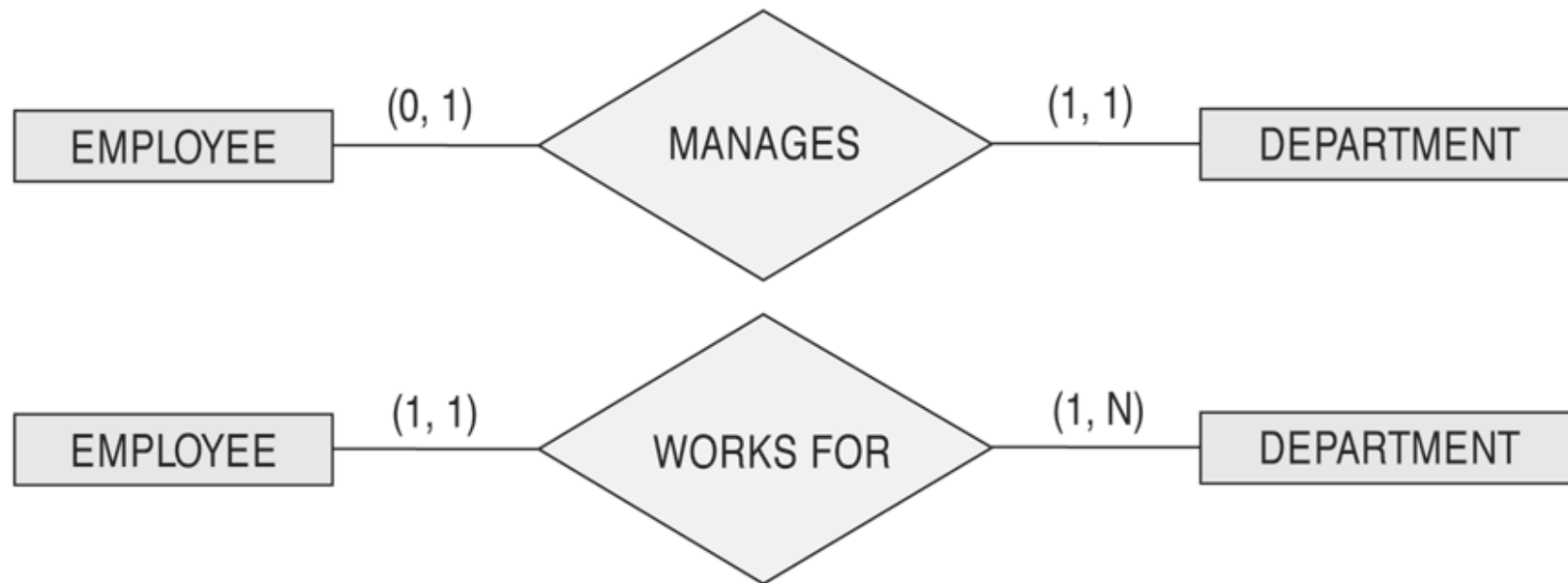
Must have $\min \leq \max$, $\min \geq 0$, $\max \geq 1$

Derived from the knowledge of mini-world constraints

Examples:

- A department has exactly one manager and an employee can manage at most one department.
 - Specify (0,1) for participation of EMPLOYEE in MANAGES
 - Specify (1,1) for participation of DEPARTMENT in MANAGES
- 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

The (min,max) notation for relationship constraints



Read the min,max numbers next to the entity type and looking **away from** the entity type

