Unit-2

DATA MODELING

- Entity Relationship Model : Types of Attributes, Relationship, Structural Constraints –
- Relational Model: Relational model Constraints –
- Mapping ER model to a relational schema
- Integrity constraints

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

Conceptual Modeling

- Conceptual modeling is a very important phase in designing a successful database application.
- The term database application refers to a particular database and the associated programs that implement the database queries and updates.
- A major part of the database application will require the design, implementation, and testing of these application programs.
- Traditionally, the design and testing of application programs has been considered to be part of software engineering rather than database design

Overview of Database Design Process

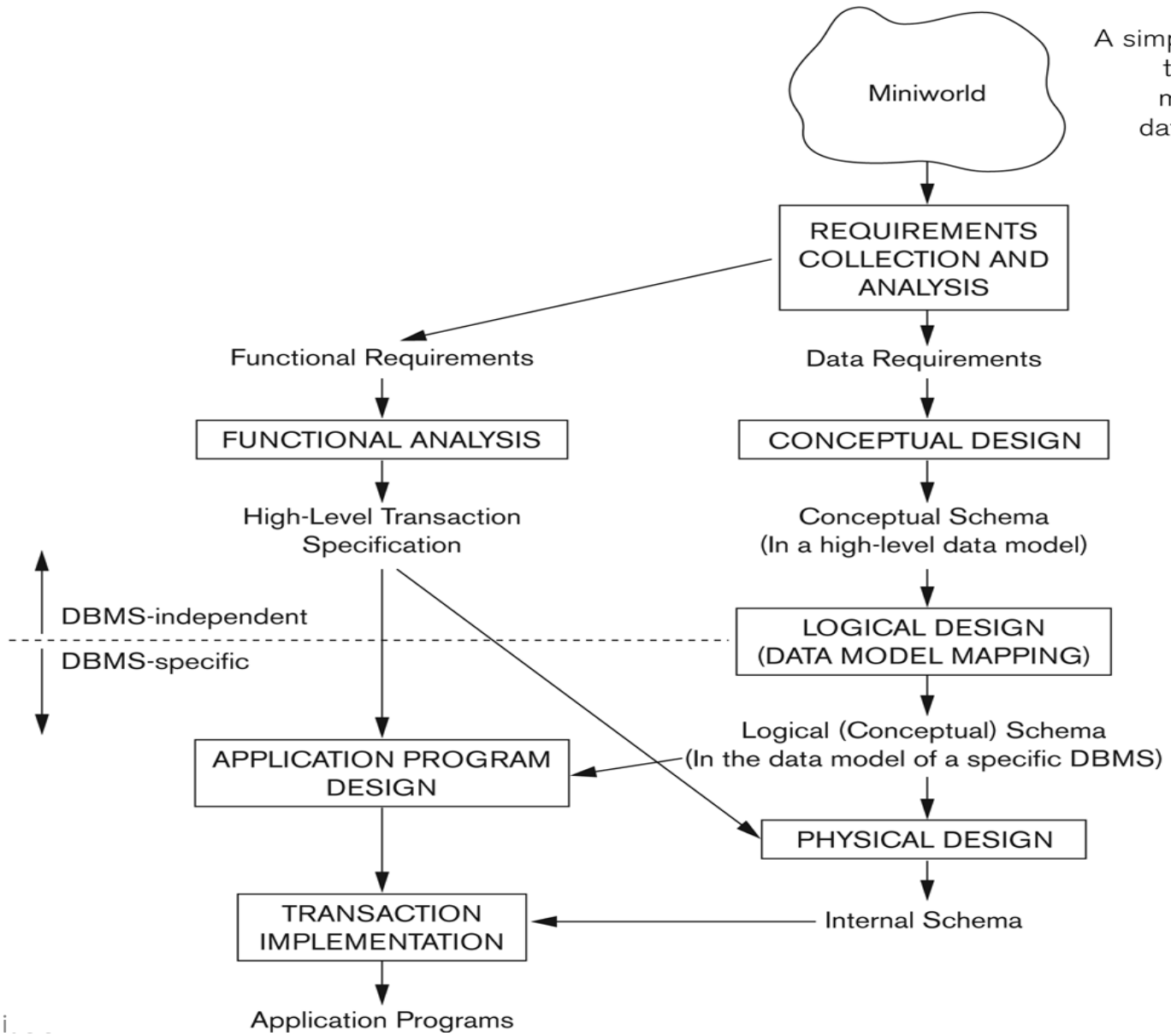
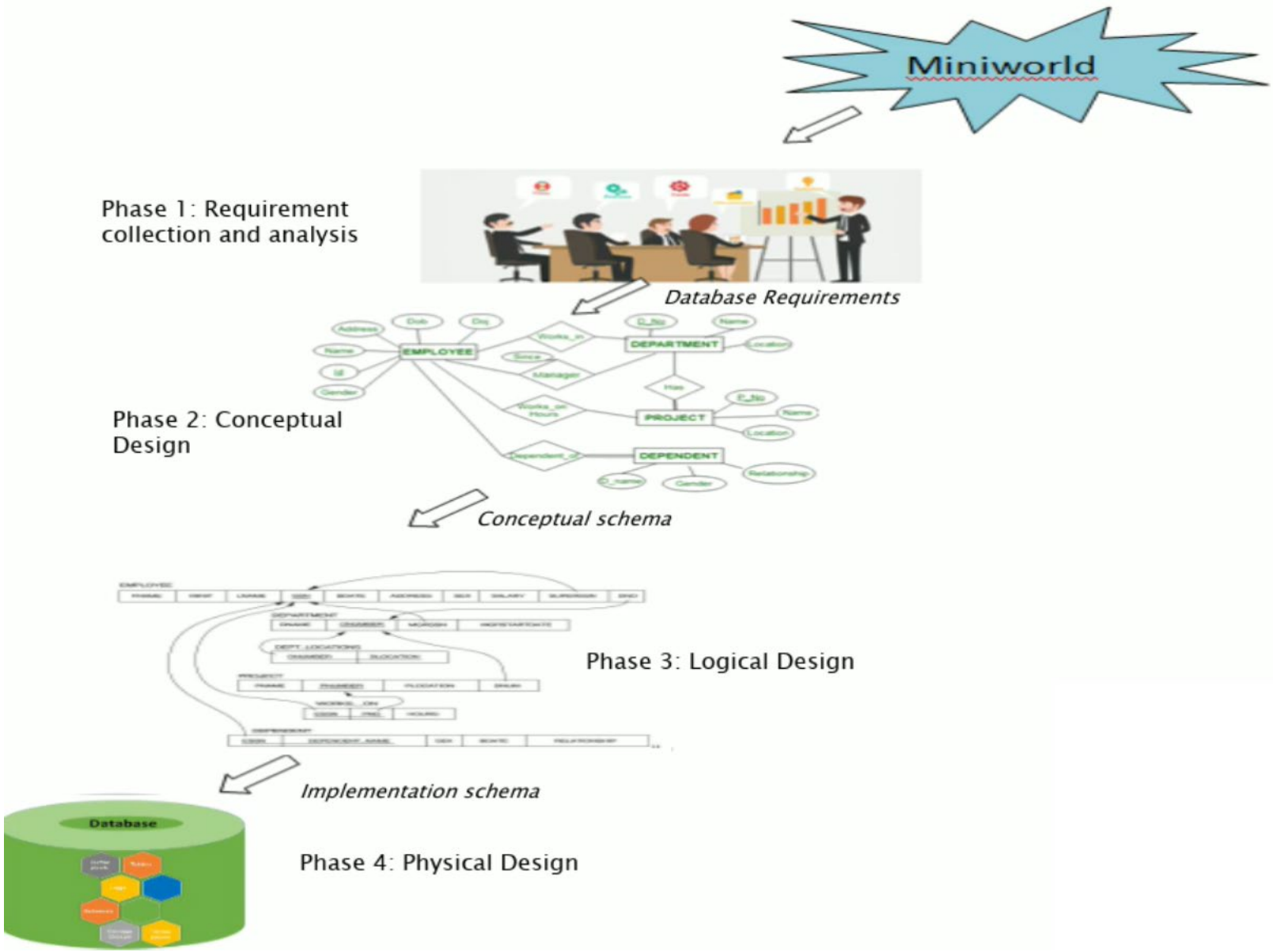


Figure 3.1
A simplified diagram to illustrate the main phases of database design.



