CS354: Database

Introduction

INSTRUCTOR & TAS

- Instructor
  - Dr. Samrat Mondal
  - Email: samrat@iitp.ac.in
- TA list for CS354/CS355
  1. Pranay, <pranay.pcs16@iitp.ac.in>
  2. Sanghamitra, <1821cs14@iitp.ac.in>
  3. Arun, <arun_1921cs20@iitp.ac.in>
  4. Ritam, <ritam_1921cs29@iitp.ac.in>
  5. Anmol, <anmol_1911cs04@iitp.ac.in>
  6. Karanjit, <karanjit_1911cs09@iitp.ac.in>
  7. Neeti, <neeti_1911cs10@iitp.ac.in>
  8. Shashi, <shashi_1911cs13@iitp.ac.in>

- Course Structure
  - Theory: CS354 (3-0-0-6)
  - Lab: CS355(0-0-3-3)
- Class Timings/ Venue
  - Mon: 11:00am - 11:55am
  - Tue: 11:00am – 11:55am
  - Thur: 11:00am – 11:55am, Lab-3:00pm – 6:00pm
- Contact Hour
  - Friday 5:00 pm to 6:00 pm or appointment through email
- Course Link
  - Slides, lecture materials, assignments will be uploaded to https://www.iitp.ac.in/~samrat/CS354_CS355

Evaluation Policy

- CS354:
  - Premid Sem Assessment: 20% (Assignment/Quiz, Class Performance)
  - MidSem: 30%
  - Postmid Sem Assessment: 20% (Assignment/Quiz, Class Performance)
  - EndSem: 30%
- CS355:
  - Assignment, Projects, Viva: 40%
  - MidTerm Test: 30%
  - EndTerm Test: 30%

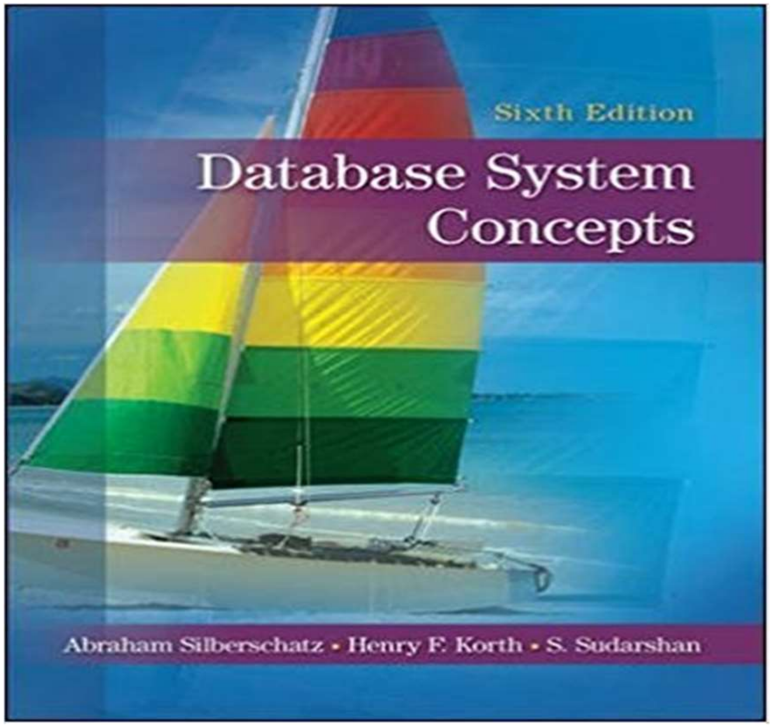
SYLLABUS (1)

- **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: SQL Injection Attack**

