# Database Management System

## Lecture 15:
## Database Design

# Contents

- Entity Relationship Diagram (ERD)
- ***Entity***
- ***Attributes***
- ***Relationships (Degree/Types)***
- ***Constraints***
  - ***Mapping Cardinalities/Relationship Types***

# Database: **library management system**

- **authors ( author_id, author_name, author_address)**
- **publisher ( publisher_id, name, address)**
- **books( book_id, book_title, *author_id, publisher_id*, price, qunatity)**
- **borrowers( id, firstname, lastname, phone, email, address)**
- **borrow_book ( id, *borrower_id, book_id*, borrowing_date, return_date)**

# Database: **ecommerce**

- **Company( company_id, name, contact, address)**
- **Customer( cust_id, name, phone, city)**
- **Product( pro_id, name, description, unit_price, quantity, *comp_id*)**
- **Order( order_id, *cust_id, pro_id*, order_date, total_price)**

# Database: **student Management System**

- Department( dept_id, dept_name, dept_location)
- Teacher( teacher_id, name, phone, designation, salary, city, *dept_id*)
- Student( student_id, name, roll, phone, cgpa, city, *dept_id*)
- Courses( course_id, title, credit, description, *teacher_id*)
- Enroll_course( id, *course_id, student_id*, enroll_date)

