# CS354: Database

## Introduction

# Instructor & TAs

- Instructor
  - Dr. Samrat Mondal
  - Office: 405, Block 3, CSE Department
  - Email: samrat@iitp.ac.in
  - Phone: 8163
  - TA list for CS354
    - Sayantan, KM Pooja, Shashank
    - Email: {sayantan.pcs15, pooja.pcs16, shashank_1921cs04}
  - TA list for CS355
    - Suryakanta, Dipanjyoti, Sanghamitra, Soumitra,
    - Email: {suryakanta.pcs15, dipanjyoti.pcs17, 1821cs14@iitp.ac.in,...}

- Course Structure
  - Theory: CS354 (3-0-0-6)
  - Lab: CS355(0-0-3-3)
- Class Timings/ Venue
  - Wed: 10:00am - 10:55am/ R308
  - Thu: 10:00am – 10:55am/R108,
    Lab-2:00pm – 4:55pm/CSE Lab
  - Fri: 11:00am – 11:55am/R308
- Contact Hour
  - Mon: 6:00pm to 7:00pm or by appointment through email
- Course Link
  - Slides, lecture materials, assignments will be uploaded to 172.16.1.252/~samrat/CS354 and CS355

# Evaluation Policy

- CS354:
  - Internal: 30% (Assignment/Quiz, Class Performance)
  - MidSem: 30%
  - EndSem: 40%
- CS355:
  - Assignment, Projects, Viva: 60%
  - MidTerm Test: 20%
  - EndTerm Test: 20%

**75% attendance is mandatory for appearing for the end sem exam**

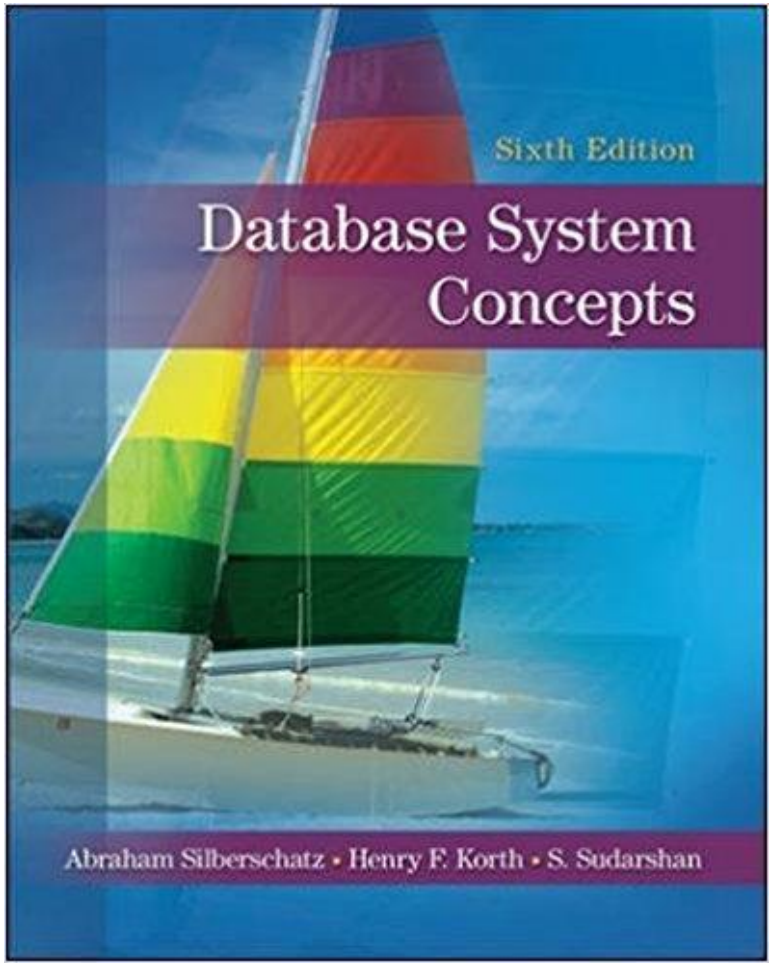
# Syllabus (1)

- **Database system architecture: Data Abstraction, Data Independence, Data Definition and Data Manipulation Languages;**
- **Data models: Entity-relationship, integrity constraints and data manipulation operations;**
- **Relational query languages: Relational algebra, tuple and domain relational calculus, SQL;**
- **Relational database design: Domain and data dependency, Armstrong's axioms, normal forms, dependency preservation, lossless design;**

# Syllabus (2)

- **Query processing and optimization: Evaluation of relational algebra expressions, query equivalence, join strategies, query optimization algorithms;**
- **Storage strategies: Indices, B-trees, hashing;**
- **Transaction processing: Recovery and concurrency control, locking and timestamp based schedulers;**
- **Advanced Topics (optional): SQL Injection Attack, Semi-structured Data Model**

# Text Book


Sixth Edition
Database System Concepts
Abraham Silberschatz • Henry F. Korth • S. Sudarshan

# Reference Books

- R. Ramakrishnan and J. Gehrke, Database Management Systems, 3rd Ed, McGraw Hill, 2003.
- R. Elmasri, S. B. Navathe and R. Sunderraman, *Fundamentals of Database Systems, 5th Ed, Pearson, 2009.*
- H. Garcia-Molina, J. D. Ullman and J. D. Widom, *Database Systems: The Complete Book, Prentice Hall, 2002.*

# Let's Begin

# Database

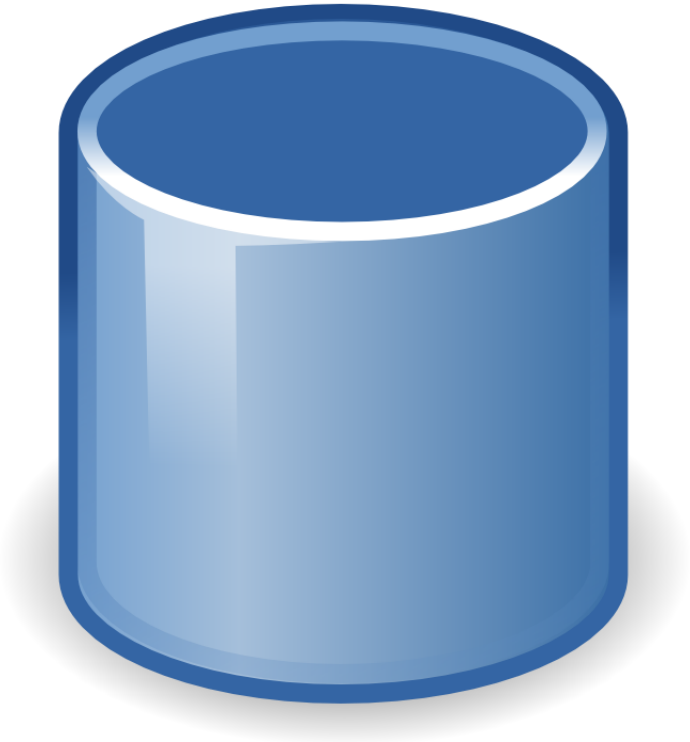
- A collection of interrelated data
- Usually designed to manage large bodies of information
- Models real world enterprise
  - Entities (e.g. student, courses)
  - Relationships (e.g. students are enrolled to courses)
- A ***database management system*** (DBMS) is a collection of interrelated data and a set of programs to access those data in a convenient and efficient way

# Some Representative Applications




# File System vs Database System

File System



Database System

# File Systems vs DBMS

- Data redundancy and inconsistency
- Difficulty in accessing data
- Data isolation
- Integrity problem
- Atomicity problem
- Concurrent access anomalies
- Security and access control

# Why use a DBMS

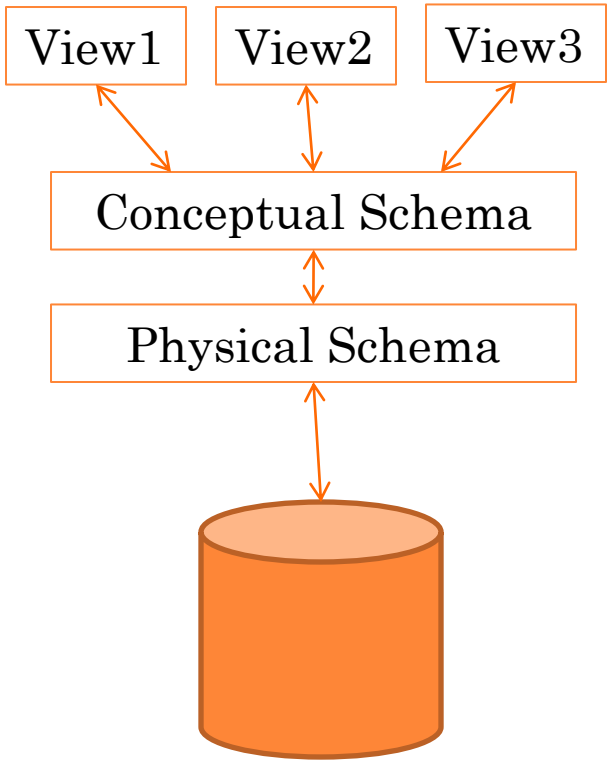
- Data independence and efficient access
- Reduced application development time
- Data integrity and security
- Uniform data administration
- Concurrent access, recovery from crashes

# View of Data

- A major purpose of database system is to provide users with an ***abstract view*** of the data
- The data from the database must be retrieved efficiently
- This need has led designers to use complex data structures
- Since many users are not computer trained
  - So developers hide the complexity from users through several levels of abstraction

# Levels of Abstraction

- Many *views*, single *conceptual (logical) schema* and *physical schema*.
- Views describe how different users see the database
- Conceptual schema describes what data are stored and what is the relationships exist among those data
- Physical schema describes how the data are actually stored



# Example: University Database

- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- Conceptual schema:
  - *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*
- External Schema (View):
  - *Course_info(cid:string, sid: string)*

# Data Independence

- Applications insulated from how data is structured and stored
- *Logical data independence*: Protection from changes in *logical* structure of data
- *Physical data independence*: Protection from changes in *physical* structure of data
- *Data Independence is one of the most important benefits of using a DBMS*

# Instances and Schemas

- Instance of the database: the collection of information stored in the database at a particular moment
- Database schema: the overall design of the database

# Data Models

- Underlying the structure of the database is the *data model*
  - It is a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints
  - The *relational model of data* is the most widely used model today
  - Main concept: *relation*, basically a table with rows and columns
  - Every relation has a *schema*, which describes the columns, or fields.