TEXT BOOK


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

Reference Books

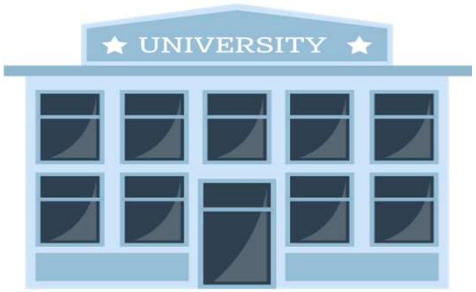
- R. Ramakrishnan and J. Gehrke, Database Management Systems, 3rd Ed, McGraw Hill, 2003.
- R. Elmasri, S. B. Navathe and R. Sunderraman, *Fundamentals of Database Systems, 5th Ed, Pearson, 2009.*
- H. Garcia-Molina, J. D. Ullman and J. D. Widom, *Database Systems: The Complete Book, Prentice Hall, 2002.*

LET'S START OUR JOURNEY

Database (DB)

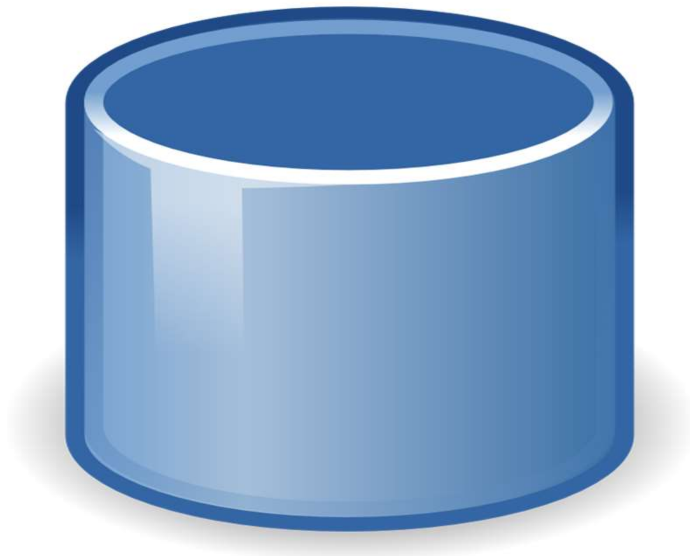
- A collection of interrelated data
- Usually designed to manage large bodies of information
- Models real world enterprise
  - Entities (e.g. student, courses)
  - Relationships (e.g. students are enrolled to courses)
- A ***database management system*** (DBMS) is a collection of interrelated data and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

Messy vs Organized

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

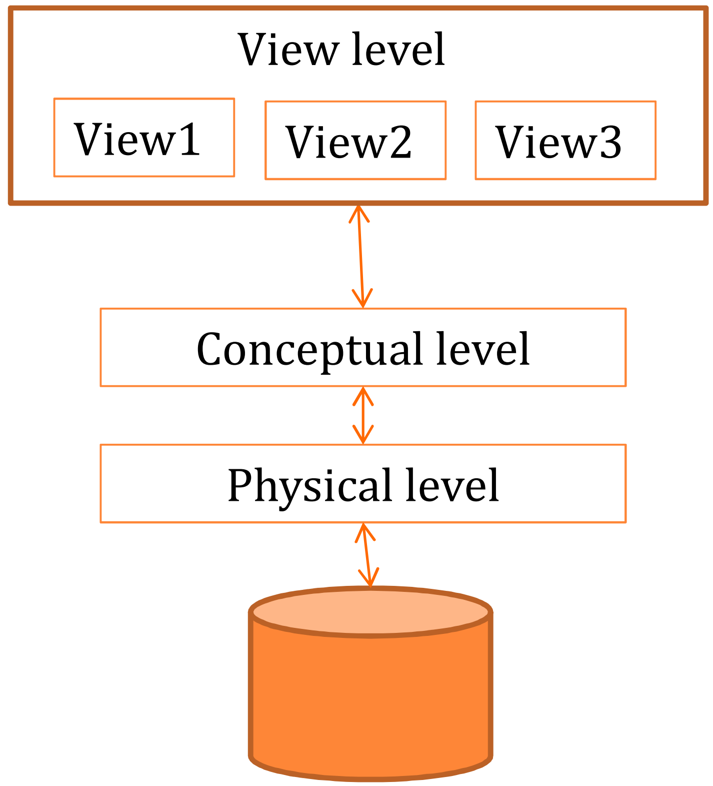
- Data independence and efficient access
- Reduced application development time
- Data integrity and security
- Uniform data administration
- Concurrent access, recovery from crashes