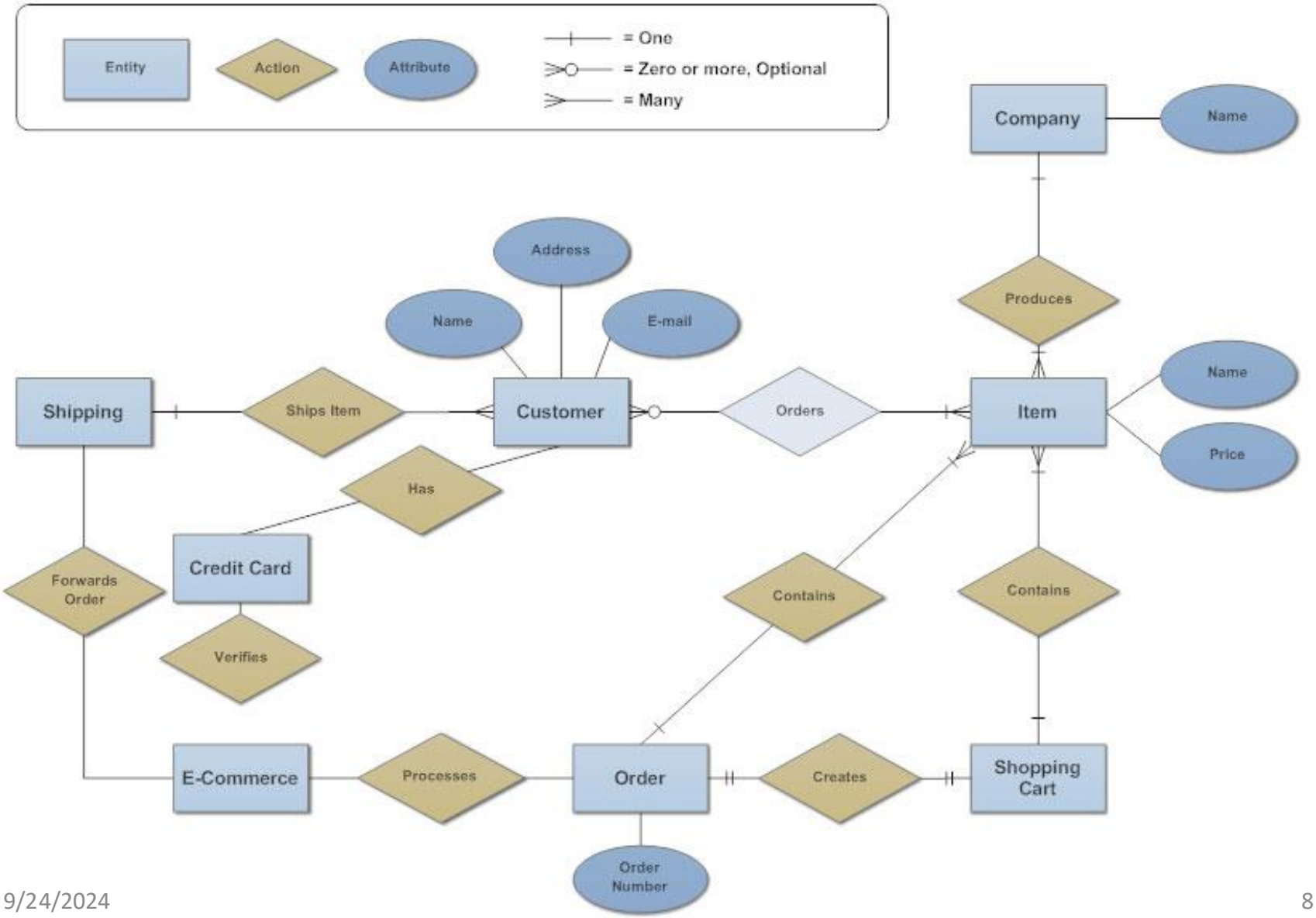
# Creating Database

- You have to design a database system for **Blood Bank Management** where the system will maintain the list of donors and recipients. It will keep blood history those are available in inventory. Anyone can send request for asking any blood.

# Creating Database

- You have to design a database system for **Blood Bank Management** that requires tables for managing donors, blood inventory, recipients, and requests.
  - The **Donors** table should store details like donor ID, name, blood type, contact information, and donation history.
  - The **Blood Inventory** table tracks blood units with fields for blood type, available units, and expiry dates.
  - A **Recipients** table records recipient information, such as name, blood type, contact details, and hospital affiliation.
  - A **Blood Requests** table manages requests with fields for recipient ID, requested blood type, units needed, request status, and request date.

# Entity Relationship Diagram - Internet Sales Model



# Entity Relationship Model

- **Entity**:
  - An **entity** is an object that exists and is distinguishable from other objects.
  - Example:  specific person, company, event, plant
- **Relationships**:
  - Correlations/Association among entities
  - Example: *employee **works_for** company, Instructor **teaches** students*

# Attributes

- An entity is represented by a set of attributes
- **Descriptive properties** possessed by all members of an entity set
- Types
  - Simple
  - Single Valued
  - Multi valued
  - Composite
  - Stored
  - Derived
  - ***Complex [Composite-multi valued]***
  - *Simple-single valued, simple-multivalued, single derived, single-composite*

# Attributes Details

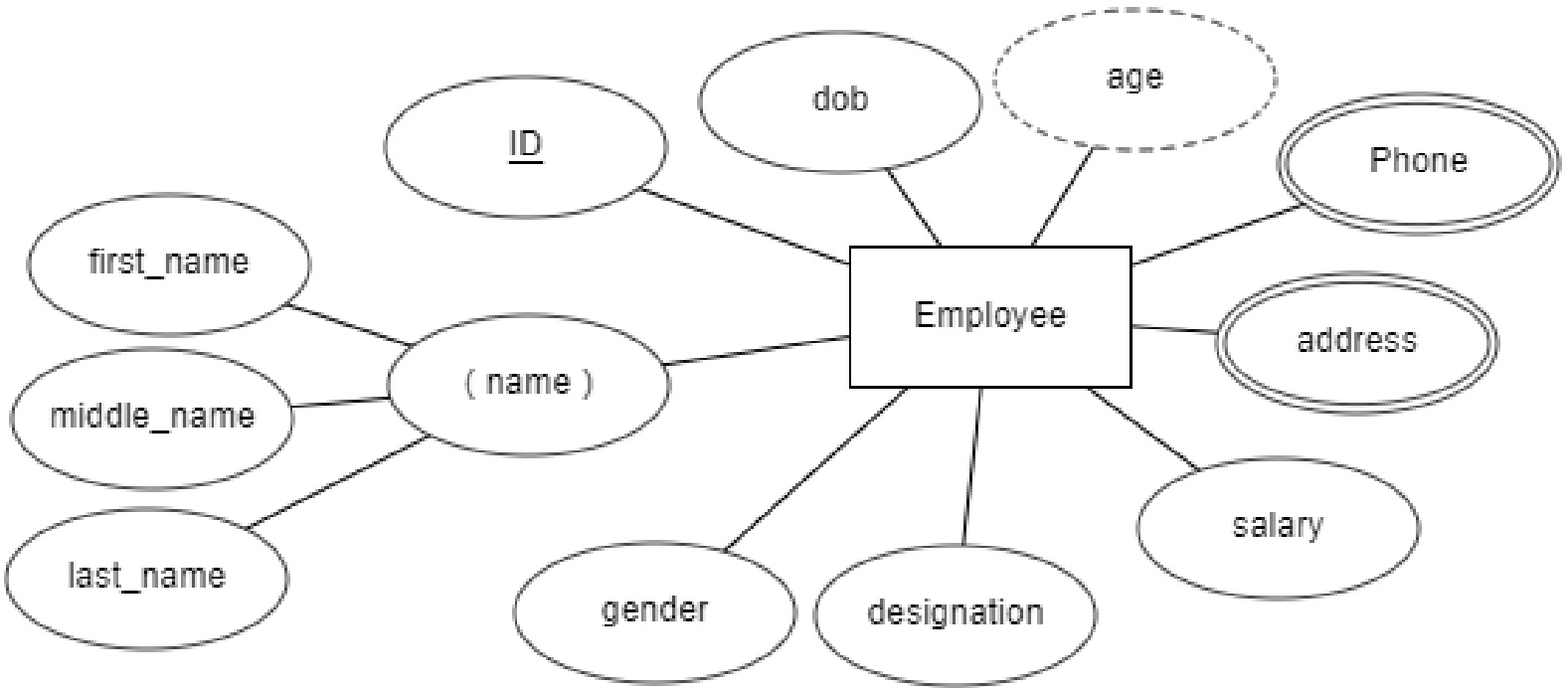
- Simple
  - Simple attributes are atomic values, which cannot be divided further
  - Example: Person(**id**, name, **phone**, address, age, dob, **email**)
- Single Valued
  - It holds only one value
  - Example: Person(**id**, name, phone, address, age, **dob**, email)
- **Multi-valued**
  - It may contain more than one values
    Example: phone, email