- Example of *customer* relation

| *customer_id* | *customer_name* | *customer_street* | *customer_city* | *account_number* |
|---|---|---|---|---|
| 192-83-7465 | Johnson | 12 Alma St. | Palo Alto | A-101 |
| 192-83-7465 | Johnson | 12 Alma St. | Palo Alto | A-201 |
| 677-89-9011 | Hayes | 3 Main St. | Harrison | A-102 |
| 182-73-6091 | Turner | 123 Putnam St. | Stamford | A-305 |
| 321-12-3123 | Jones | 100 Main St. | Harrison | A-217 |
| 336-66-9999 | Lindsay | 175 Park Ave. | Pittsfield | A-222 |
| 019-28-3746 | Smith | 72 North St. | Rye | A-201 |

# Database Languages

- A database system provides –
  - DDL (Data Definition Language): to specify the database schema
  - DML (Data Manipulation Language): to express database queries and updates

These two form parts of a single database language such as the widely used SQL (Structured Query Language)

# DDL Example

**create table** *account* (
*account_number* **char**(10),
*branch_name* **char**(10),
*balance* **integer**)

# DML

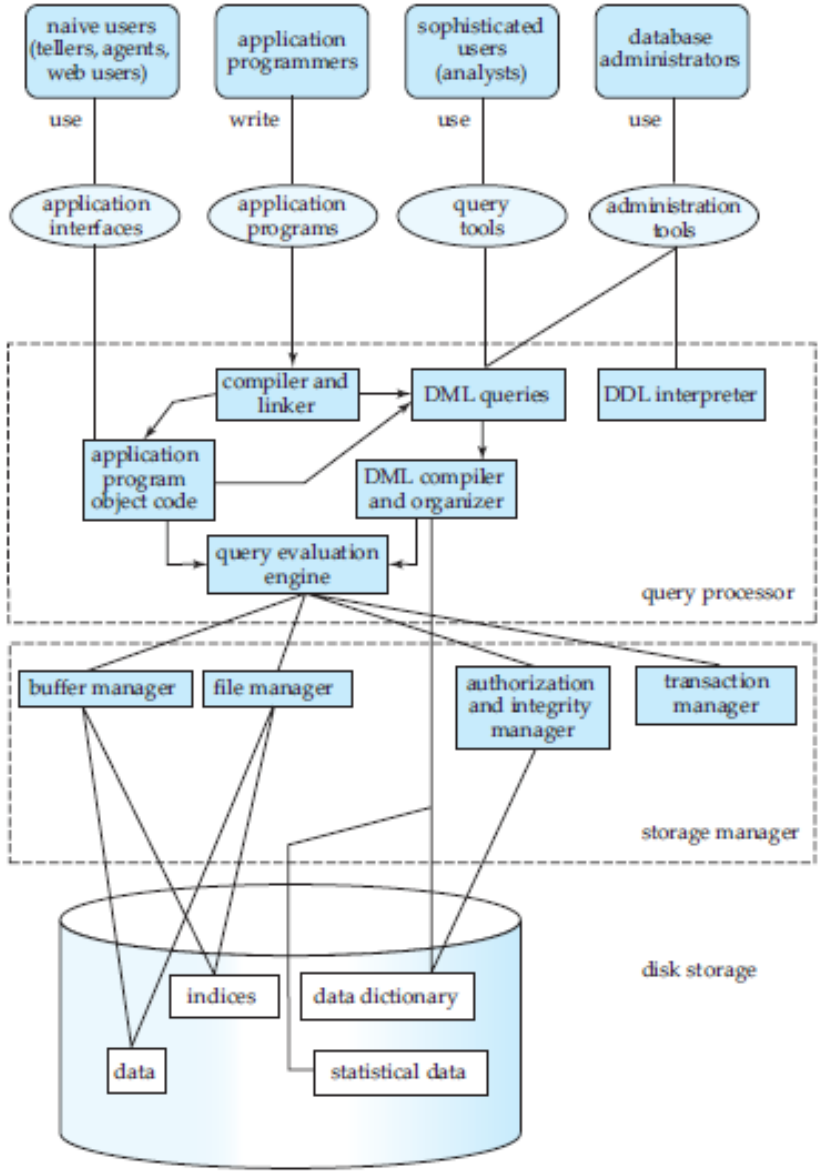
- Data manipulation may be-
  - the retrieval of the information stored in the database
  - the insertion of the new information in the database
  - the deletion of the information in the database
  - the modification of the information stored in the database
- Example

  **select** customer_name

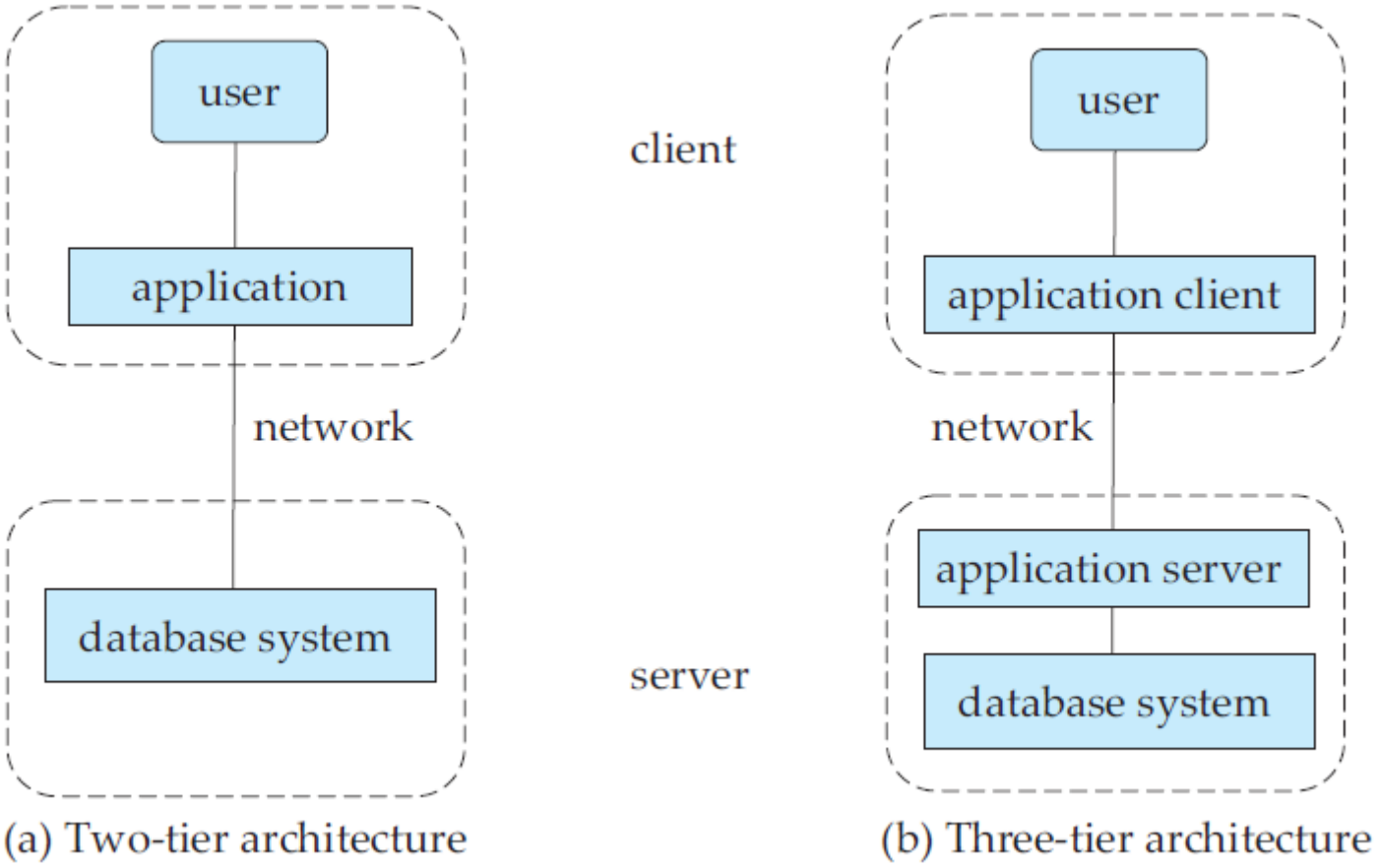
  **from** customer

  **where** customer_city = “Palo Alto”

# Database Architecture



# Two and Three Tier Architectures




(a) Two-tier architecture

(b) Three-tier architecture

# Entity Relationship Model

- Widely used conceptual level data model
  - proposed by Peter P Chen in 1970s
- The ER model is one of the most cited articles in Computer Science
  - *"The Entity-Relationship model – toward a unified view of data"* Peter Chen, 1976

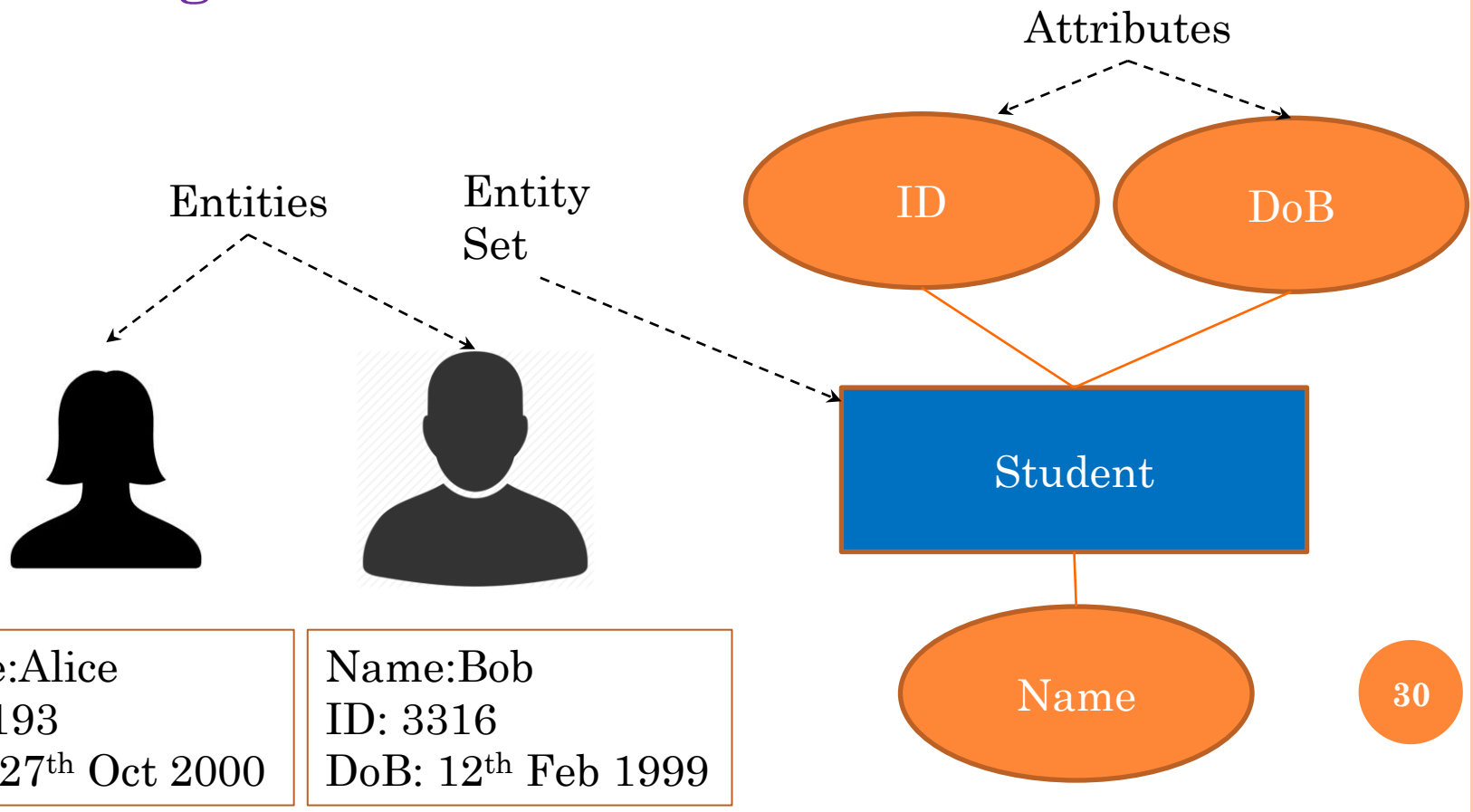
- Data model to describe the database system at the requirements collection stage
  - high level description
  - easy to understand for the enterprise managers
  - rigorous enough to be used for system building
- Concepts available in the model
  - entities and attributes of entities
  - relationships between entities
  - diagrammatic notation

# Entities

- **Entity:**
  - Real-world object distinguishable from other objects
  - An entity is described (in DB) using a set of attributes
    - In the University database context, an individual student, faculty member, a class room, a course, etc. are entities
- **Entity Set or Entity Type-**
  - Collection of entities all having the same properties.
  - Student entity set –collection of all student entities.
  - Course entity set –collection of all course entities.

# Entity vs Entity Set

- **Entities are not explicitly represented in ER Diagram**



# Attribute

- Each entity is described by a set of attributes/properties.
- Student entity
  - StudName–name of the student.
  - ID–the unique ID given to each student.
  - Sex–the gender of the student etc.
- All entities in an Entity set/type have the same set of attributes.