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) level* and *physical level*
- Views describe how different users see the database
- Conceptual or Logical level describes what data are stored and what is the relationships exist among those data
- Physical level describes how the data are actually stored



Example: University Database

- Physical level:
  - Relations stored as unordered files.
  - Index on first column of Students.
- Conceptual level:
  - *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 level (View):
  - *Course_info(cid:string, sid: string)*

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
- Several schemas are possible:
  - Physical schema: describes database design at the physical level
  - Logical schema: describes database deign at the logical level
  - Subschemas: schemas at the view level

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*

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

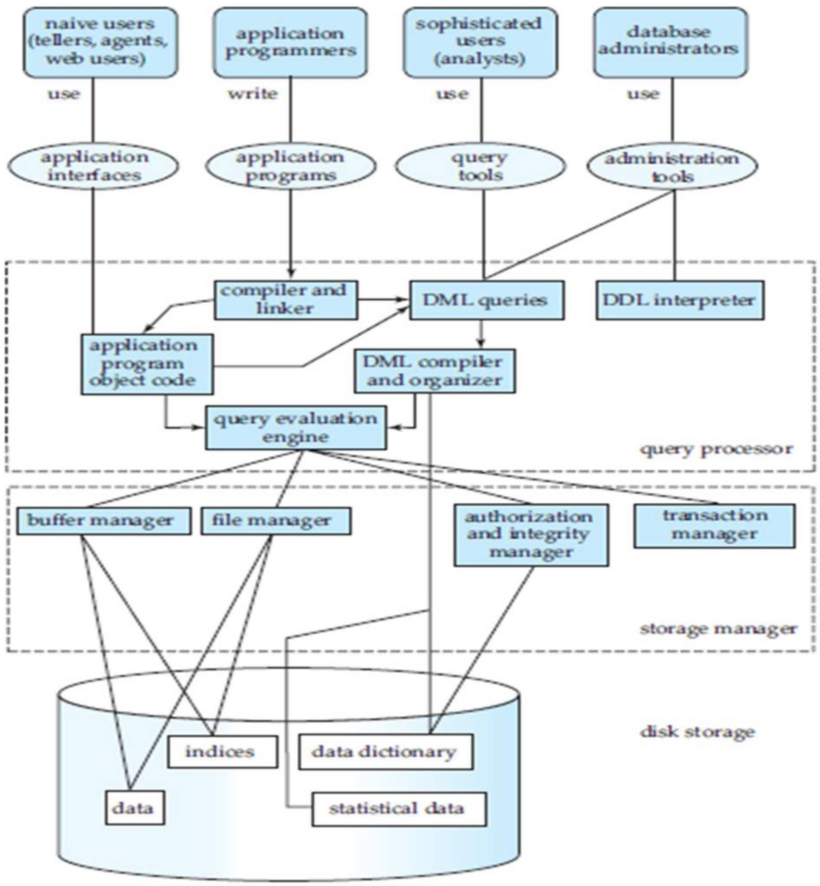
- Data manipulation may be-
  - the retrieval of the information stored in the database
  - the insertion of the new information in the database
  - the deletion of the information in the database
  - the modification of the information stored in the database
- Example