- Courtesy- https://www.youtube.com/watch?v=9jc3lt_kekY

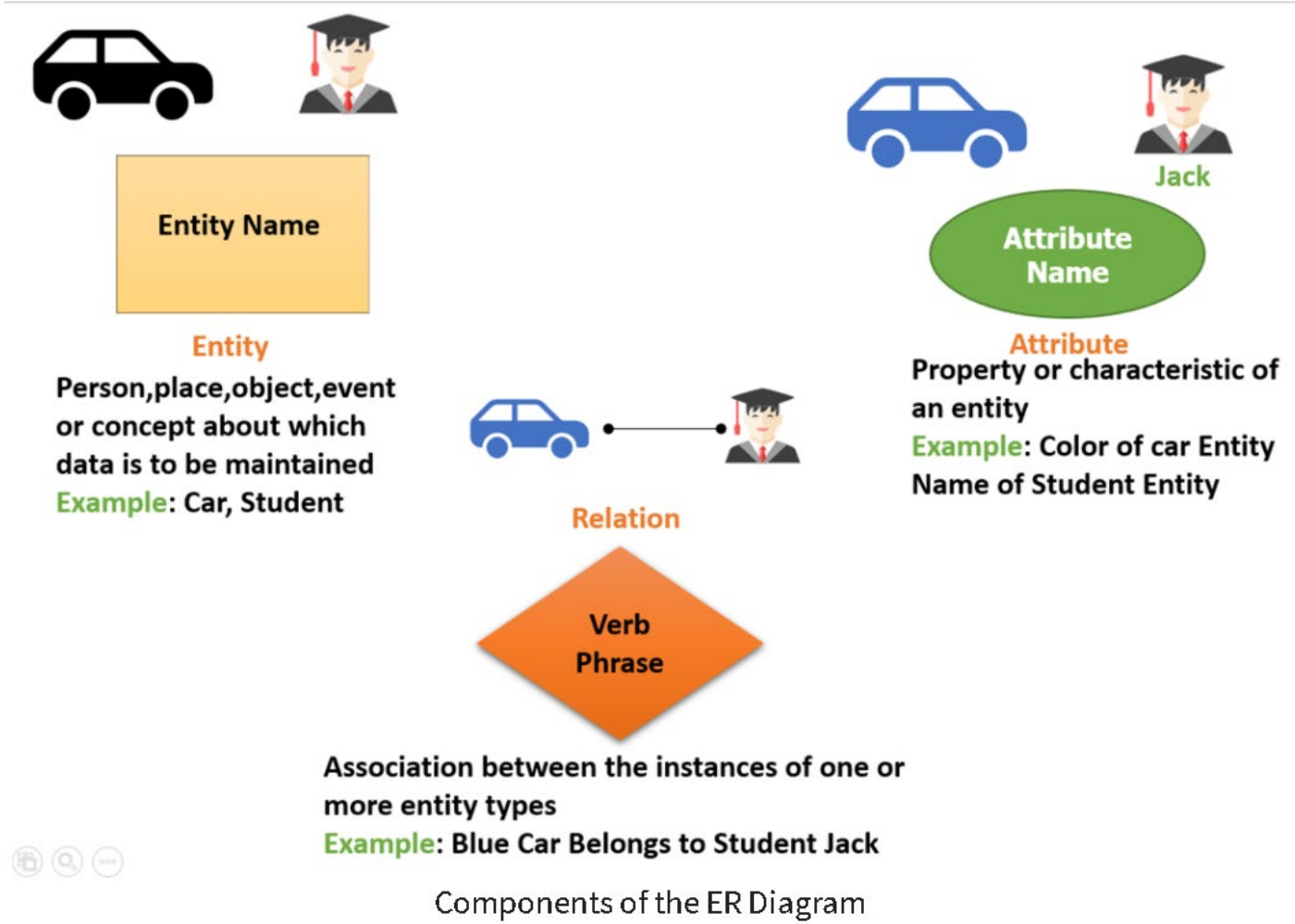
Entity Relationship Model

- It is a high-level data **model**.
- This **model** is used to define the data elements and **relationship** for a specified system.
- It develops a conceptual design for the database.
- It also develops a very simple and easy to design view of data.

ER Model Concepts

- Entities
  - Entities are specific objects or things in the mini-world that are represented in the database.
- Attributes
  - Attributes are properties used to describe an entity.
- Relationships
  - Relationship is an association among two or more entities.

Entity Relationship Model



Components of the ER Diagram

- Courtesy- https://www.guru99.com/er-diagram-tutorial-dbms.html

Characteristics of an Entity

- **Existence**:
  - concrete – have a physical existence in the real world
  - abstract – object with a conceptual existence
- **Described by its attributes**
- **Determined by particular value** of its attributes
  - a specific entity will have a value for each of its attributes(domain)
    - **Domain**: the set of permitted values for each attribute. e.g. integer, string, date, enumerated type

Entity Type and Entity Set

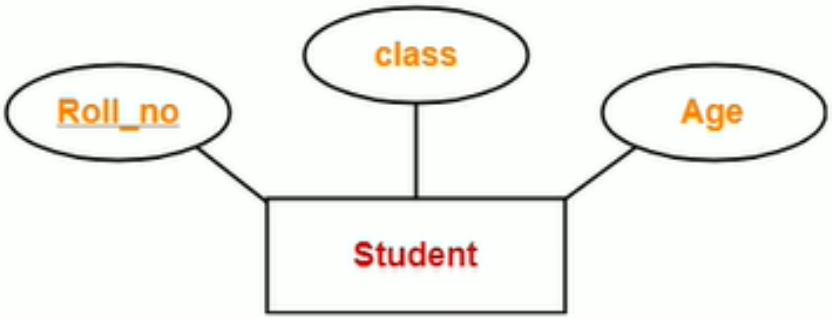
- An Entity Type defines a collection of entities that have the same attributes.
- An Entity Set is a collection of entities of an entity type at a point of time
  - Entity set is the current *state* of the entities of that type that are stored in the database

Types of Attributes

- Simple Attributes
- Composite Attributes
- Single valued Attributes
- Multivalued Attributes
- Derived Attributes
- Key Attributes