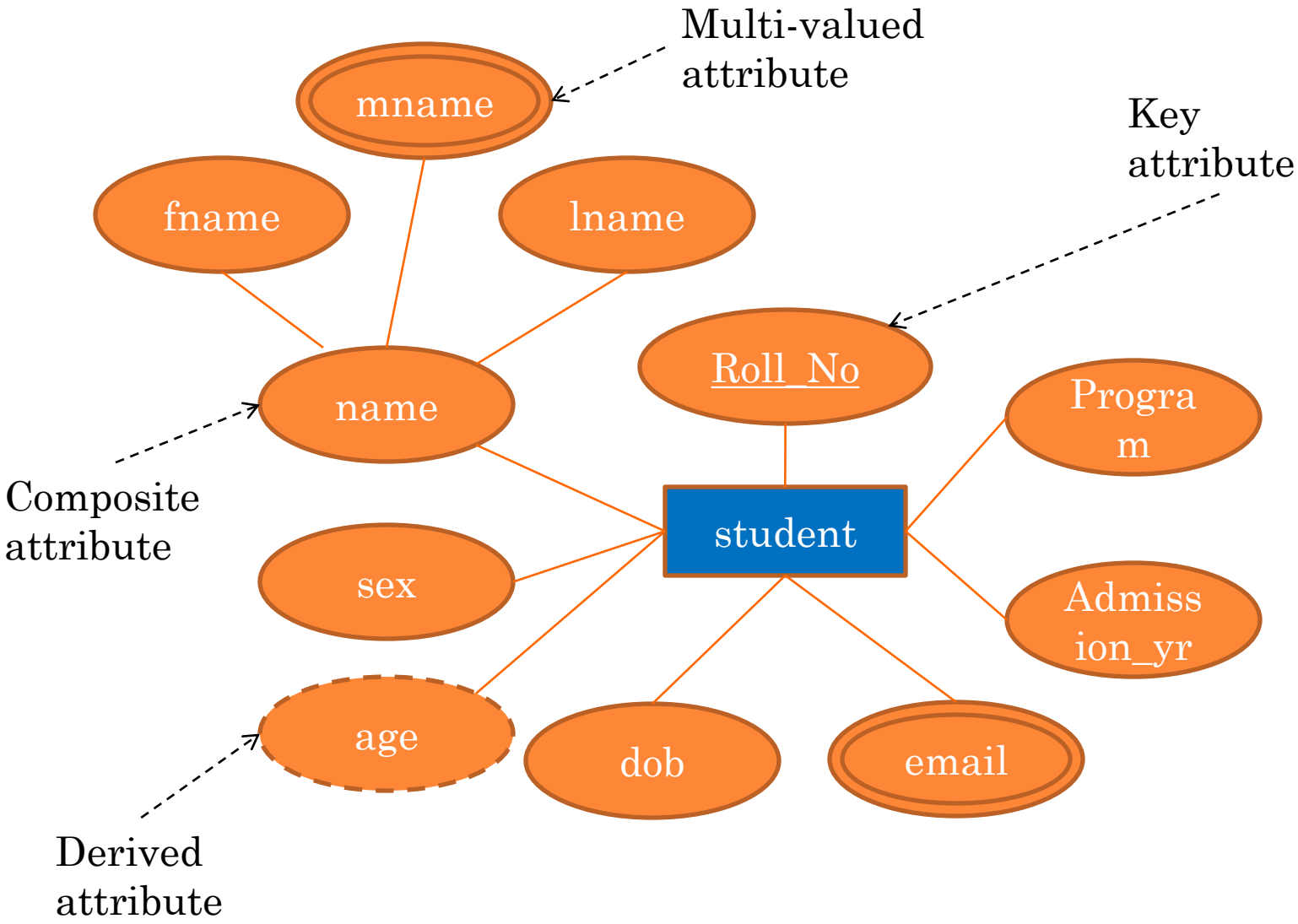
# Types of Attributes

- Simple Attributes
  - having atomic or indivisible values.
  - E.g. Dept–a string
  - PhoneNumber–a ten digit number
- Composite Attributes
  - having several components in the value.
  - E.g. Qualification with components
  - (DegreeName, Year, UniversityName)
- Derived Attributes
  - Attribute value is dependent on some other attribute.
  - E.g: Age depends on DateOfBirth. So age is a derived attribute.

# Types of Attributes (2)

- Single-valued
  - having only one value rather than a set of values.
  - E.g., PlaceOfBirth–single string value.
- Multi-valued
  - having a set of values rather than a single value.
  - E.g., CoursesEnrolled attribute for student
  - EmailAddress attribute for student
  - PreviousDegree attribute for student.
- Attributes can be:
  - simple single-valued, simple multi-valued,
  - composite single-valued or composite multi-valued.

# ER Diagram: Notations



# Domains of Attributes

- Each attribute takes values from a set called its *domain*
- For example,
  - StudentAge–{17,18, …, 55}
  - HomeAddress–character strings of length 35
- Domain of composite attributes –
  - cross product of domains of component attributes
- Domain of multi-valued attributes –
  - set of subsets of values from the basic domain

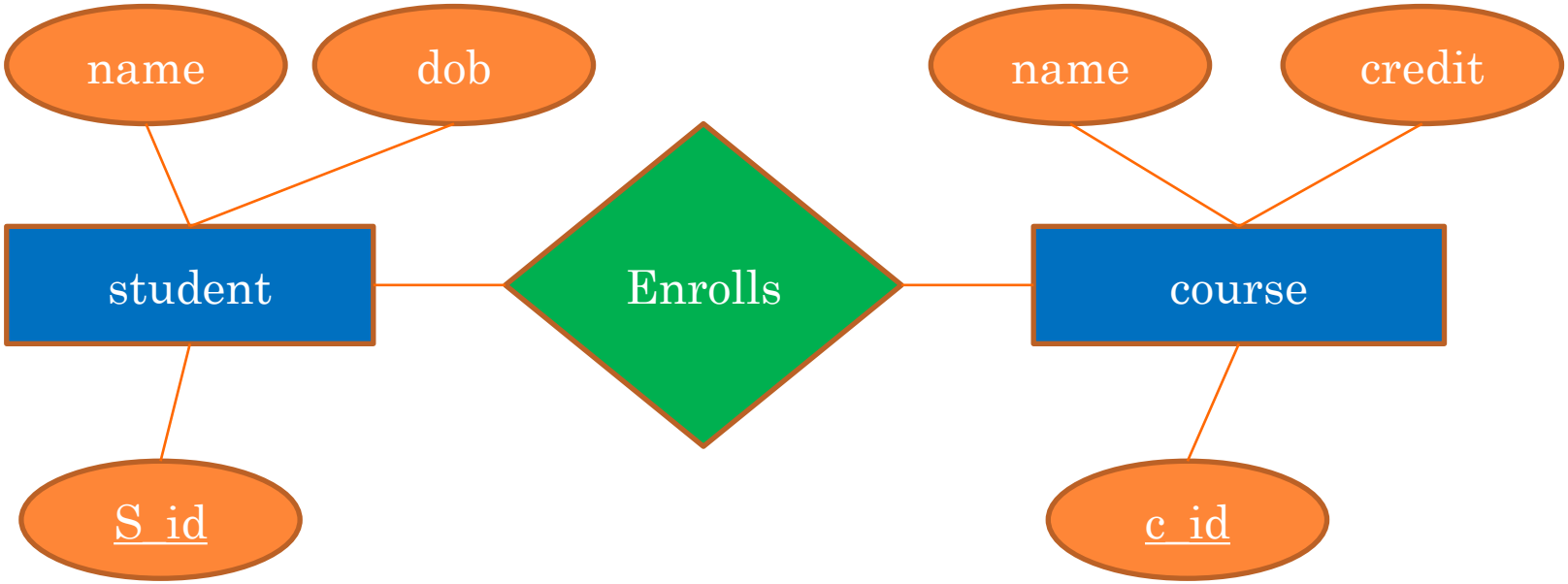
# Entity sets and key attributes

- Key–an attribute or a collection of attributes whose value(s) uniquely identify an entity in the entity set
- For instance,
  - RollNumber- Key for Student entity set
  - EmpID- Key for Faculty entity set
  - HostelName, RoomNo- Key for Student entity set (assuming that each student gets to stay in a single room)
- A key for an entity set may have more than one attribute
- An entity set may have more than one key
- Determined by the designers

# Relationship

When two or more entities are associated with each other, we have an instance of a *relationship*

E.g: *student* Alice *enrolls* in Discrete Mathematics *course*

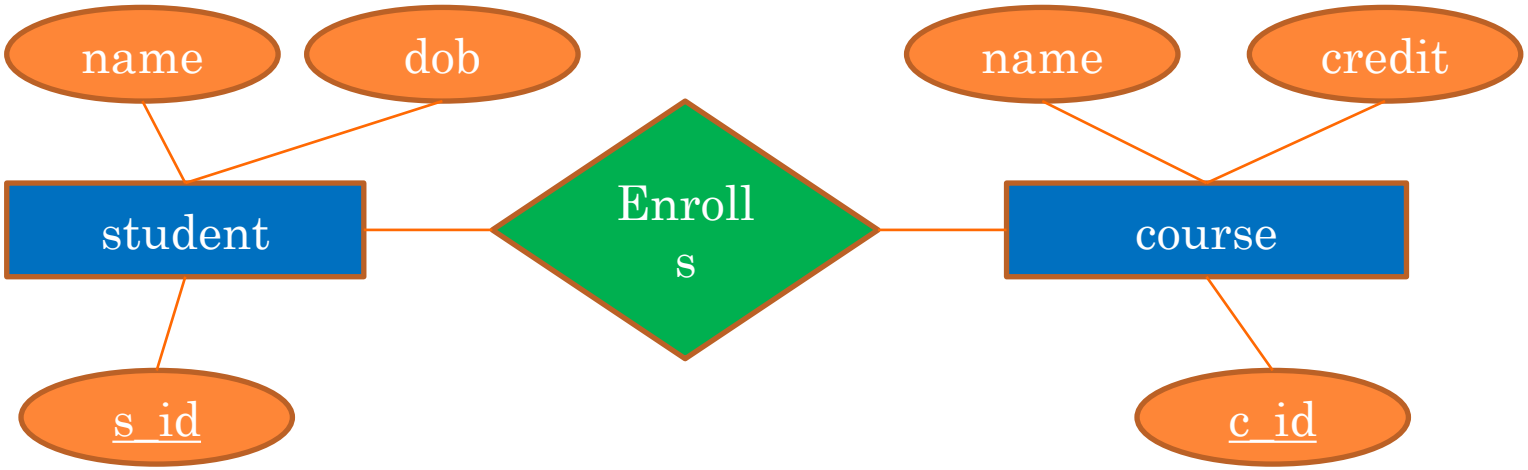


# Mathematical Interpretation

- Relationship *Enrolls* has *Student* and *Course* as the participating entity sets
- Formally, Enrolls ⊆ Student × Course
  - Operation 'x' indicates cross product
- (s,c) ∈ Enrolls ⇔ Student 's' has enrolled in Course 'c'
- Tuples in *Enrolls* known as relationship instances
- *Enrolls* is called a relationship Type/Set