  select balance

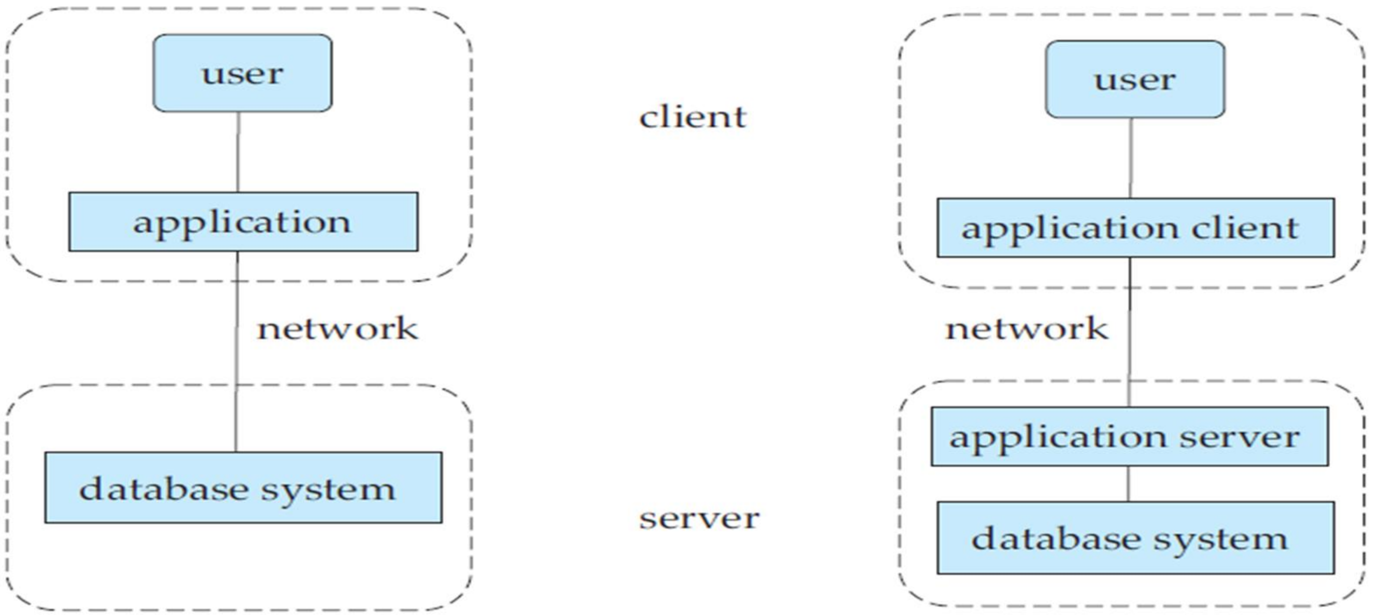
  from account

  where account_number = “A123”