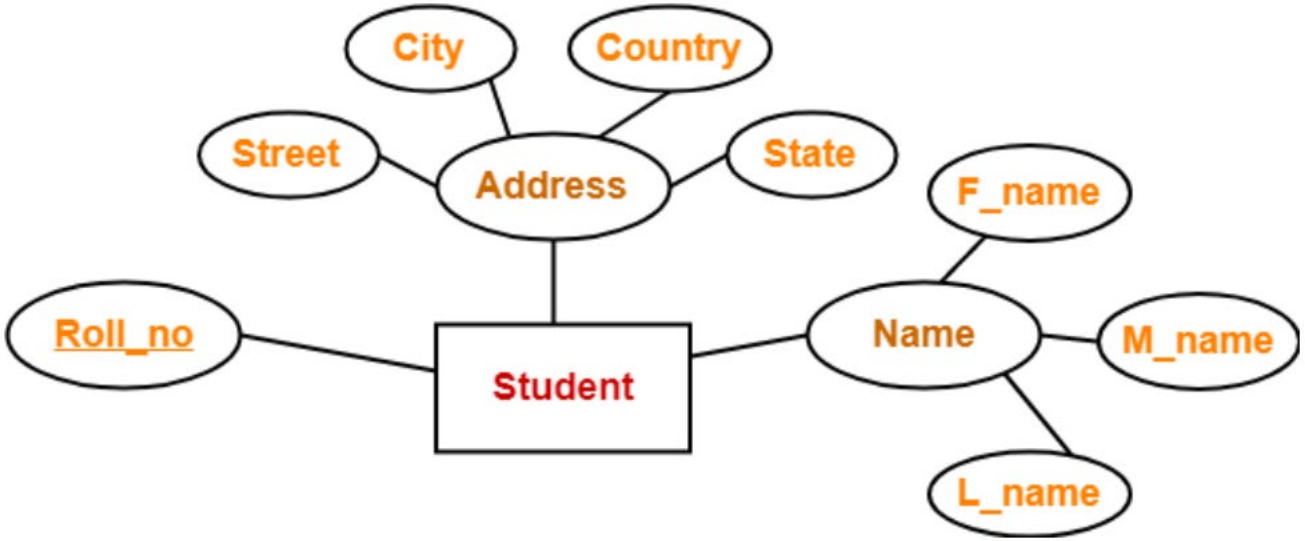
Simple attribute

Simple attributes are those attributes which can not be divided further



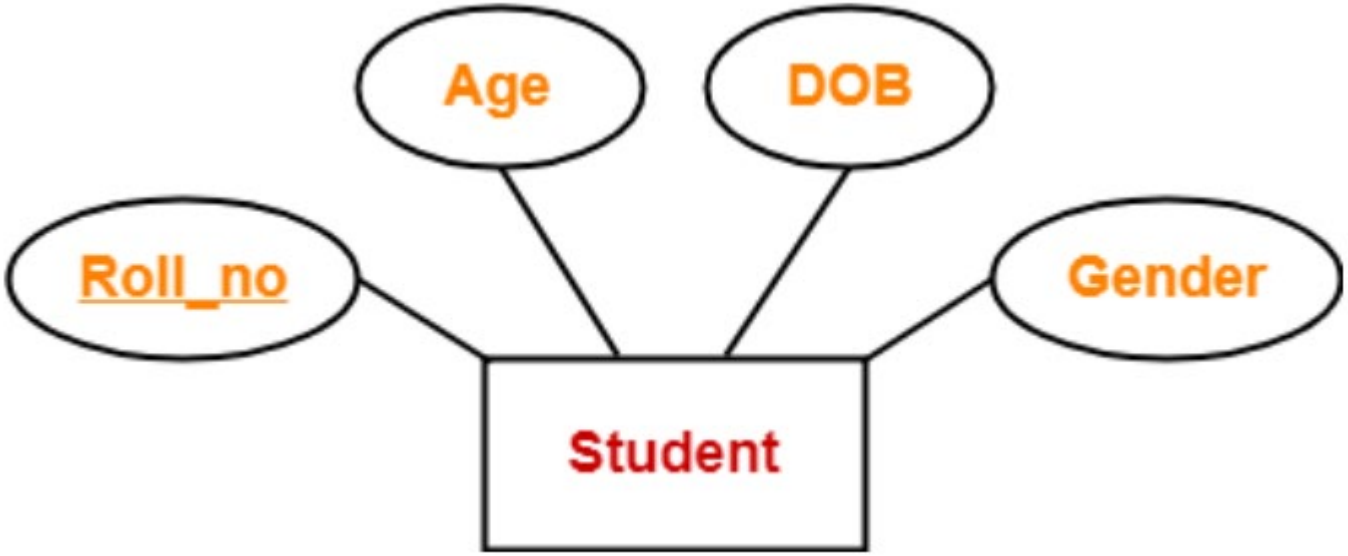
Composite Attributes

- 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.



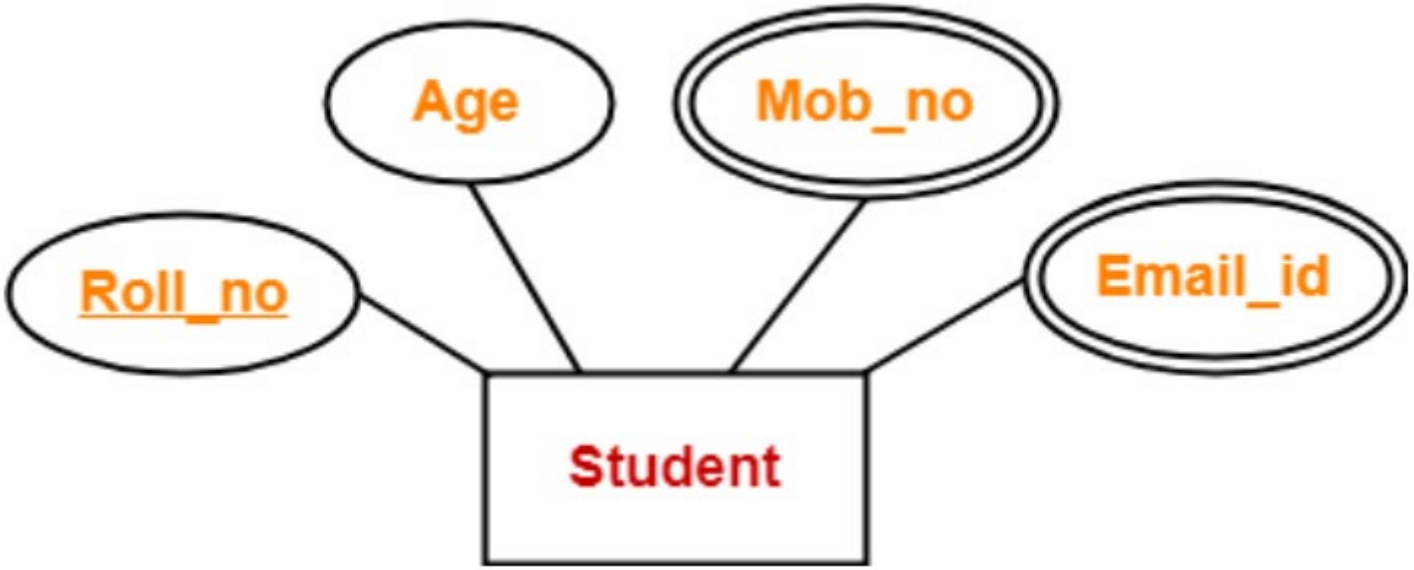
Single Valued Attributes

- Single valued attributes are those attributes which can take only one value for a given entity from an entity set.



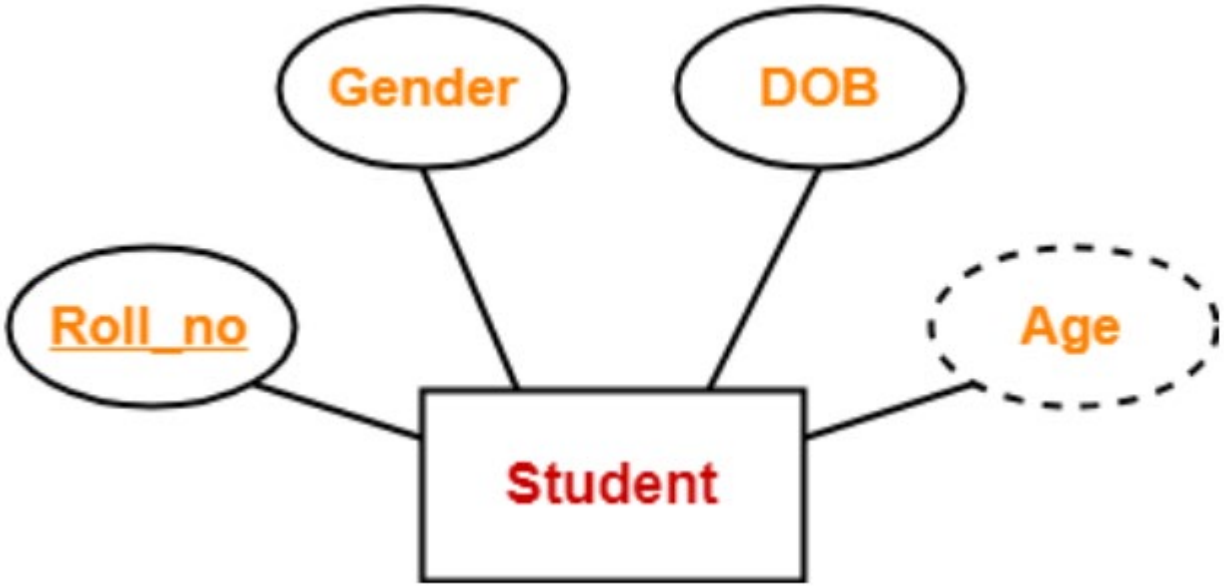
Multi Valued Attributes-

- Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set.