# KEYS OF RELATIONSHIP

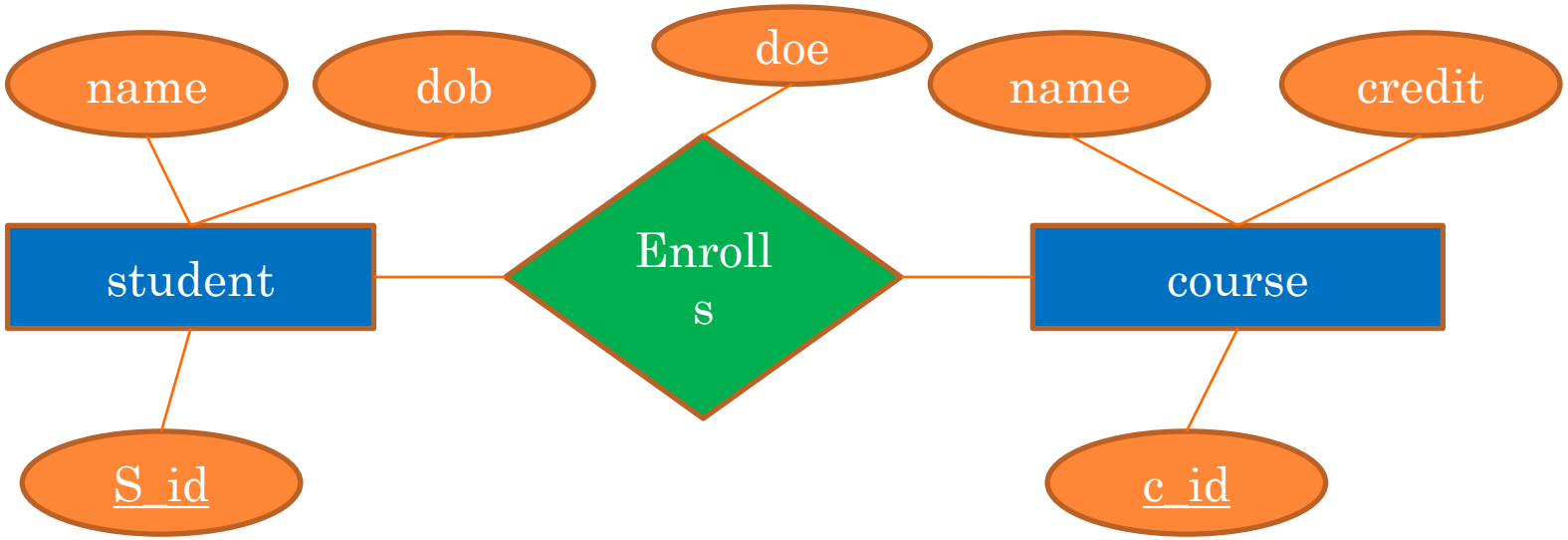
- There can only be **one relationship for every unique combination of entities**
- This also means that **the relationship is uniquely determined by the keys of its entities**
- *Example: the "key" for Enrolls is {s_id, c_id}*




$$Key_{Enrolls} = Key_{student} \cup Key_{course}$$

# Relationships and Attributes

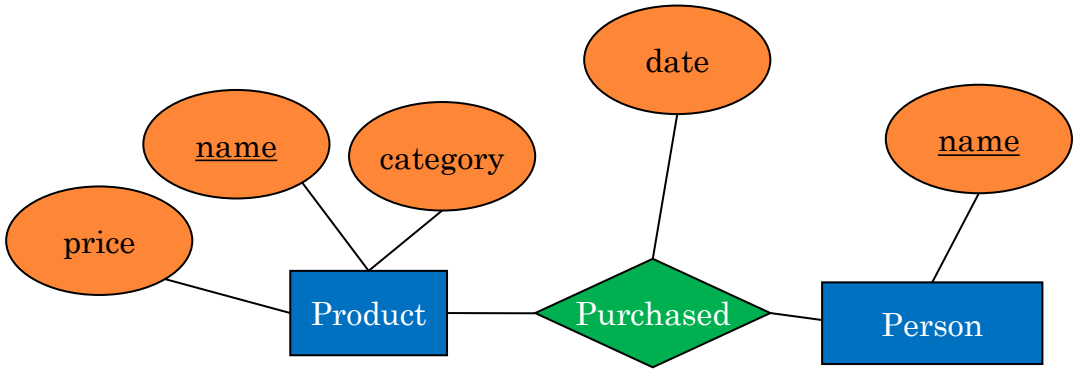
- Relationships may have attributes as well.




For example: “doe” or date of enrollment records when a student enrolled for the course. “doe” is neither an attribute of student nor course

# Decision: Relationship vs. Entity?

- **Q:** What does this say?

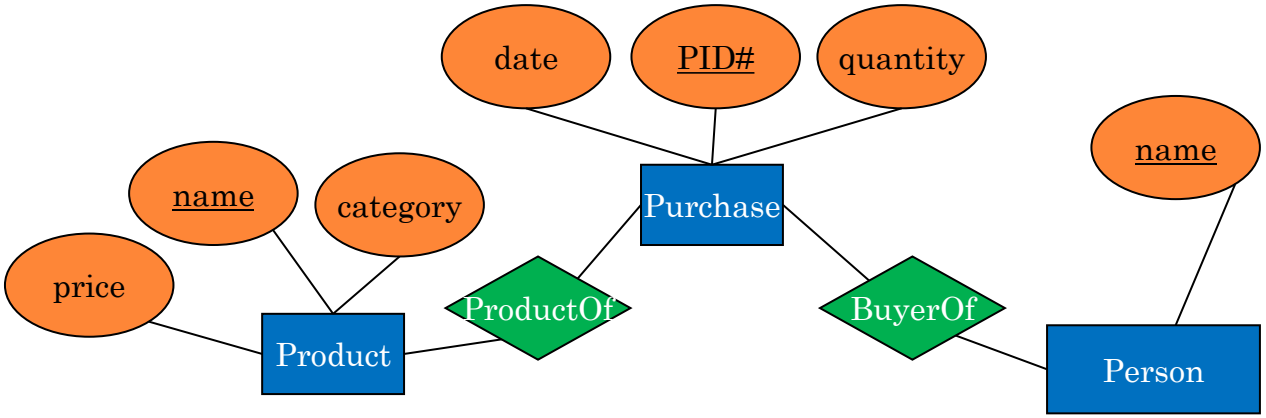



- **A:** A person can only buy a specific product once (on one date)

Modeling something as a relationship makes it unique; what if not appropriate?

# Decision: Relationship vs. Entity?

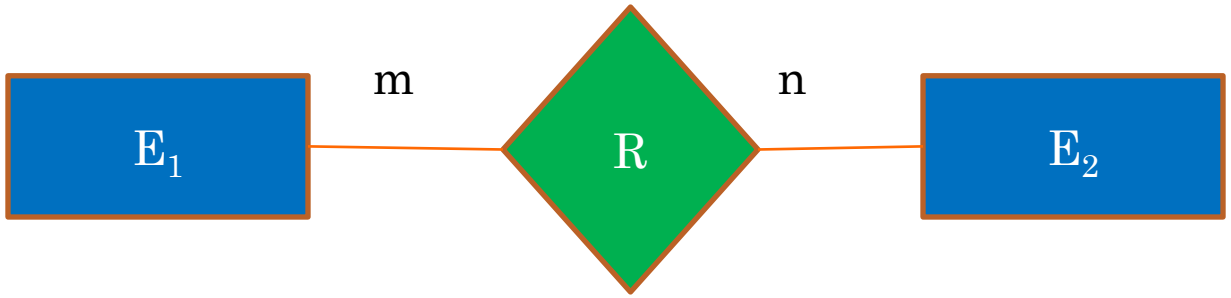
- What about this way?




- *Now we can have multiple purchases per product, person pair!*

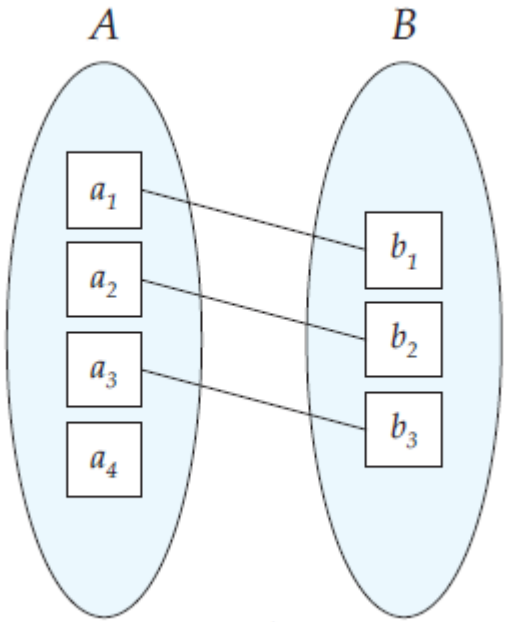
We can always use **a new entity** instead of a relationship. For example, to permit multiple instances of each entity combination!

# Binary Relation & Cardinality

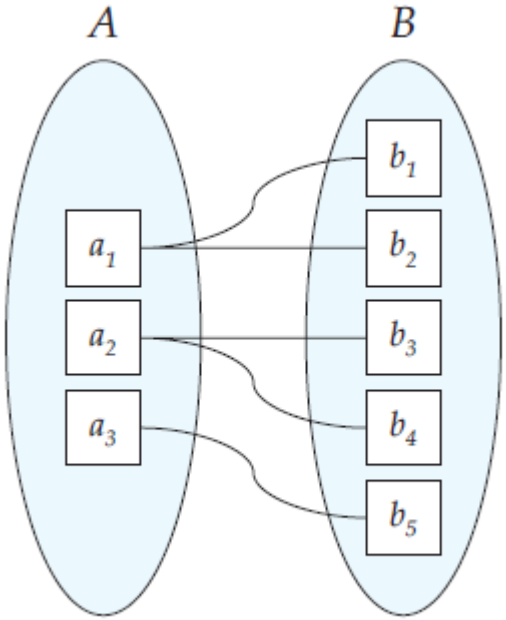



The number of entities from E2 that an entity from E1 can possibly be associated through R (and vice-versa) determines the cardinality ratio of R.

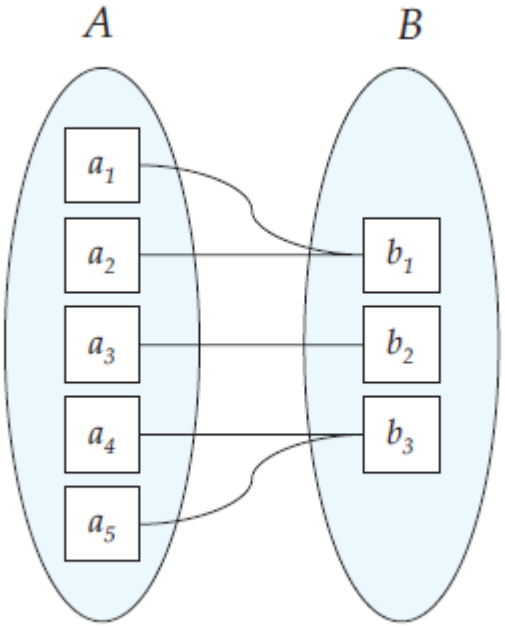
Four possibilities-
one to one, one to many, many to one and many to many