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

ER Model

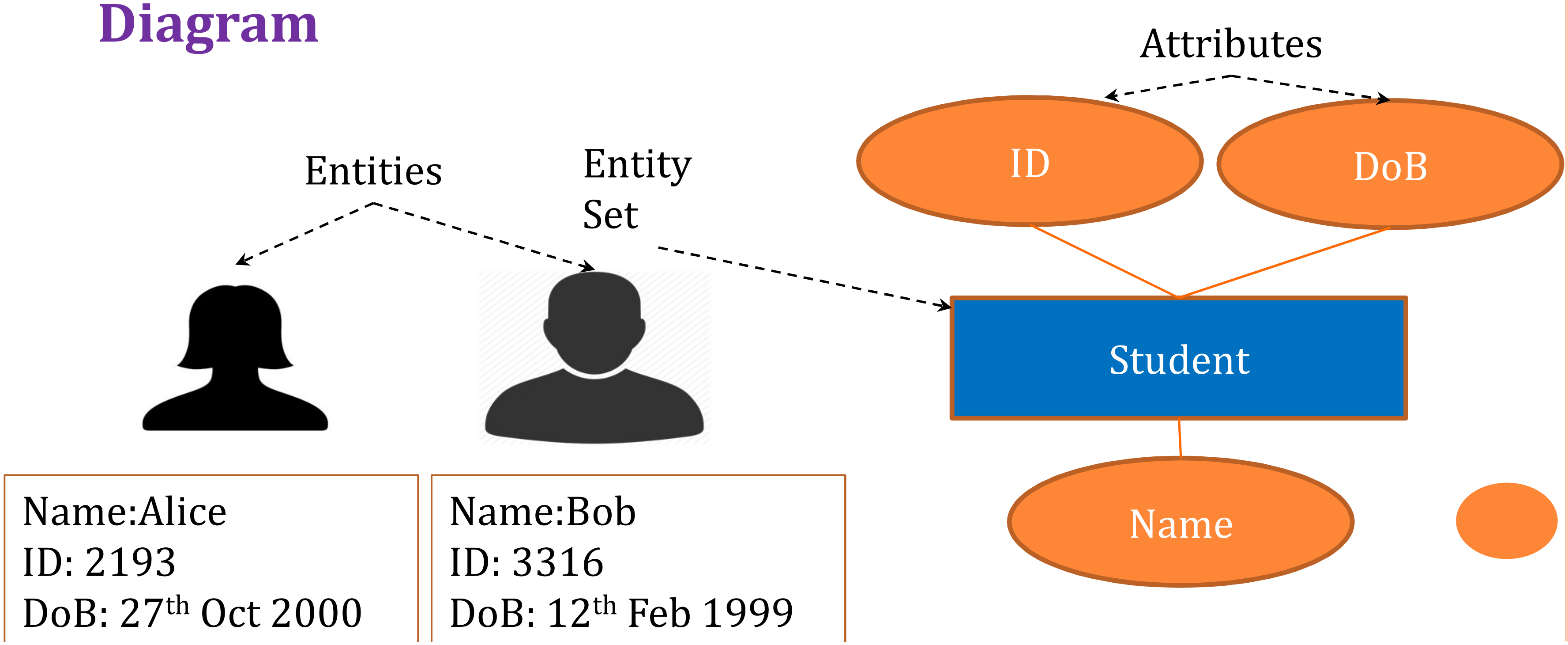
- Data model to describe the database system at the requirements collection stage
  - high level description
  - easy to understand for the enterprise managers
  - rigorous enough to be used for system building
- Concepts available in the model
  - entities and attributes of entities
  - relationships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