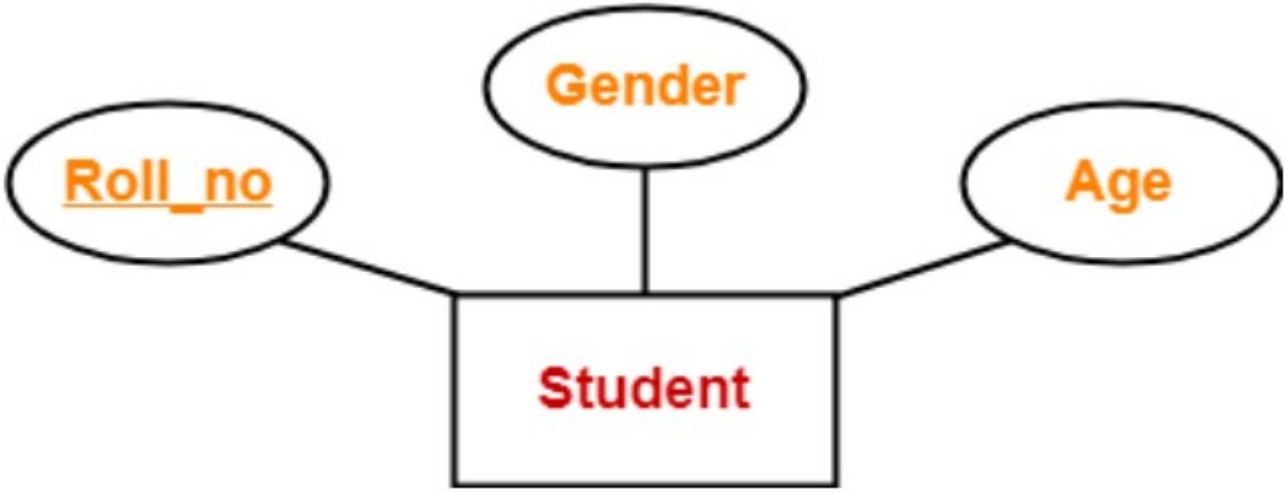
5. Derived Attributes-

- Derived attributes are those attributes which can be derived from other attribute(s).



6. Key Attributes-

- Key attributes are those attributes which can identify an entity uniquely in an entity set.
- It may be composite.



Value Sets (Domains) of Attributes

- Each simple attribute is associated with a value set
  - E.g., Lastname has a value which is a character string of upto 15 characters
  - Date has a value consisting of MM-DD-YYYY where each letter is an integer

- A **value set** specifies the set of values associated with an attribute

Displaying an Entity type

- In ER diagrams, an entity type is displayed in a rectangular box
- Attributes are displayed in ovals
  - Each attribute is connected to its entity type
  - Components of a composite attribute are connected to the oval representing the composite attribute
  - Each key attribute is underlined
  - Multivalued attributes displayed in double ovals

Notations for ER Diagram

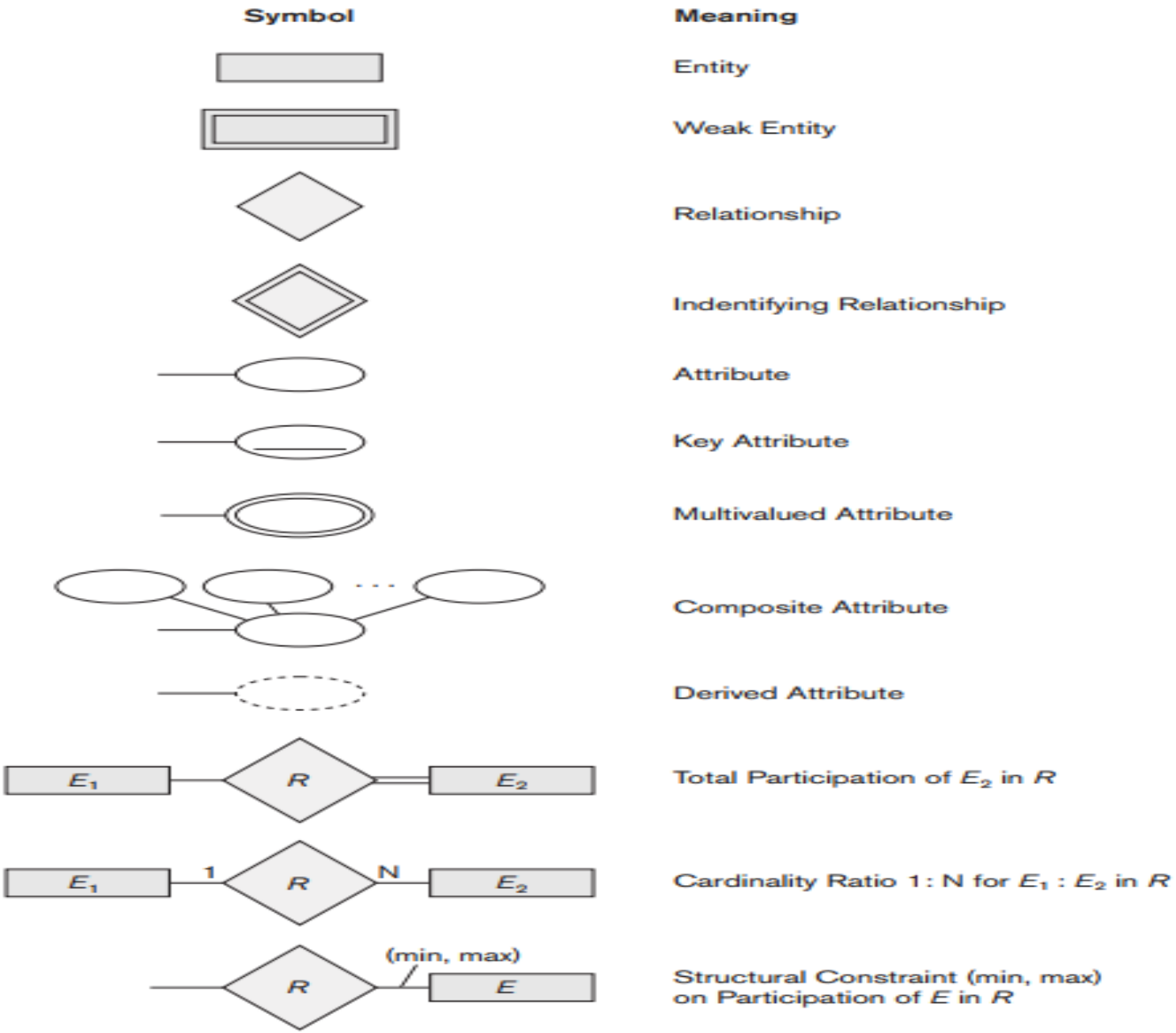


Figure 3.14
Summary of the notation for ER diagrams.

Relationship in ER Model

Relationships and Relationship types

- A **relationship** relates two or more distinct entities with a specific meaning.
  - For example, EMPLOYEE Sami works for IT DEPARTMENT
- Relationships of the same type are grouped or typed into a **relationship type**.
  - For example, the WORKS_FOR relationship type in which EMPLOYEEs and DEPARTMENTs participate

Relationship type vs Relationship state

- Relationship Type
  - Is the schema description of a relationship
  - Identifies the relationship name and the participating entity types
  - Also identifies certain relationship constraints
- Relationship Set
  - The current set of relationship instances represented in the database
  - The current *state* of a relationship type