COMPANY ER Schema Diagram using (min, max) notation

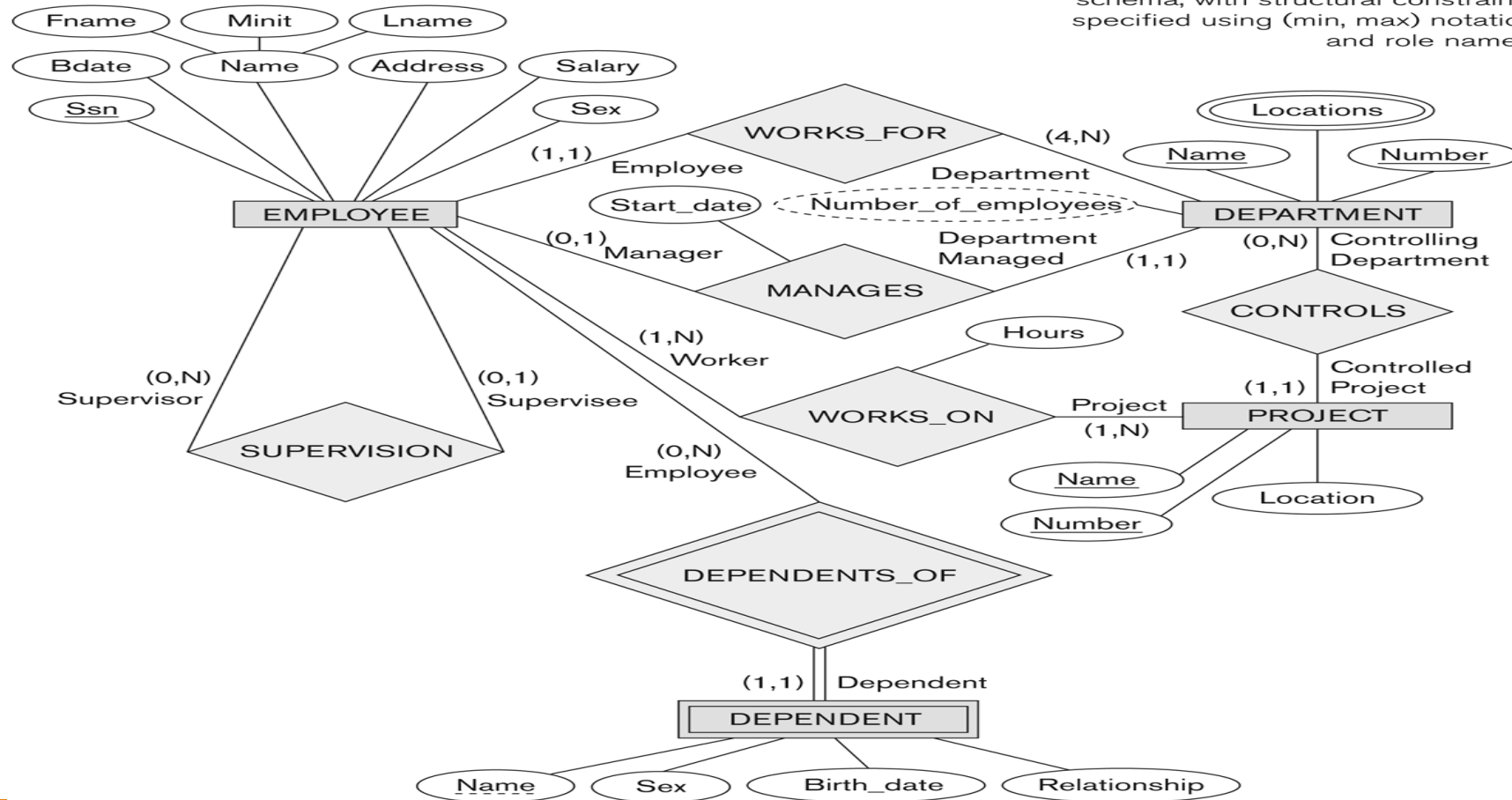


Figure 3.15

ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

Alternative diagrammatic notation

Chen's Notation

One to represent one



M to represent Many



Crows foot notation

 **Exactly One**

 **Zero or One**

 **Zero, One, or More**

 **One or More**

Extended E-R features

EER is a high-level data model that incorporates the extensions to the original ER model.

It is a diagrammatic technique for displaying the following concepts

Sub Class and Super Class

Specialization and Generalization

Aggregation

Features of EER Model

EER creates a design more accurate to database schemas.

It reflects the data properties and constraints more precisely.

It includes all modeling concepts of the ER model.

Diagrammatic technique helps for displaying the EER schema.

It includes the concept of specialization and generalization.