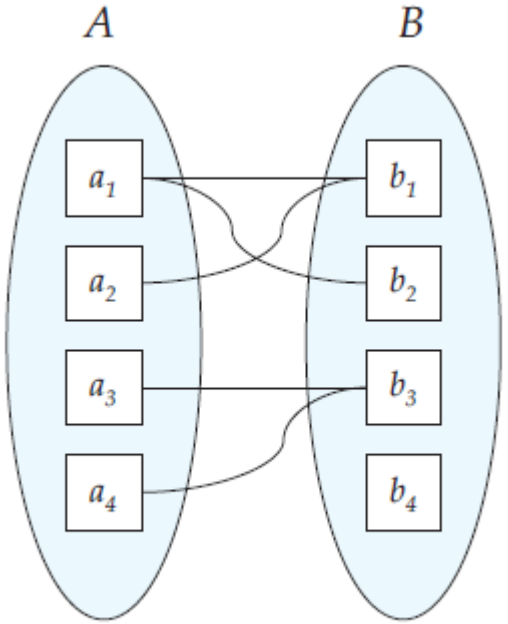

One to One




One to Many

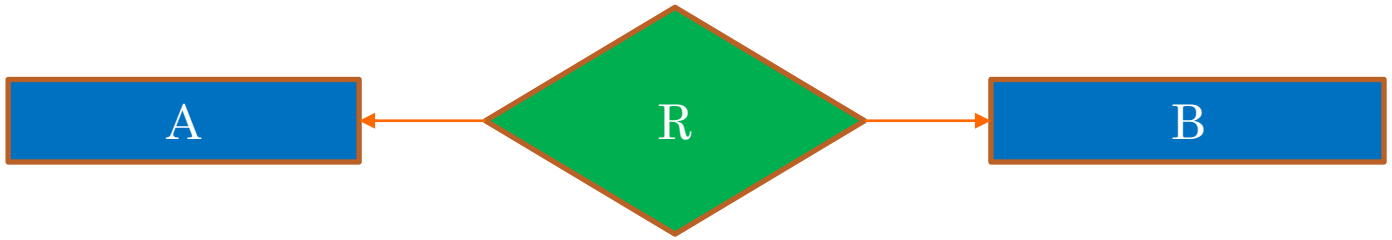



Many to One

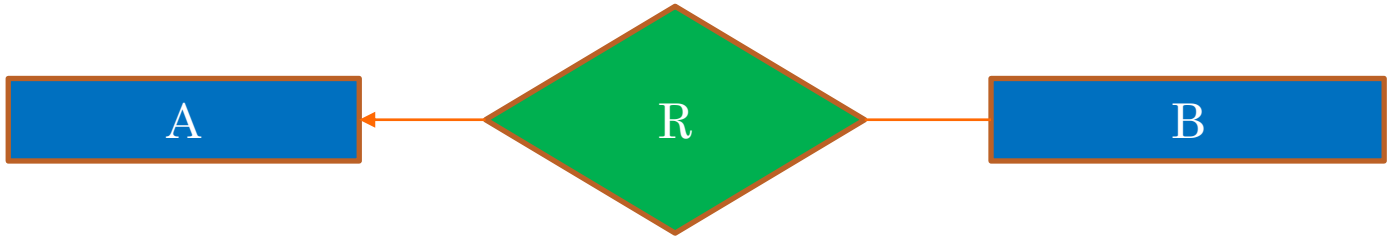



Many to Many

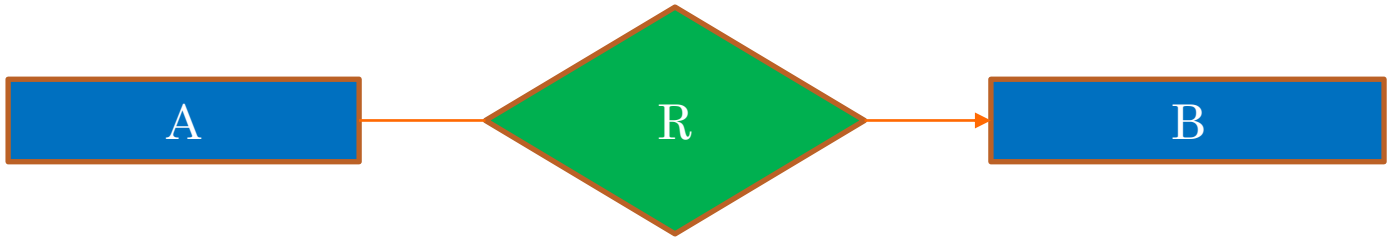
One to One



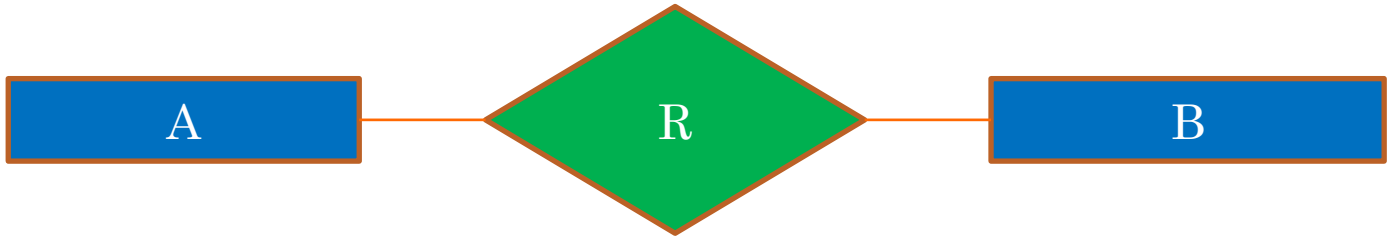

One to Many



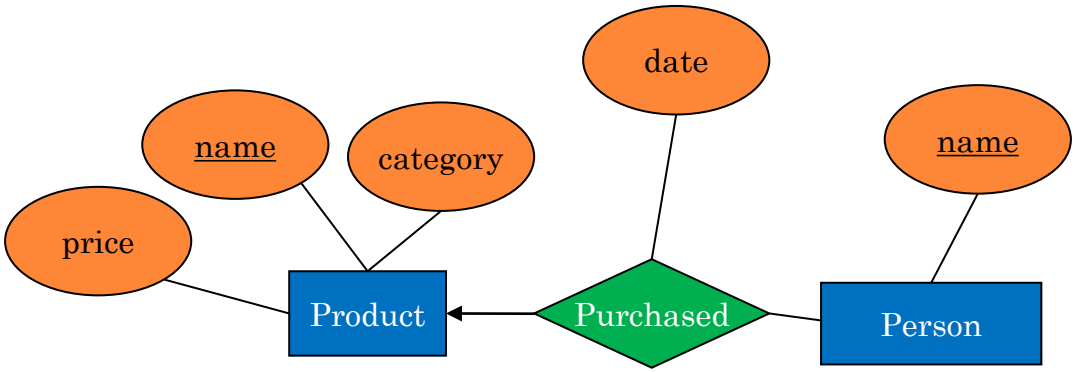

Many to One




Many to Many

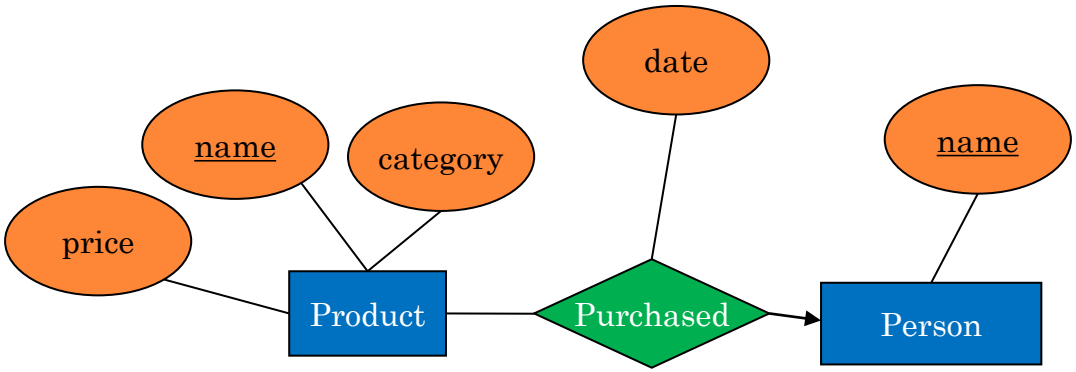


# EXAMPLE: ONE TO MANY

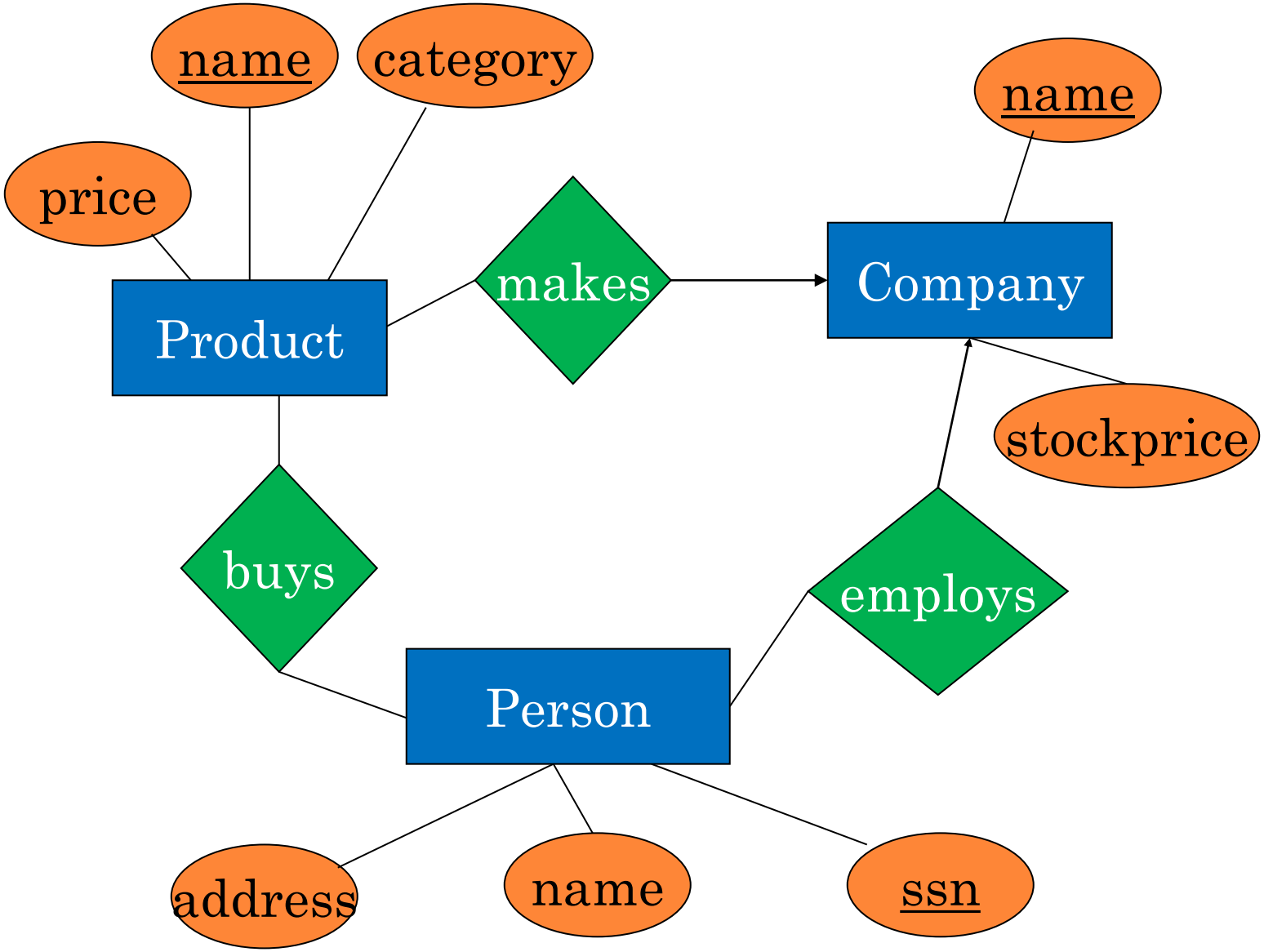



One person can be associated with atmost one product and one product can be associated with any (0 or more) number of persons through Purchased relationship

# Example: Many to One




One product can be associated with atmost one person and one person can be associated with any (0 or more) number of products through Purchased relationship






What does this say?