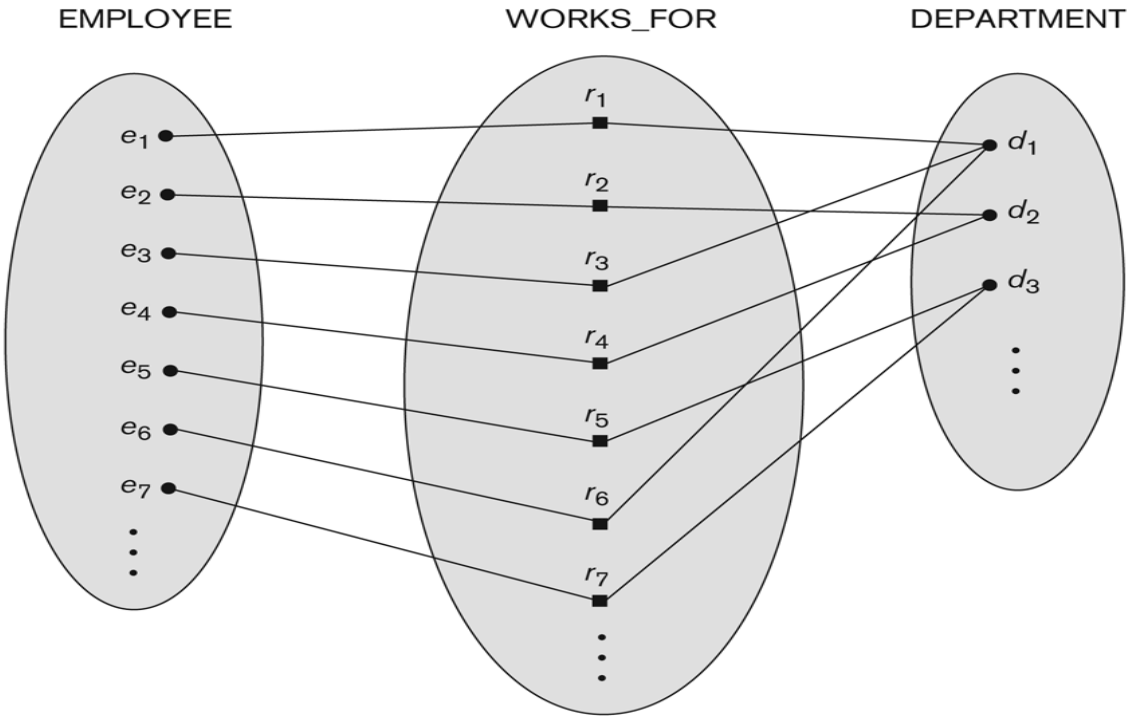
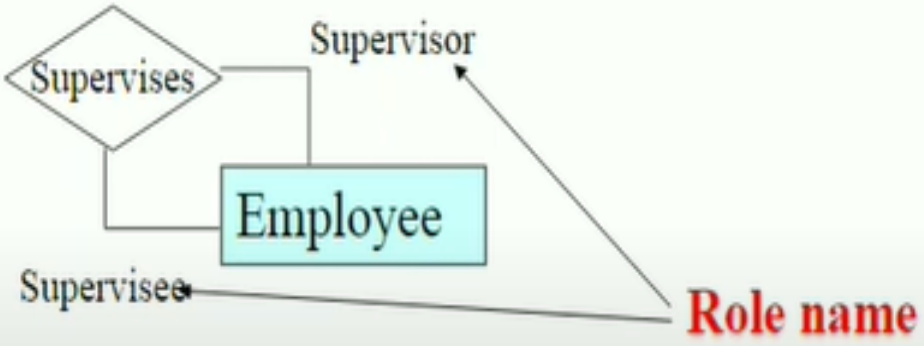


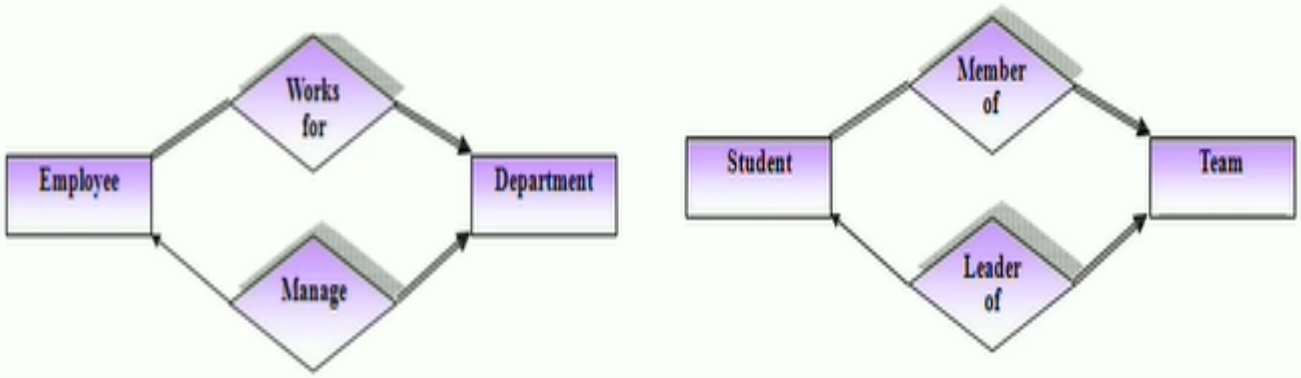
Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Recursive relationship

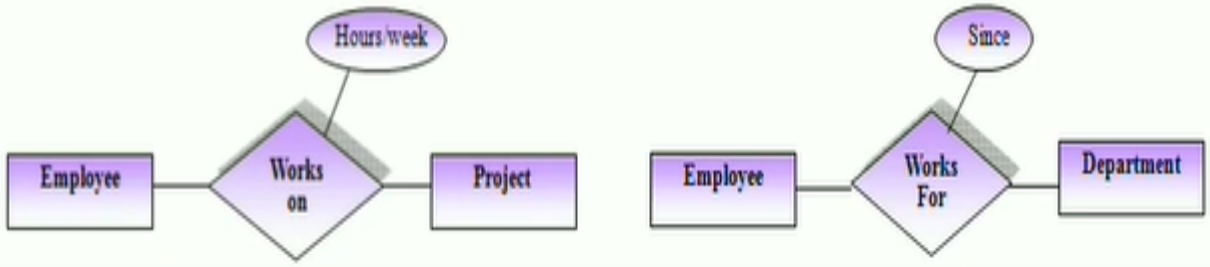
- Relationship types may associate an entity type with itself.
- The **roles** of the entity types in the relationship type are listed on the edges.



- More than one relationship type can exist with the same participating entity types

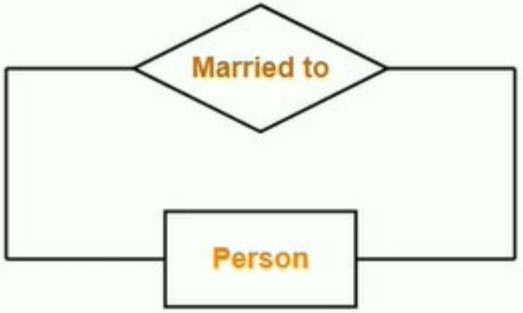


- A relationship type can have attributes called *descriptive* attributes

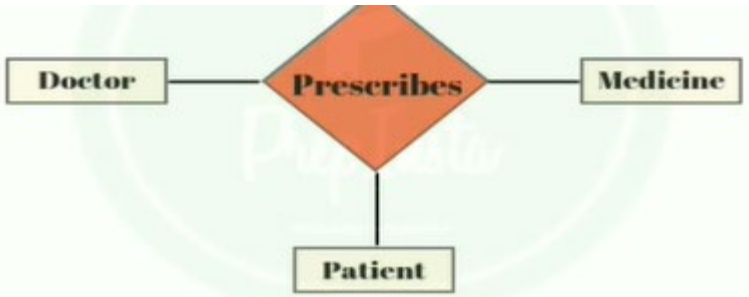


Degree of relationship

- The degree of a relationship type is the number of participating entity types.
  - ***Unary:*** related to another of the same entity type. Also called recursive relationships.
  - ***Binary*:** entities of two different types related to each other. (Two participating entities).
  - ***Ternary:*** entities of three different types related to each other. (three participating entities).
  - ***n-ary:*** entities of more than three different types related to each other.

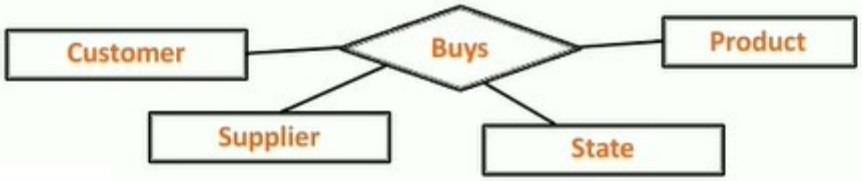


Unary relationship



Ternary Relationship

Binary relationship



N-ary relationship

Structural Constraints on relationship

- Mapping cardinalities
- Participation Constraints

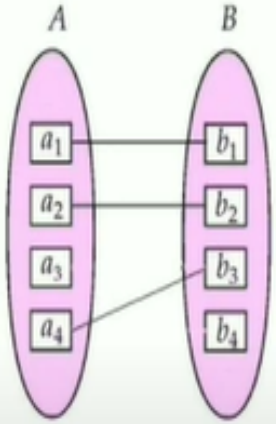
Mapping Cardinalities

- Mapping cardinalities, or cardinality ratio, express the number of entities to which another entity can be associated via a relationship set. (Specifies maximum participation).

- There are four types of cardinality: 1:1, 1:N, N:1, or M:N.