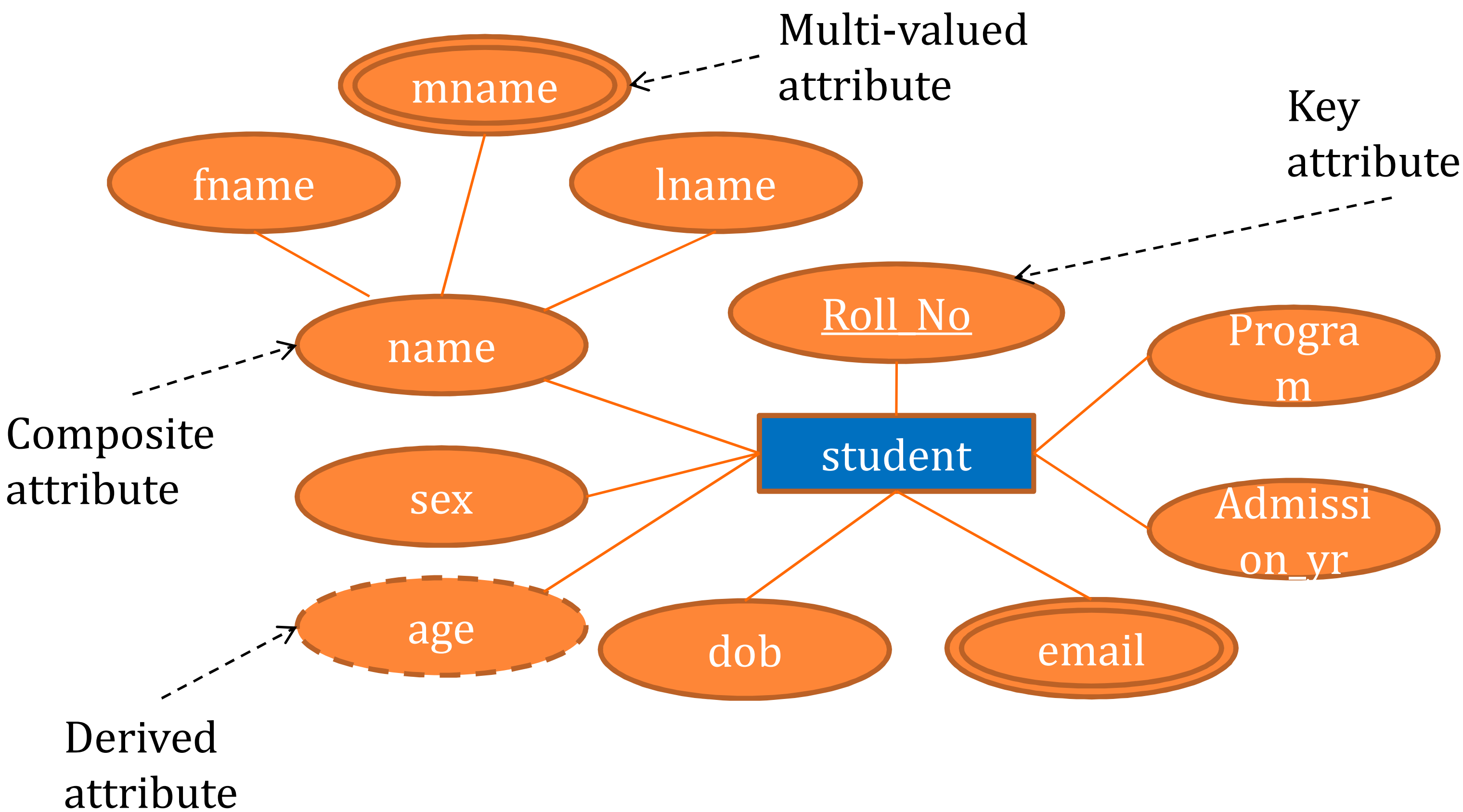
Types of Attributes

- 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

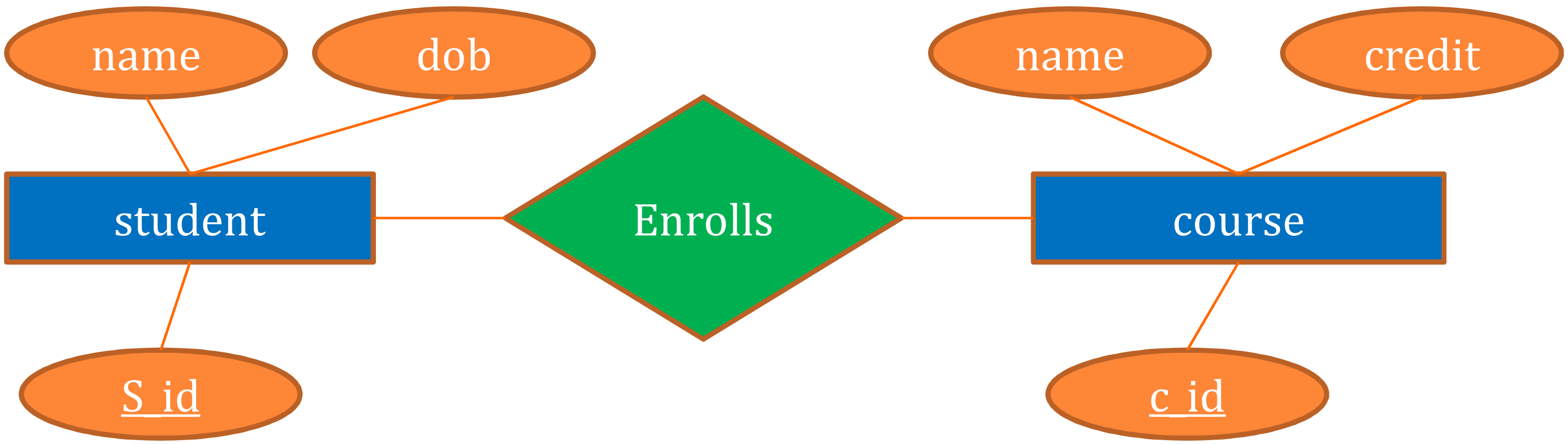
ENTITY SETS AND KEY ATTRIBUTES

- 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*

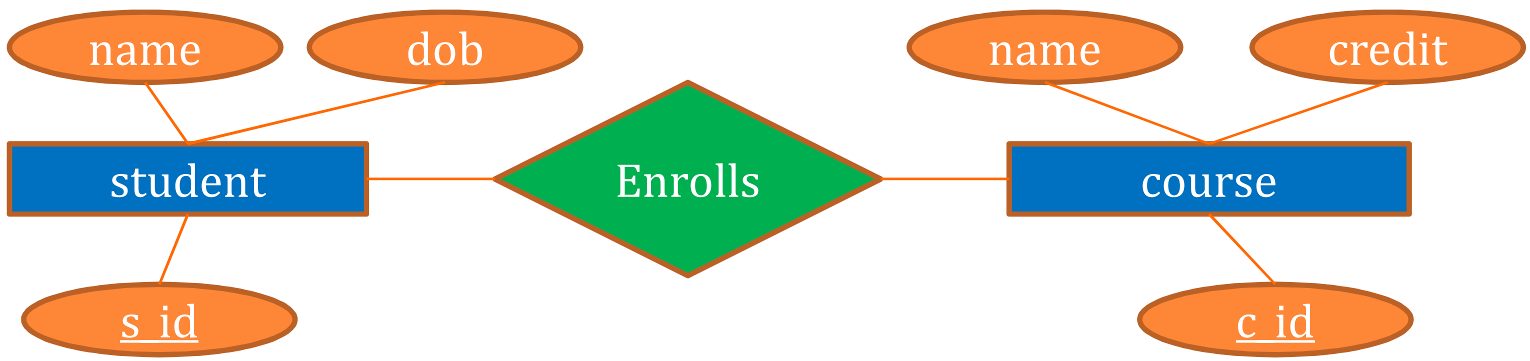