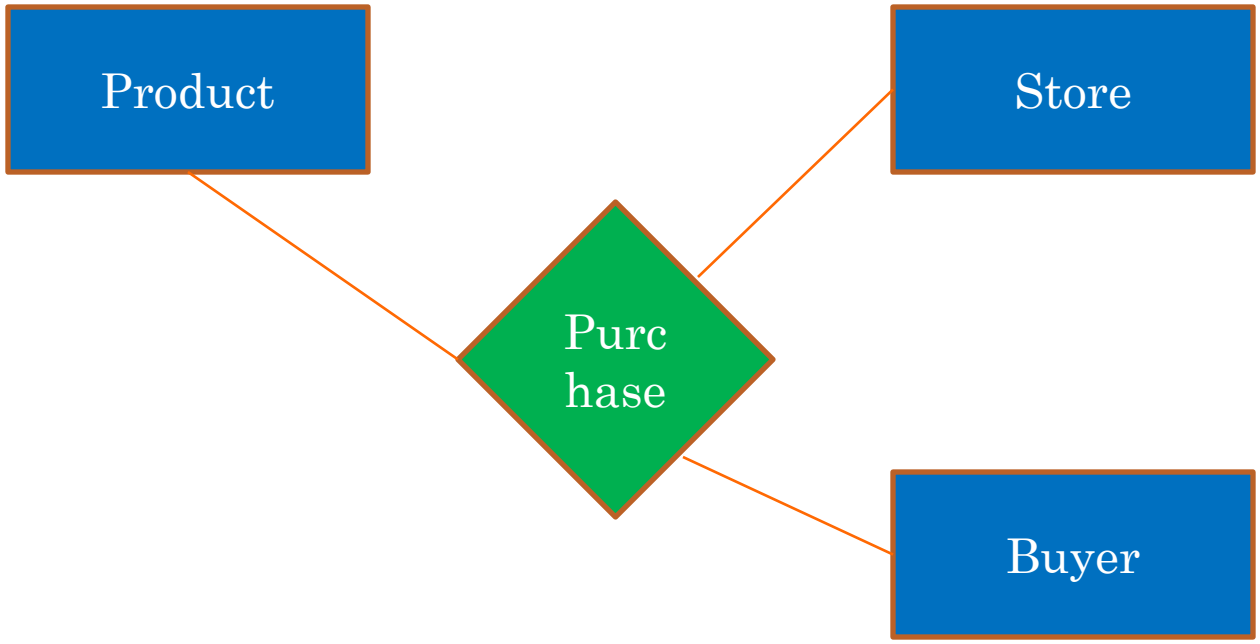
# Degree of a relationship

- Degree: the number of participating entities
  - Degree 2: binary
  - Degree 3: ternary
  - Degree n: n-ary

  Binary relationships are very common and widely used

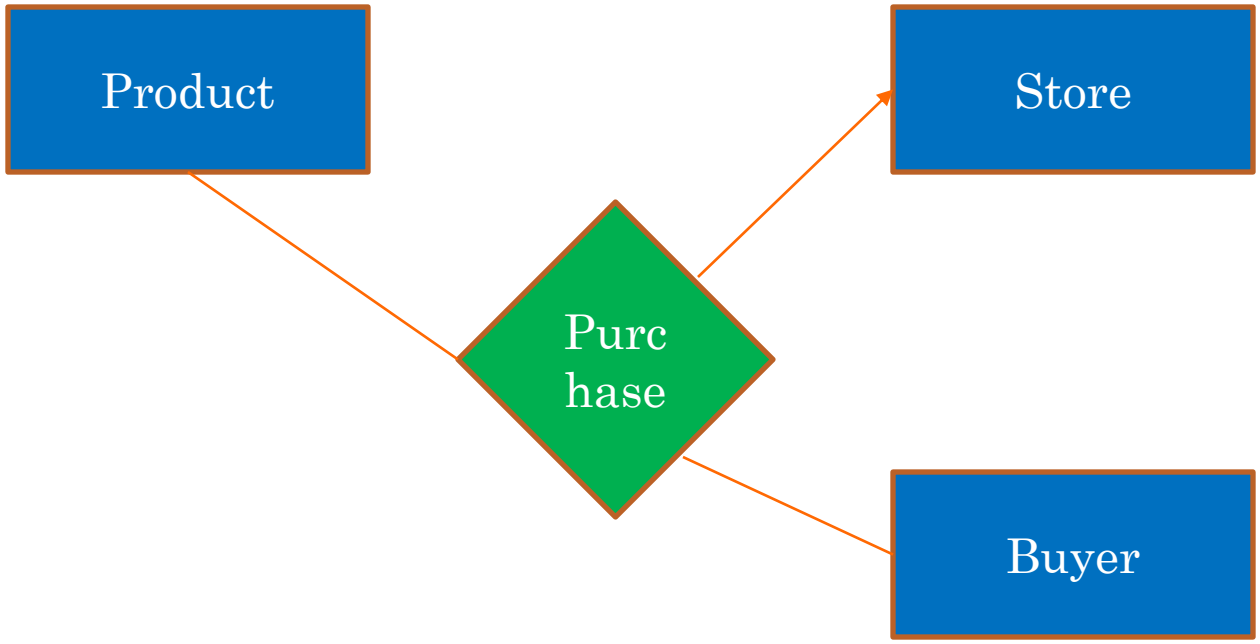
# Multi-way Relationship

Modeling a purchase relationship between product, store and buyers



# Multi-way Relationship

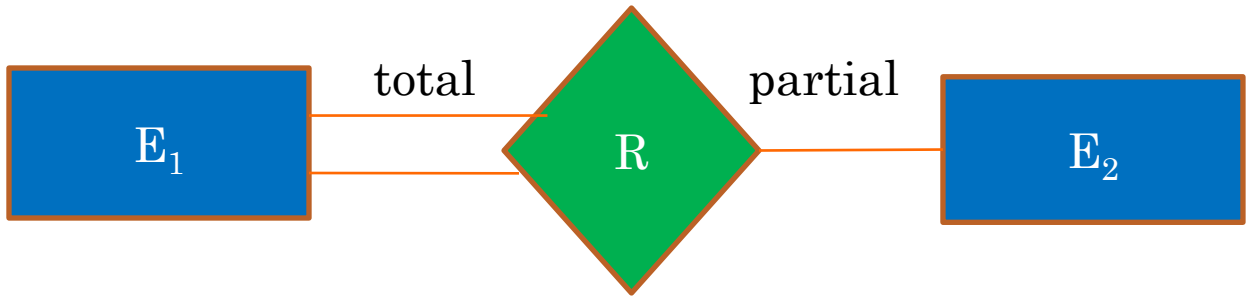
What is the meaning of the following relationship?




For each unique combination of product and buyer, there will be atmost one store associated

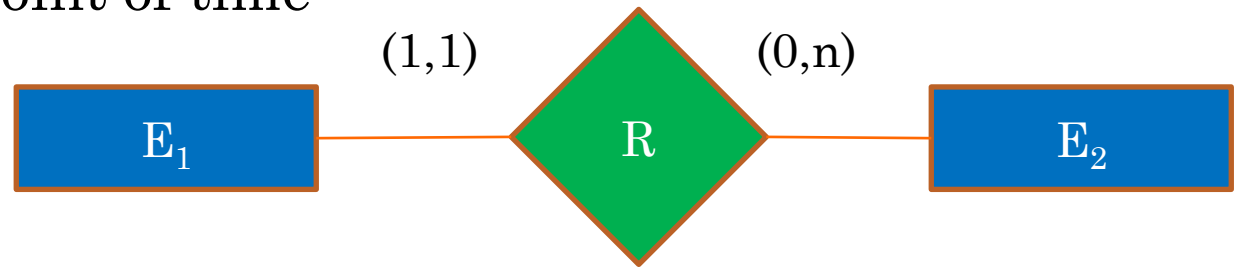
# Participation Constraint

- An entity set may participate in a relation either ***totally*** or ***partially***
- Total participation: Every entity in the set is involved in some association (or tuple) of the relationship
- Partial participation: Not all entities in the set are involved in association (or tuples) of the relationship



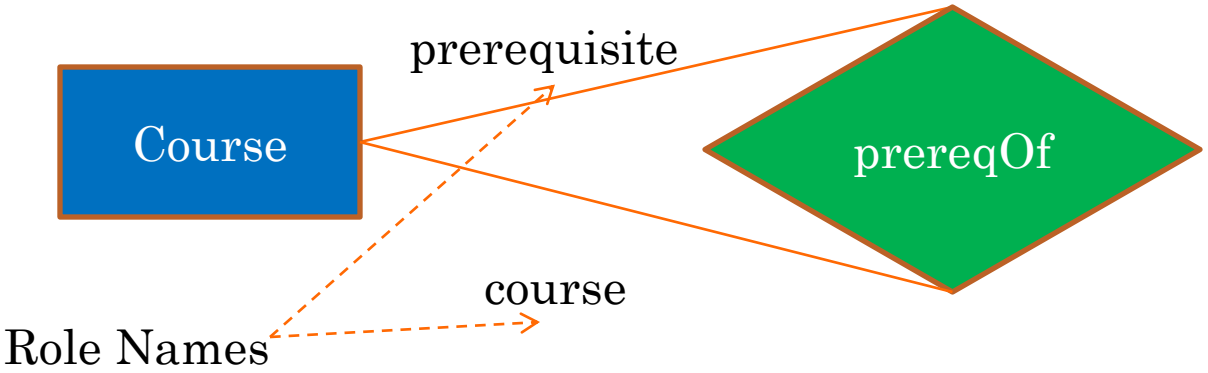
# Structural Constraints

- Cardinality Ratio and Participation Constraints are together called ***Structural Constraints***
- They are called constraints as the data must satisfy them to be consistent with the requirements
- Min-Max notation: pair of numbers (m,n) placed on the line connecting an entity to the relationship
- m: the minimum number of times a particular entity must appear in the relationship tuples at any point of time
  - 0 –partial participation
  - ≥1 –total participation
- n: similarly, the maximum number of times a particular entity can appear in the relationship tuples at any point of time



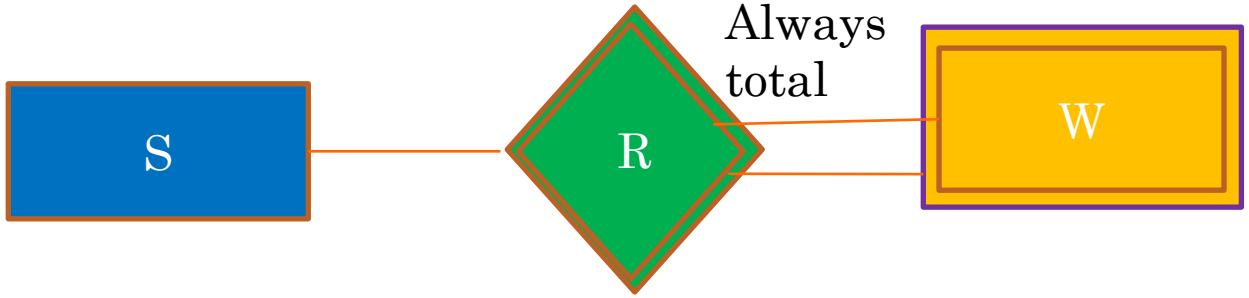
# Recursive relationship and Role Name

- Recursive relationship: An entity set relating to itself gives rise to a recursive relationship
  - E.g., the relationship prereqOf is an example of a recursive relationship on the entity Course
- Role Names –used to specify the exact role in which the entity participates in the relationships
- Role Names are essential in case of recursive relationships



# Weak Entity Set

- Weak Entity Set: An entity set whose members owe their existence to some entity in a strong entity set
  - Entities are not of independent existence
  - Each weak entity is associated with some entity of the owner entity set through a special relationship
  - Weak entity set may not have a key attribute
  - The **discriminator** (*or partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set