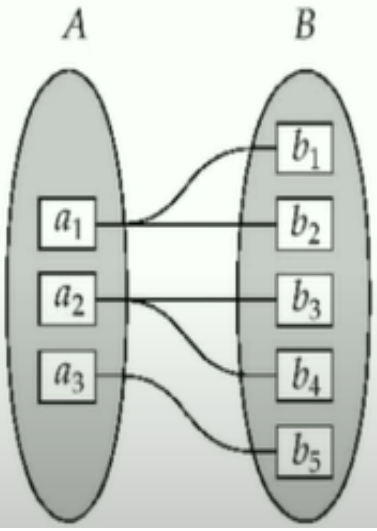
One to One relationship set

- **One-to-One (1:1)** cardinality ratio, an entity in one set is associated with at most one entity in another.



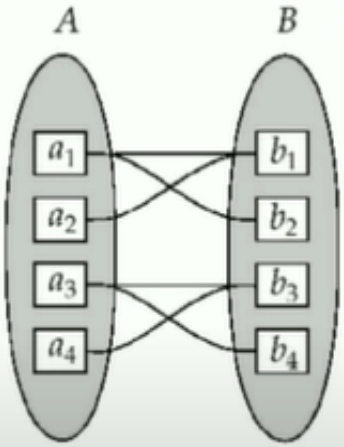
One to Many relationship

- **One-to-many (1: N)** cardinality ratio, an entity in the first set is associated with 0 or more entities in the second set. However, those entities in the second set can be associated with at most one entity in the first.



Many-to-many (N:M)

- **Many-to-many (N:M)** cardinality ratio, entities of either set may be associated with any number of entities in the other.



Participation Constraints
(Existence Dependency Constraints)

- Specifies minimum participation
- Specifies whether the existence of an entity **e** ϵ entity type **E** depends on its being related to another entity via the relationship type **R**.

Partial Participation

- If only some entities in E participate in relationships in R.
  - Each entity e ϵ E ***can*** participate in a relationship set.
  - Optional participation, not existence-dependent.
  - The minimum value is zero.
  - Shown by a single line.

Fig. Partial Participation

Total Participation

- If every entity in E participates in at least one relationship in R.
  - Each entity e ∈ E ***must*** participate in a relationship set.
  - Mandatory participation, existence-dependent
  - The minimum value is one or more.
  - Every entity in E participates in at least one relationship in R

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.

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

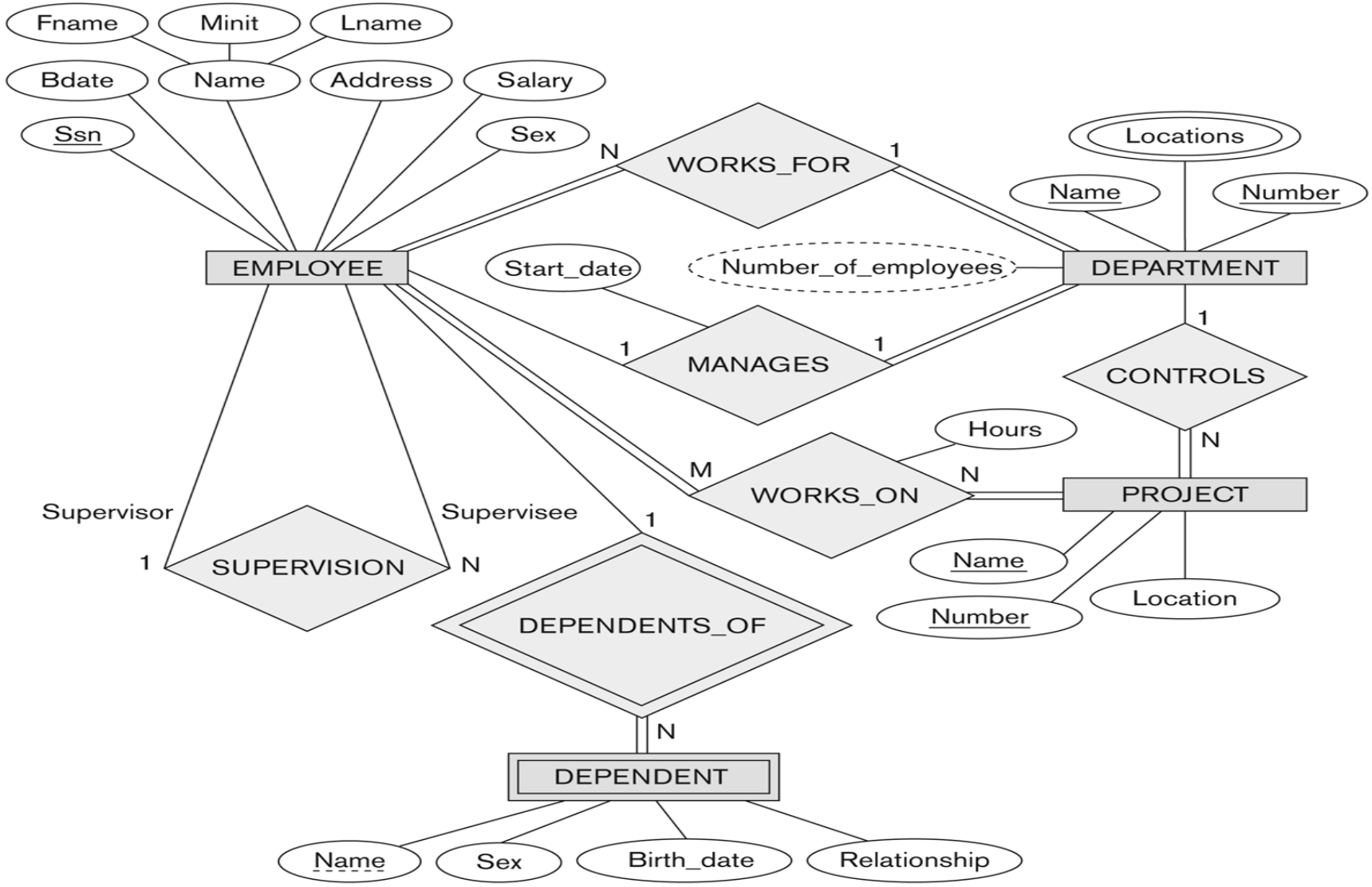


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

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.

Recursive Relationship Type

- An relationship type whose with the same participating entity type in **distinct roles**
- Example: the SUPERVISION relationship
- EMPLOYEE participates twice in two distinct roles:
  - supervisor (or boss) role
  - supervisee (or subordinate) role
- Each relationship instance relates two distinct EMPLOYEE entities:
  - One employee in *supervisor* role
  - One employee in *supervisee* role