Mathematical Interpretation

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

KEYS OF RELATIONSHIP

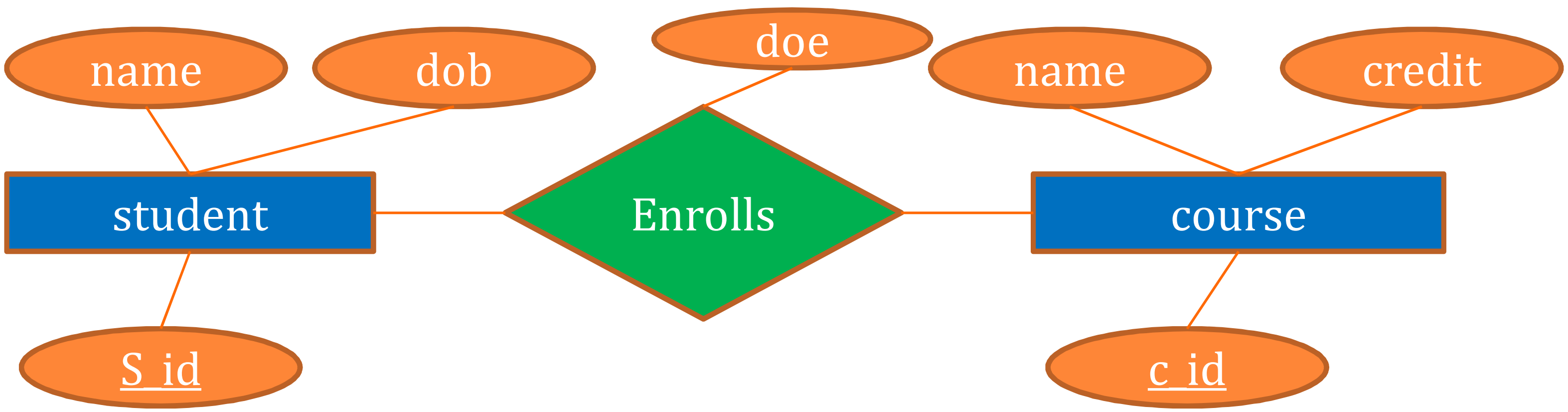
- Generally, **the relationship is uniquely determined by the keys of its entities**
- *Example: the “key” for Enrolls is {s_id, c_id}*