# Attributes Details

- **Derived**
  - Which can be derived from another attribute or set of attributes
  - Example: age→dob
- **Composite**
  - Composite attributes are made of more than one simple attribute
  - Example: full name→firstname, lastname
- ***Complex [Composite multi-valued]***
  - Combination of composite and multi-valued attributes
  - Example: Address
  - Present address (post code, district , country)
  - Permanent address (Same pattern)
- *Simple-single valued, simple-multivalued, single derived, single-composite*

# Representing Attributes with Entities

**Employees**(*id, name, email, phone, address, dob, gender, designation, salary*)



# Relationships

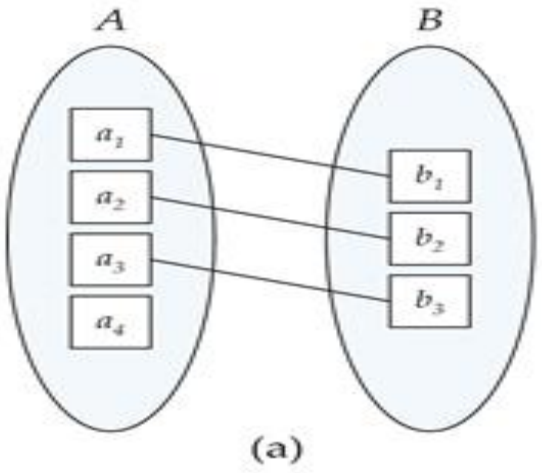
- Degree
  - Unary (1 Degree): CR ***inform*** Students
  - Binary (2 Degree): instructor ***teaches*** Student
  - Ternary (Three/More degree): instructor, Room, Students (***Teaches***)
- Types/ ***Mapping Cardinalities Constraints***
  - One to One
  - One to Many/ Many to One
  - Many to Many
  - Recursive Relationship
- Attributes for Relationship: *Employee works for department (**Start Date**)*

# Database: **student Management System**

- Department( dept_id, dept_name, dept_location)
- Teacher( teacher_id, name, phone, designation, salary, city, *dept_id*)
- Student( student_id, name, roll, phone, cgpa, city, *dept_id*)
- Courses( course_id, title, credit, description, *teacher_id*)
- Enroll_course( id, *course_id, student_id*, enroll_date)

# One to One

- One entity from entity set A can be associated with at most one entity of entity set B and vice versa.
- Example: Employee ***manages*** department

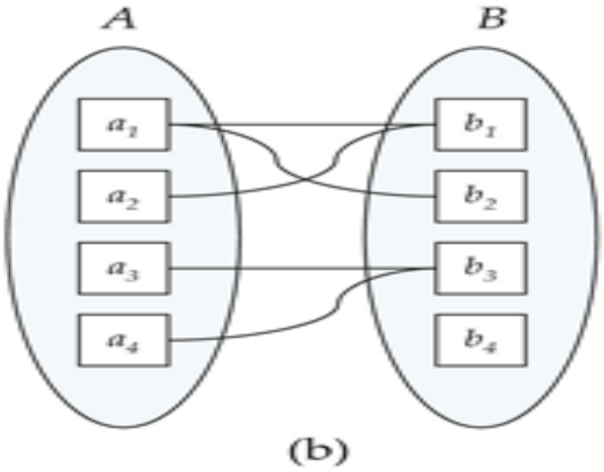

One to one

# One to Many

- One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.
- Example: Department ***Controls*** projects

# Many to Many

- One entity from A can be associated with more than one entity from B and vice versa.
- Example: employees ***works_on*** projects


Many to many

# END OF LECTURE