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

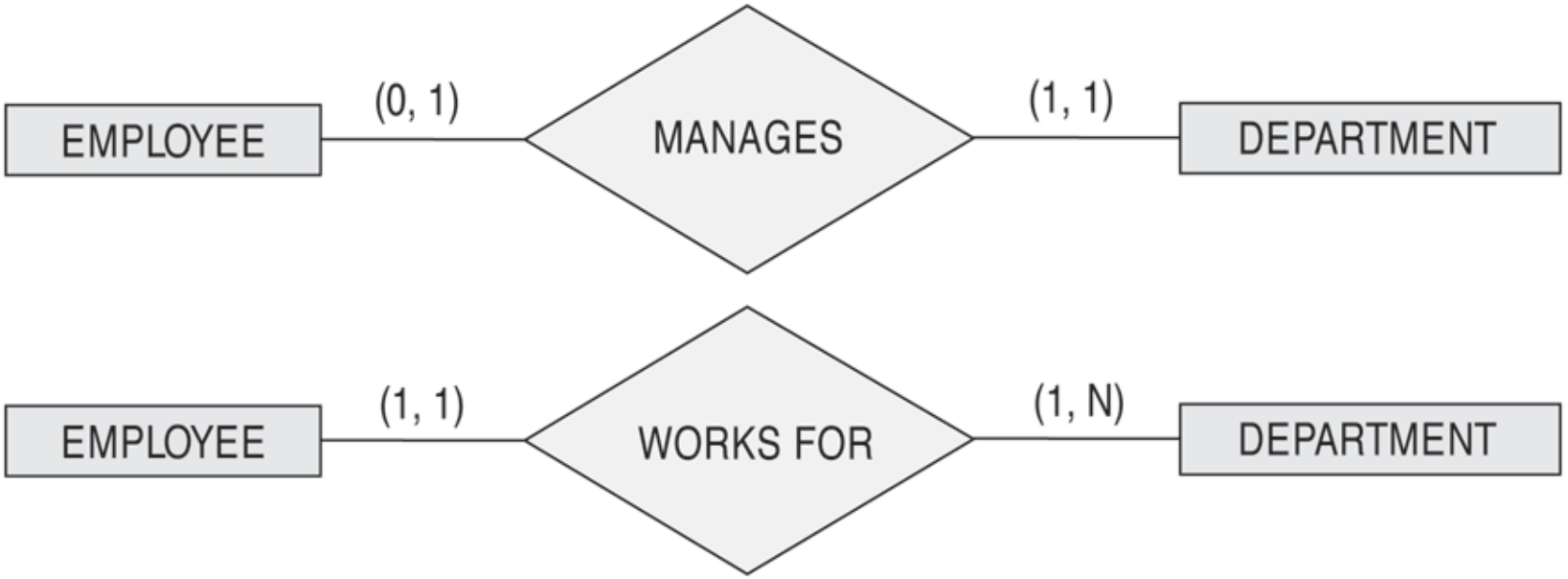
Constraints on Relationships

- 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)

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)
- 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

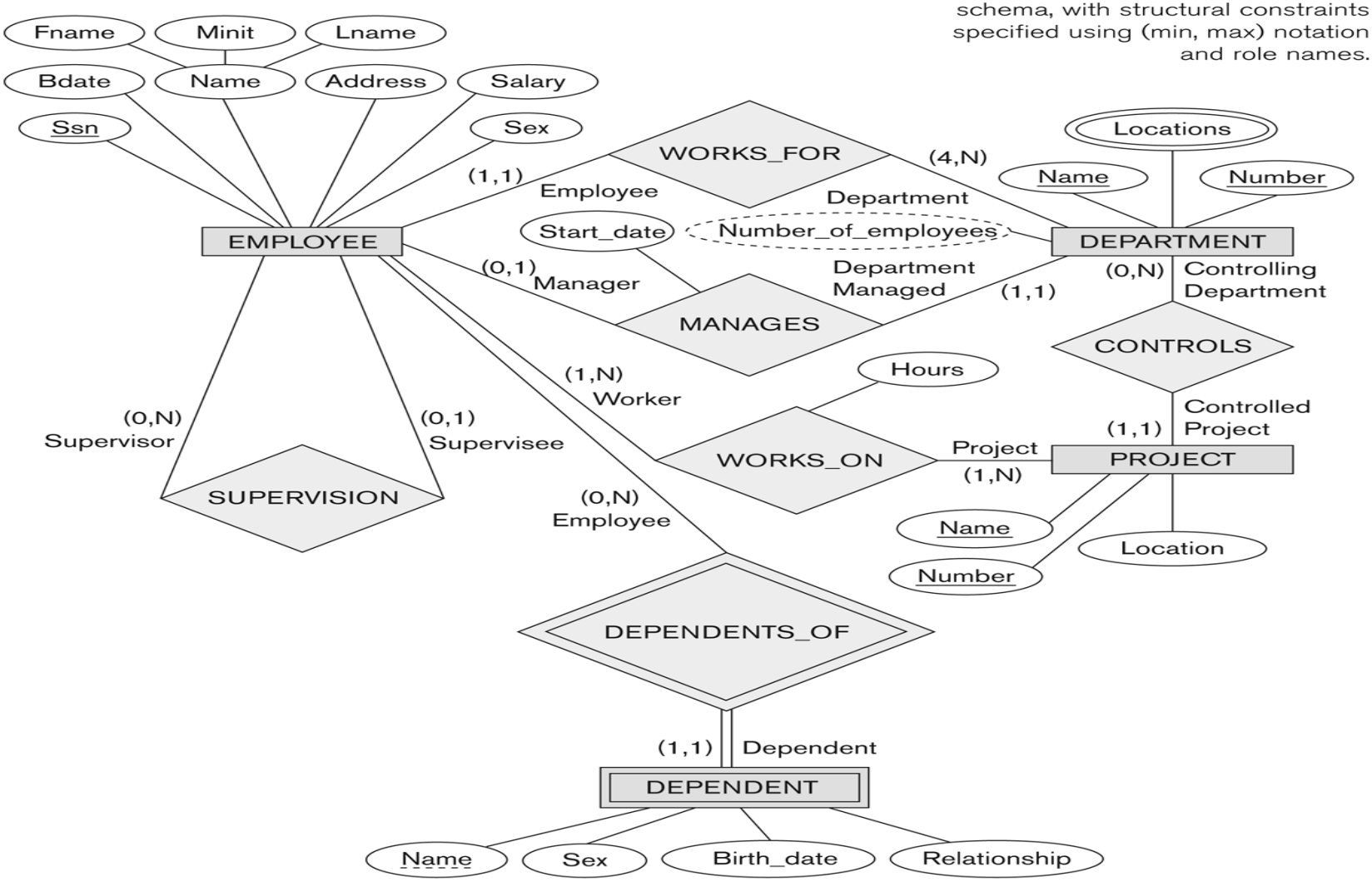
The (min,max) notation for relationship constraints



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

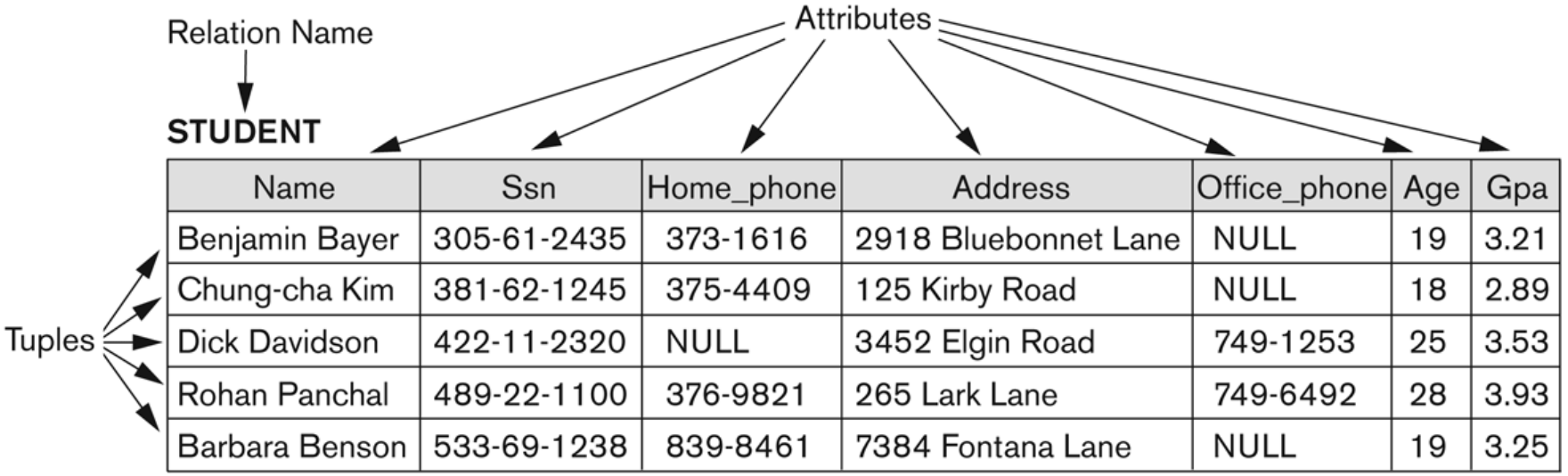
Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.



Informal Definitions-Relation

- Informally, a **relation** looks like a **table** of values.
- A relation typically contains a **set of rows**.
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
  - In the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
  - In the formal model, the column header is called an **attribute name** (or just **attribute**)

Example of a Relation



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

Informal Definitions

- Key of a Relation:
  - Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
    - Called the *key*
  - In the STUDENT table, SSN is the key

Formal Definitions - Schema

- The **Schema** (or description) of a Relation:
  - Denoted by R(A1, A2, ...An)
  - R is the **name** of the relation
  - The **attributes** of the relation are A1, A2, ..., An
- Example:

  CUSTOMER (Cust-id, Cust-name, Address, Phone#)
  - CUSTOMER is the relation name
  - Defined over the four attributes: Cust-id, Cust-name, Address, Phone#
- Each attribute has a **domain** or a set of valid values.
  - For example, the domain of Cust-id is 6 digit numbers.