Generalization

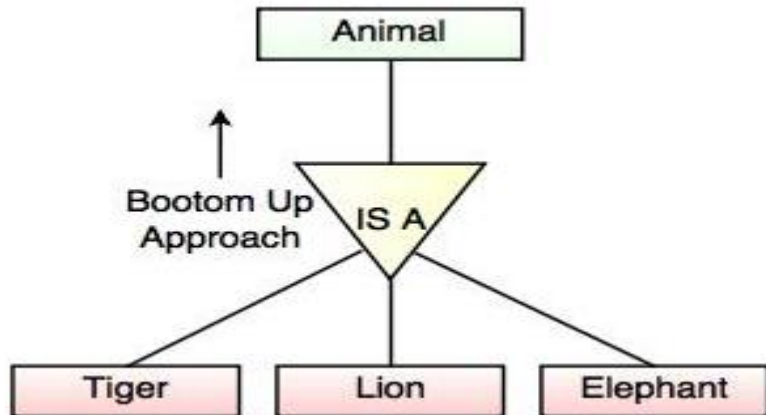
Generalization is the process of generalizing the entities which contain the properties of all the generalized entities.

It is a bottom approach, in which two lower level entities combine to form a higher level entity.

Generalization is the reverse process of Specialization.

It defines a general entity type from a set of specialized entity type.

It minimizes the difference between the entities by identifying the common features.



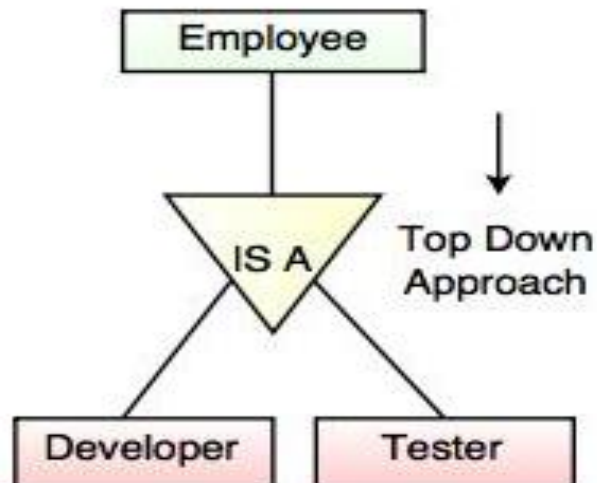
Specialization

Specialization is a process that defines a group entities which is divided into sub groups based on their characteristic.

It is a top down approach, in which one higher entity can be broken down into two lower level entity.

It maximizes the difference between the members of an entity by identifying the unique characteristic or attributes of each member.

It defines one or more sub class for the super class and also forms the superclass/subclass relationship.



Aggregation

Aggregation is a process that represent a relationship between a whole object and its component parts.

It abstracts a relationship between objects and viewing the relationship as an object.

It is a process when two entity is treated as a single entity.

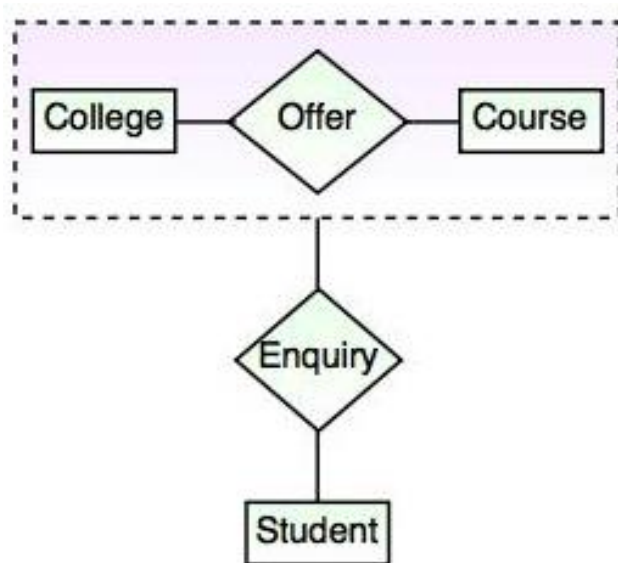


Fig. Aggregation

Reduction to E-R database schema

ER diagram is converted into the tables in relational model.