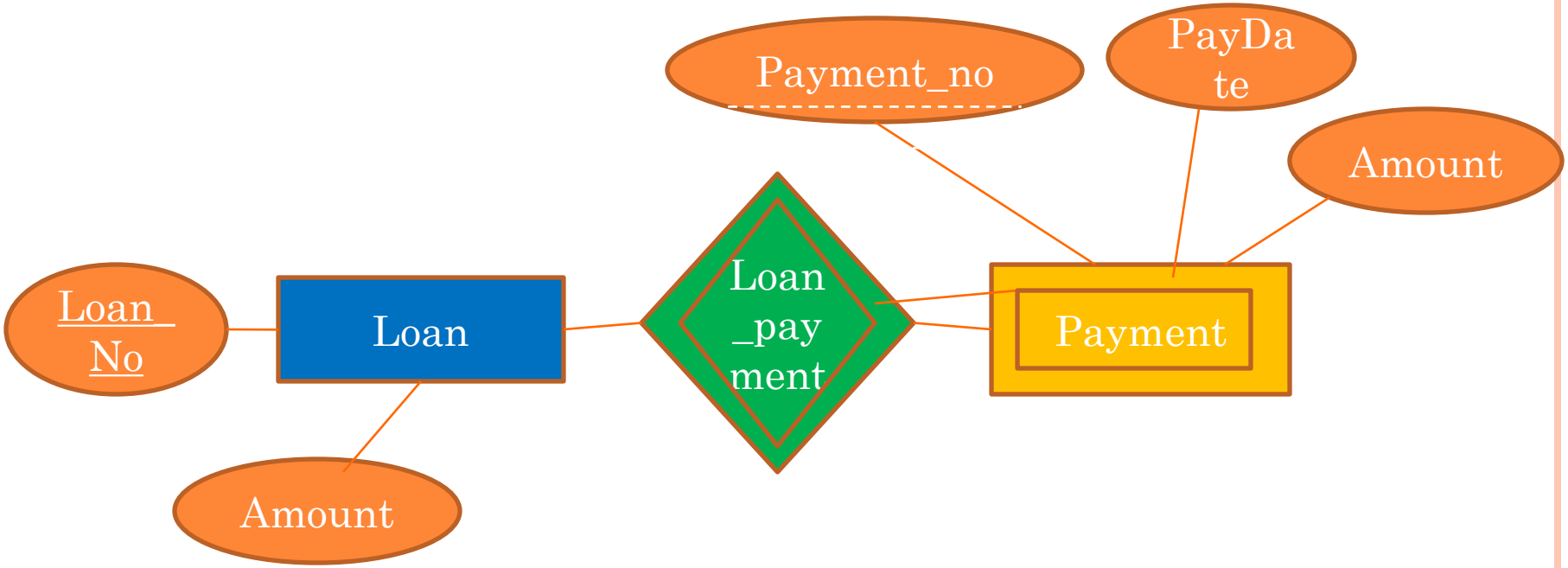



Owner Entity

Identifying Relationship

Weak entity

# Weak Entity Set Example



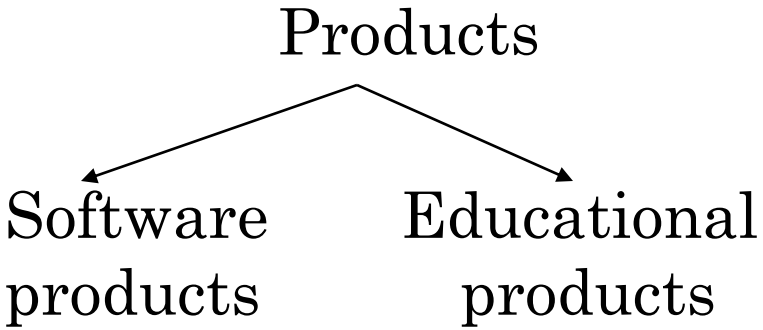
# Modeling Subgroupings

- Some entities in a entity set may be special, i.e. worthy of their own entity set

  Define a new entity set?

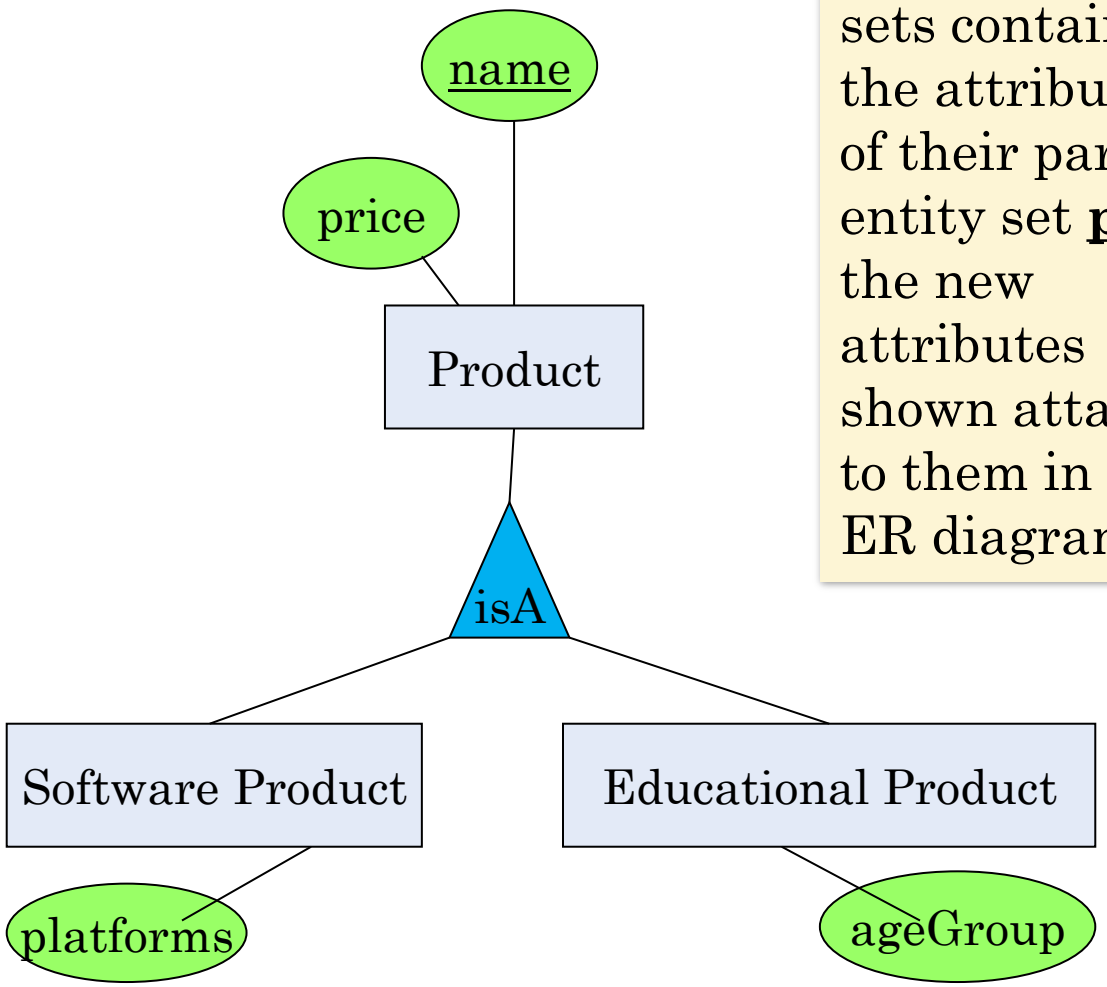
  - *But what if we want to maintain connection to current entity set?*

- Better: define a *sub-entity set*
  - *Ex:*




We can define **subgroups** in ER Diagram

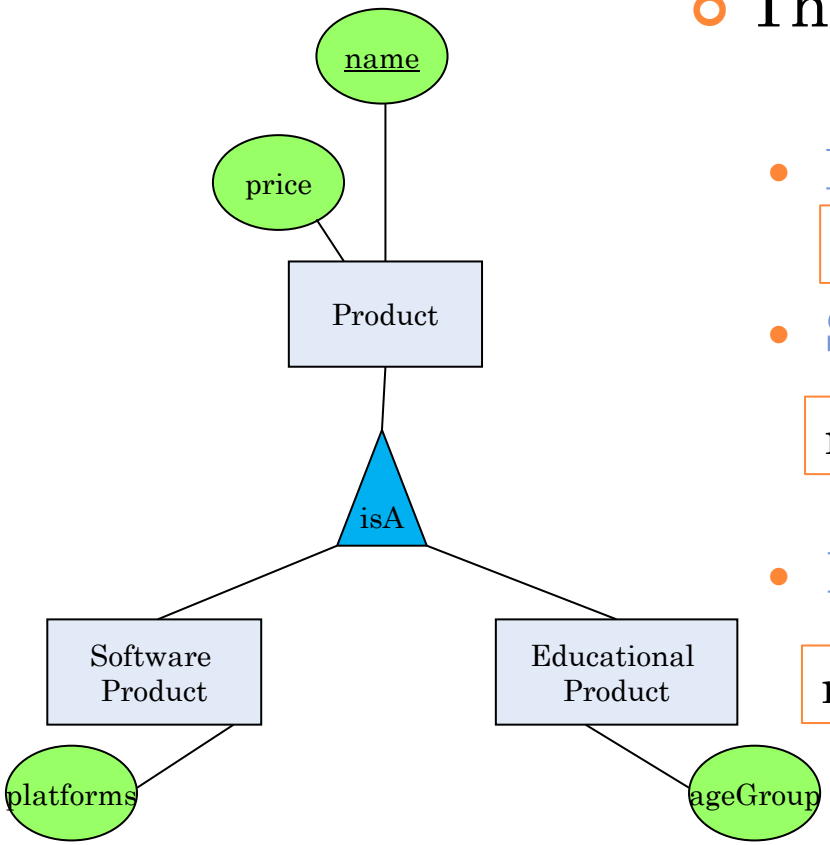
# Modeling Subgroups




Child entity sets contain all the attributes of their parent entity set **plus** the new attributes shown attached to them in the ER diagram

# Understanding Subgroups

Child subgroups contain all the attributes of *all* of their parent groups **plus** the new attributes shown attached to them in the ER diagram




- Think in terms of records; ex:
  - Product

    | name | price |
    |---|---|
  - SoftwareProduct

    | name | price | platform |
    |---|---|---|
  - EducationalProduct

    | name | price | ageGroup |
    |---|---|---|

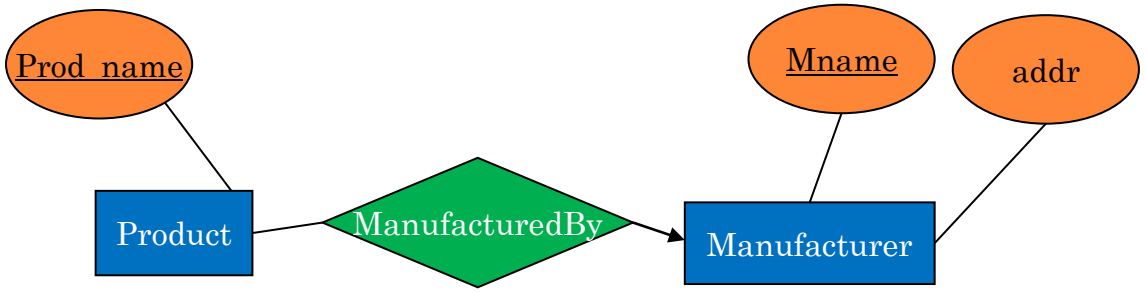
# Some Design Tips

- Avoid redundancy
- Limit the use of weak entity sets
- Don’t use an entity set when an attribute will do

# Avoiding Redundancy

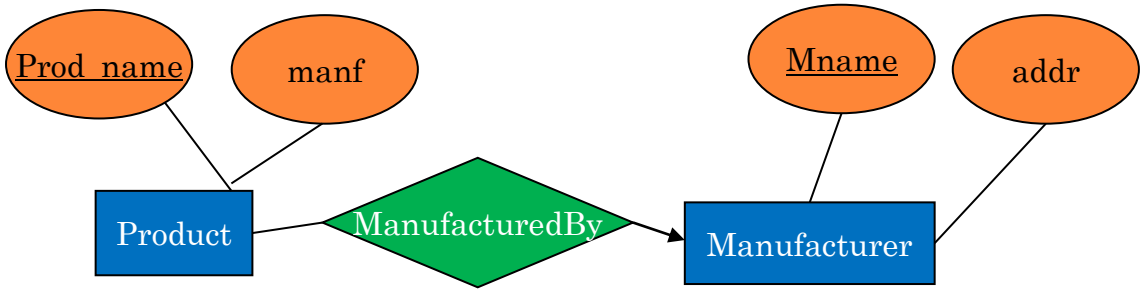
- Redundancy = saying the same thing in two (or more) different ways.
- Wastes space and (more importantly) encourages inconsistency.
- Two representations of the same fact become inconsistent if we change one and forget to change the other.