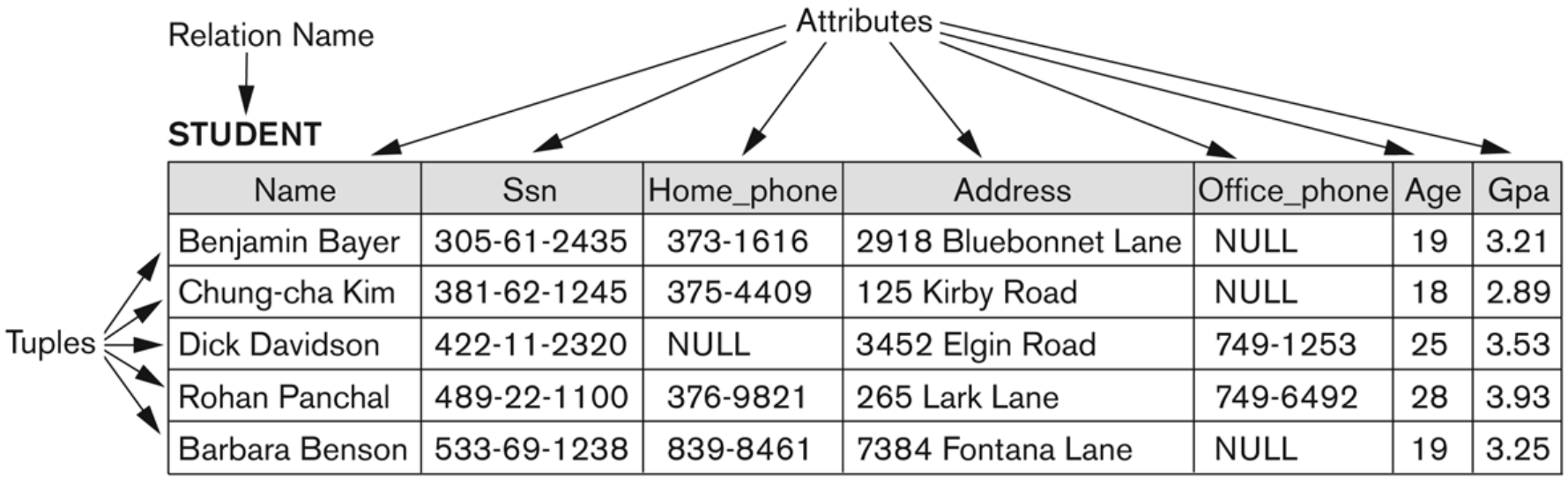
Formal Definitions - Tuple

- A **tuple** is an ordered set of values (enclosed in angled brackets '< … >')
- Each value is derived from an appropriate *domain*.
- A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:
  - <632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">
  - This is called a 4-tuple as it has 4 values
  - A tuple (row) in the CUSTOMER relation.
- A relation is a **set** of such tuples (rows)

Definition Summary

Informal Terms		Formal Terms
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

Example – A relation STUDENT



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states.
- There are three *main types* of constraints in the relational model:
  - **Key** constraints
  - **Entity integrity** constraints
  - **Referential integrity** constraints
- Another implicit constraint is the **domain** constraint
  - Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)

Domain Constraints

- These constraints specify that within each tuple, the value of each attribute 'A' must be an atomic value from the domain dom(A).

- The data types associated with domains typically include standard numeric data types for integers and real numbers.

- Consider domain of EID is number and accepts three digits.

EMP	E_ID	E_name	E_salary	E_Dno
	111	Govind	20000	1
	112	Adarsh	15000	1
	113	Bharath	20000	2
	11167	Yathish	15000	2

E_ID 11167 does not match with domain hence insertion will be rejected

Key Constraints and Constraints on NULL values

- A relation is a set of tuples, and each tuple's "identity" is given by the values of its attributes. Hence, it makes no sense for two tuples in a relation to be identical .

- No two tuples may have the same combination of values in their attributes.

EMP	E_ID	E_name	E_salary	E_Dno
	111	Govind	20000	1
	112	Adarsh	15000	1
	113	Bharath	20000	2
	111	Govind	20000	1

This is not allowed

Key and Super key

- Super Key
  - It is a subset of attributes SK where any two distinct tuples t1 and t2 in a relation state r of R, we have the constraint that t1[SK] ≠ t2[SK]
- Key
  - A key K of a relation schema R is a super key of R with the additional property that removing any attribute A from K leaves set of attributes K' that is not a super key of R any more.

EMP	E_ID	E_name	E_PAN	E_Passport
	111	Govind	APRT123	P234GTHD
	112	Adarsh	STQP427	NULL
	113	Bharath	NULL	P2768RPA
	114	Chetan	MKLW891	P237GTHD

Super keys:
SK1:E_ID
SK2:E_PAN
SK3:E_Passport
SK4:(E_ID, E_name)
SK5:(E_ID, E_PAN)
SK6:(E_ID, E_name,E_PAN)
There can be many more

- Candidate Key
  - A relation schema may have more than one key. In this case, each of the keys is called a candidate key.

- Primary Key
  - This is the candidate key whose values are used to identify tuples in the relation.
  - The primary key of the relation schema are underlined.

EMP	E_ID	E_name	E_PAN	E_Passport
	111	Govind	APRT123	P234GTHD
	112	Adarsh	STQP427	NULL
	113	Bharath	NULL	P2768RPA
	114	Chetan	MKLW891	P237GTHD

Candidate keys:
CK1:E_ID
CK2:E_PAN
CK3:E_Passport

Relational Databases and Relational Database Schemas

- Relational Database Schema
  - A relational database schema S is a set of relation schemas $S = \{R_1, R_2, \ldots, R_m\}$ and a set of integrity constraints IC.

- Relational Database State
  - A relational database state DB of S is a set of relation states $DB = \{r_1, r_2, \ldots, r_m\}$ such that each r_i is a state of R_i and such that the r_i relation states satisfy the integrity constraints specified in IC.

- A database state that does not obey all the integrity constraints is called an **invalid state**, and a state that satisfies all the constraints in IC is called a **valid state**.

Entity Integrity Constraints

- The entity integrity constraint states that no primary key can be NULL.
- This is because the primary key value is used to identify individual tuples in a relation.
- Having NULL values for the primary key implies that we can not identify some tuples.
- For example, if two or more tuples had NULL for their primary keys, we might not be able to distinguish them if we tried to reference them from other relations.

Entity integrity constraints

EMP	E_ID	E_name	E_salary	E_Dno
	111	Govind	20000	1
	112	Adarsh	15000	1
	113	Bharath	20000	2

Consistent database: E_ID does not have NULL values and repetition

111	Yathish	15000	2

Not allowed E_ID=111 already exists in tuple t1

NULL	Yathish	15000	2

E_ID=NULL is not allowedsince t4 tuple identification can not possible

Referential Integrity Constraints

- The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations.

- Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.

Foreign Key

- The condition for a foreign key specify a referential integrity constraint between the two relation schemas R_1 and R_2.
- A set of attributes FK in the relation schema R_1 is a foreign key of R_1 that references relation R_2 if it satisfies the following rules:
  - The attributes in FK have the same domain(s) as the primary key attributes PK of R_2; the attributes FK are said to reference or refer to the relation R_2.
  - The value of FK in a tuple t_1 of the current state $r_1(R_1)$ either occurs as a value of PK for some tuple t_2 in the current state $r_2(R_2)$ or is NULL. In the former case, we have $t_1[FK] = t_2[PK]$, and we say that the tuple t_1 references or refers to the tuple t_2.
- In this definition, R_1 is called the referencing relation and R_2 is the referenced relation.`

Referential integrity constraint

EMP	E_ID	E_name	E_salary	E_Dno
	111	Govind	20000	1
	112	Adarsh	15000	1
	113	Bharath	20000	2
	114	Yathish	15000	3

DEPT	D_No	D_name
	1	CS
	2	IT

Not allowed : Department with D_No =3 does not exist

Update Operations and Dealing with Constraint Violations

- The operations of the relational model can be categorized into retrievals and update.

- There are three basic update operations:
  - Insert
  - Delete
  - Modify