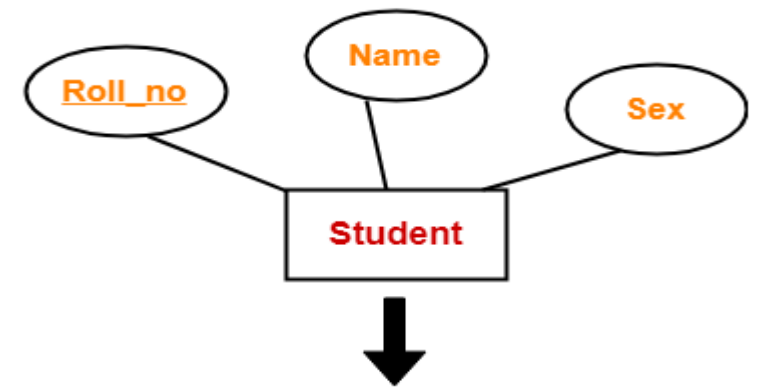
This is because relational models can be easily implemented by RDBMS like MySQL , Oracle etc.

Rule-01: For Strong Entity Set With Only Simple Attributes-

A strong entity set with only simple attributes will require only one table in relational model.

Attributes of the table will be the attributes of the entity set.

The primary key of the table will be the key attribute of the entity set.



<u>Roll no</u>	Name	Sex

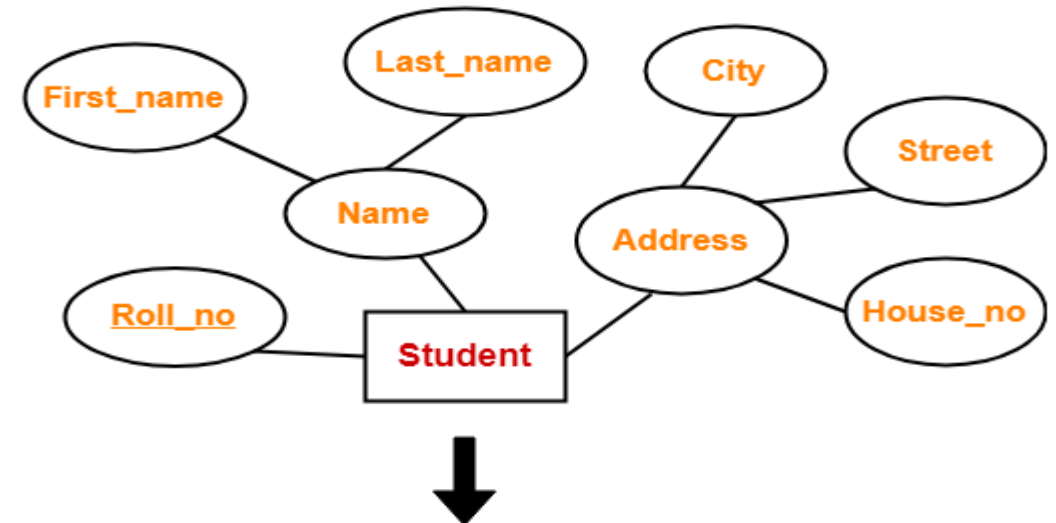
Schema : Student (Roll no , Name , Sex)

Reduction to E-R database schema

Rule-02: For Strong Entity Set With Composite Attributes-

A strong entity set with any number of composite attributes will require only one table in relational model.

While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.



<u>Roll_no</u>	First_name	Last_name	House_no	Street	City

Schema : Student (Roll_no , First_name , Last_name , House_no , Street , City)