# EXAMPLE: GOOD



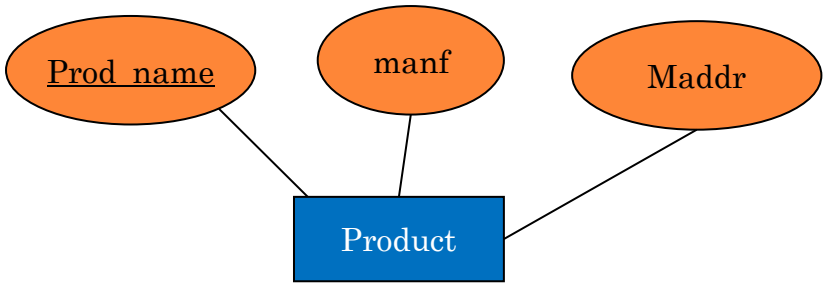

This design gives the address of each manufacturer exactly once.

# Example: Bad




This design states the manufacturer of a product twice- as an attribute and as a related entity

# Example: Bad

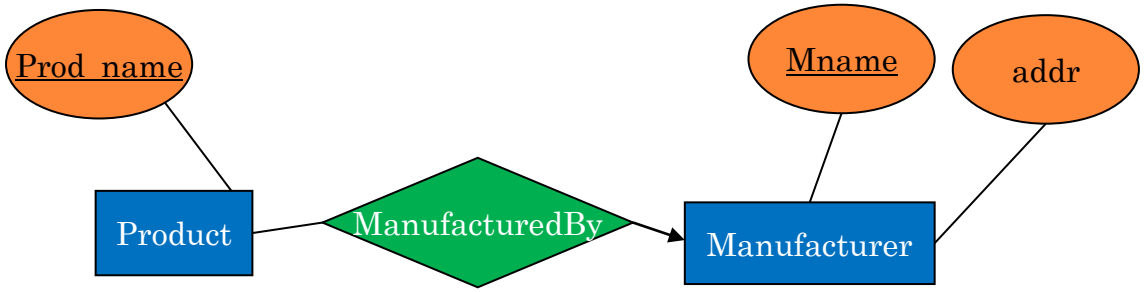



This design repeats the manufacturer's address once for each product and loses the address if there are temporarily no product for a manufacturer.

# Entity Sets vs Attributes

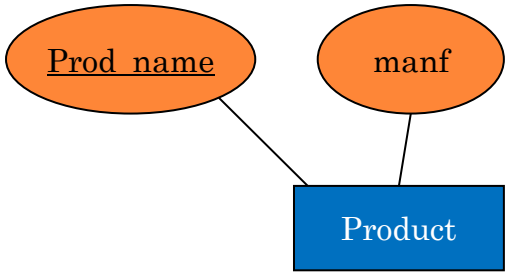
- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one nonkey attribute or
  - It is the "many" in a many-one or many-many relationship.

# EXAMPLE: GOOD



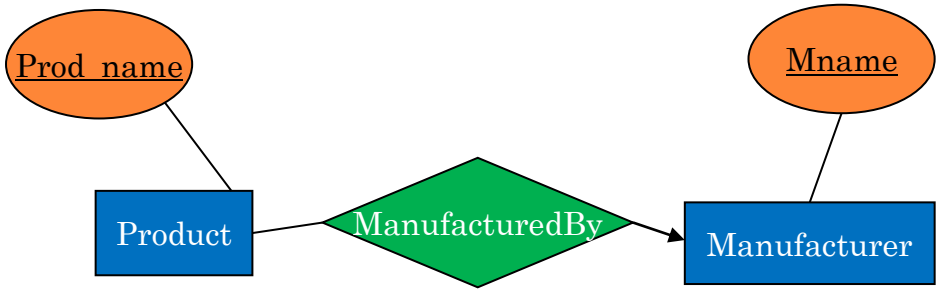

Manufacturer deserves to be an entity set because of the nonkey attribute addr.
Product deserves to be an entity set because it is the "many" of the one-to-many relationship ManufacturedBy.

# EXAMPLE: GOOD




There is no need to make the manufacturer an entity set, because we record nothing about manufacturers besides their name

# EXAMPLE: BAD




Since the manufacturer is nothing but a name, and is not at the "many" end of any relationship, it should not be an entity set.

# Don't overuse weak entity set

- Beginning database designers often doubt that anything could be a key by itself.
  - They make all entity sets weak, supported by all other entity sets to which they are linked.
- In reality, we usually create unique ID's for entity sets.
  - Examples include social-security numbers, automobile VIN's etc.

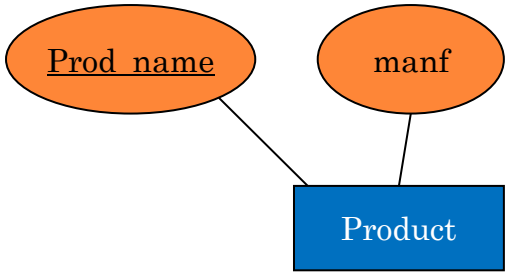
# When do we need weak entity set

- The usual reason is that there is no global authority capable of creating unique ID's.
- Example: it is unlikely that there could be an agreement to assign unique player numbers across all football teams in the world.

# From ER Diagram to Relations

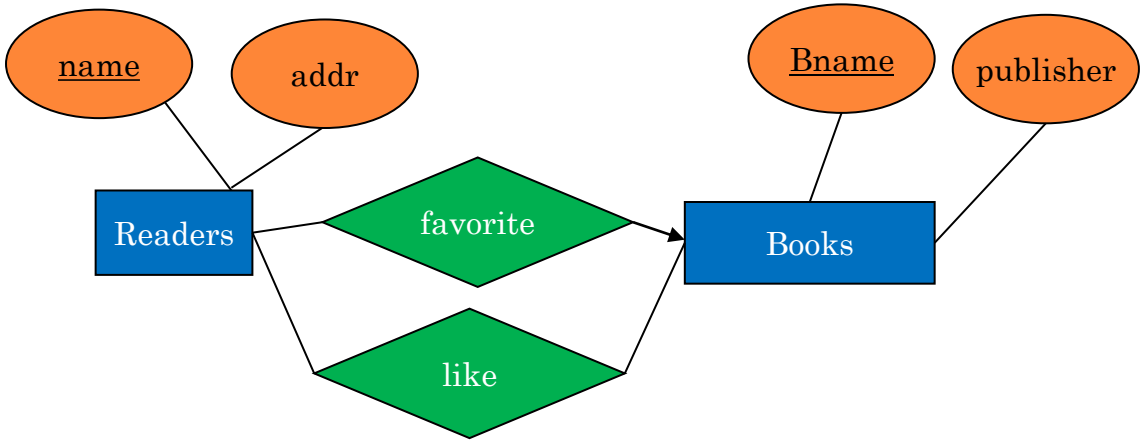
- Entity set -> relation.
- Attributes -> attributes.
- Relationships -> relations whose attributes are only:
  - The keys of the connected entity sets.
  - Attributes of the relationship itself

# Entity set to relation




Relation: Product(Prod_name, manf)

# Relationship to relation




like(name, Bname)
favorite(name, Bname)

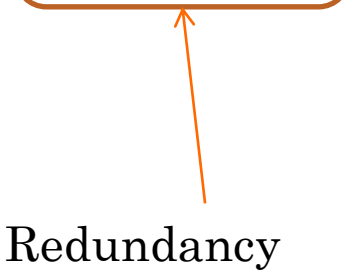
# Combining relations

- OK to combine into one relation:
  - 1. The relation for an entity-set E
  - 2. The relations for many-one relationships of which E is the "many."
    - Example: Readers(name, addr) and Favorite(name, Bname) combine to make Reader1(name, addr, favBook).

# Risk with many-many relationships

- Combining Readers with Likes would be a mistake. It leads to redundancy, as:

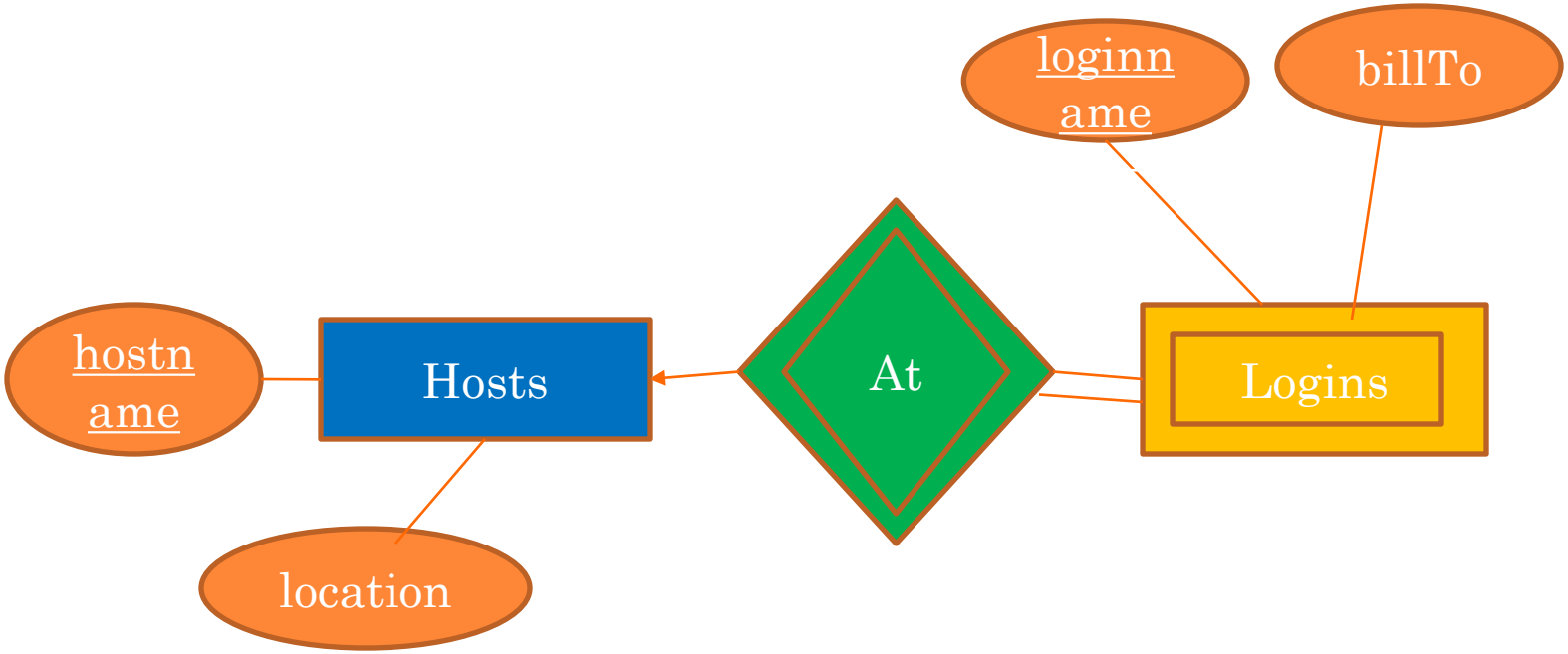
| Name | Addr | Bname |
|---|---|---|
| Rabi | Patna | The Post Office |
| Rabi | Patna | Gitanjali |



# Handling weak entity sets

- Relation for a weak entity set must include attributes for its complete key (including those belonging to other entity sets), as well as its own, nonkey attributes.
- A supporting relationship is redundant and yields no relation (unless it has attributes).

# Example: Weak Entity Set to Relation




Hosts(hostName, location)
Logins(loginName, hostName, billTo)

At becomes part of Logins

# Practice Problem

- Construct an E-R diagram for a car insurance company whose *customers* own one or more *cars* each. Each *car* is associated with it zero to any number of recorded *accidents*. Each insurance *policy* covers one or more *cars*, and has one or more premium *payments* associated with it. Each *payment* is for a particular period of time, and has an associated due date, and the date when the *payment* was received.