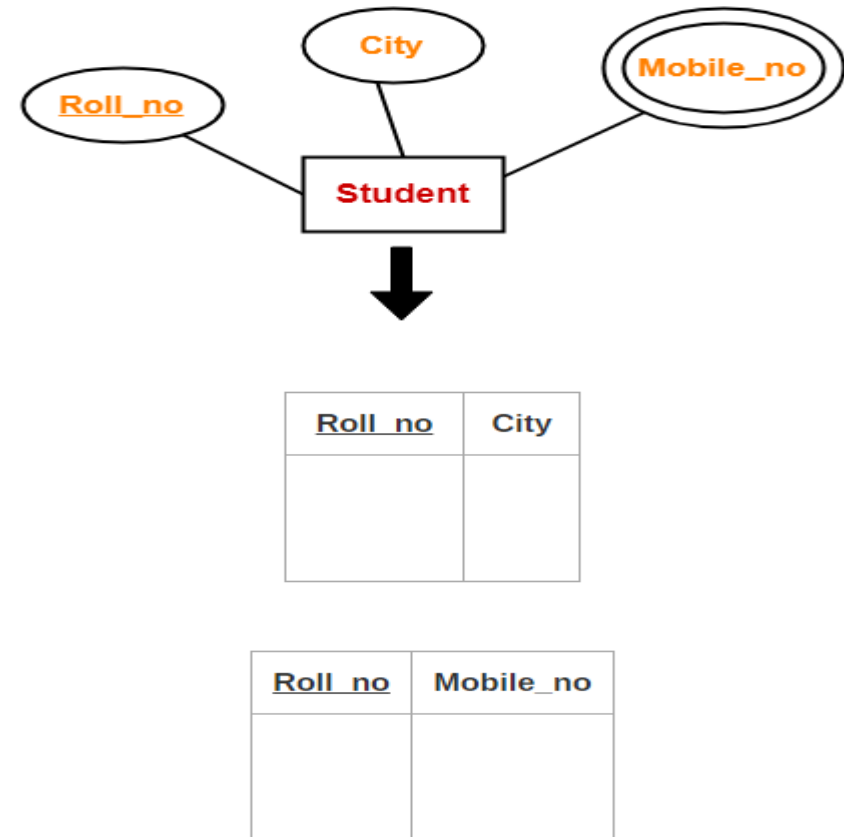
Reduction to E-R database schema

Rule-03: For Strong Entity Set With Multi Valued Attributes-

A strong entity set with any number of multi valued attributes will require two tables in relational model.

One table will contain all the simple attributes with the primary key.

Other table will contain the primary key and all the multi valued attributes.



Reduction to E-R database schema

Rule-04: Translating Relationship Set into a Table-

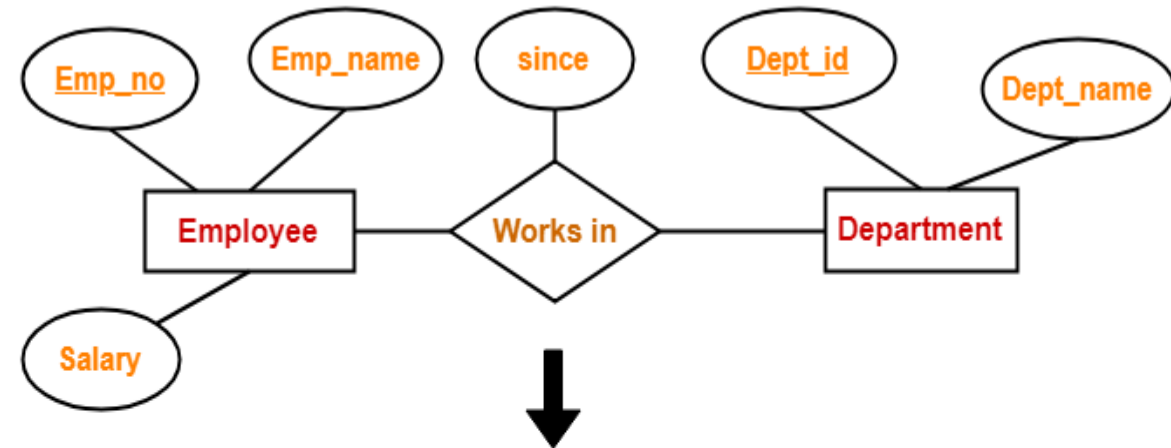
A relationship set will require one table in the relational model.

Attributes of the table are-

- Primary key attributes of the participating entity sets

- Its own descriptive attributes if any.

Set of non-descriptive attributes will be the primary key.



<u>Emp_no</u>	<u>Dept_id</u>	since

Schema : Works in (Emp_no , Dept_id , since)

References

<https://www.gatevidyalay.com/er-diagrams-to-tables/>