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

RELATIONSHIPS AND ATTRIBUTES

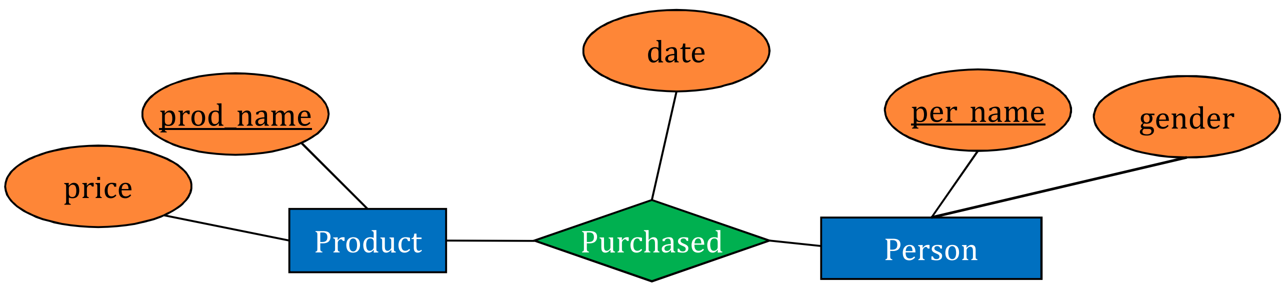
- Relationships may have attributes as well.



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?

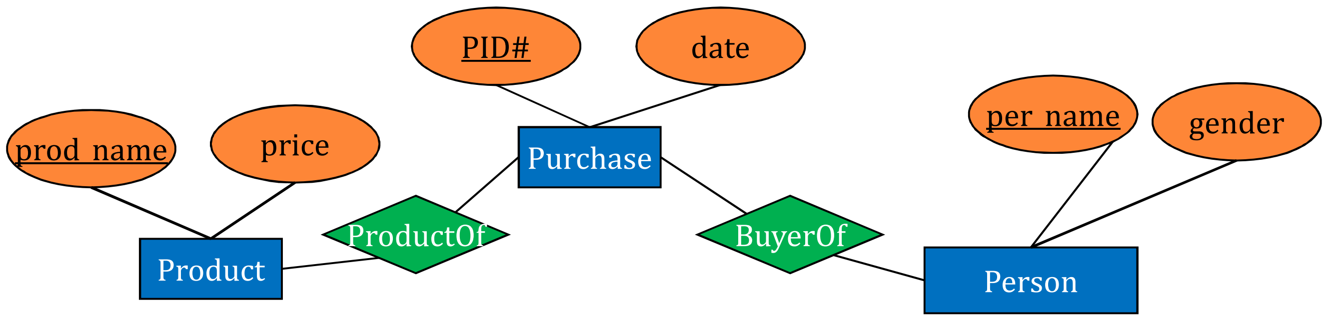


- **A:** A person can only buy a specific product once (on one date)

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!*

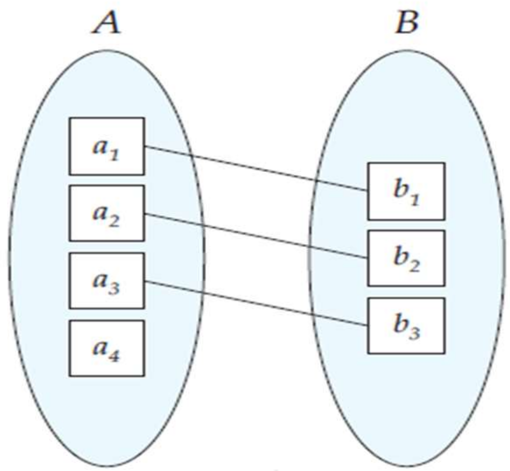
We can always use **a new entity** instead of a relationship. For example, to permit multiple instances of each entity combination!

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.

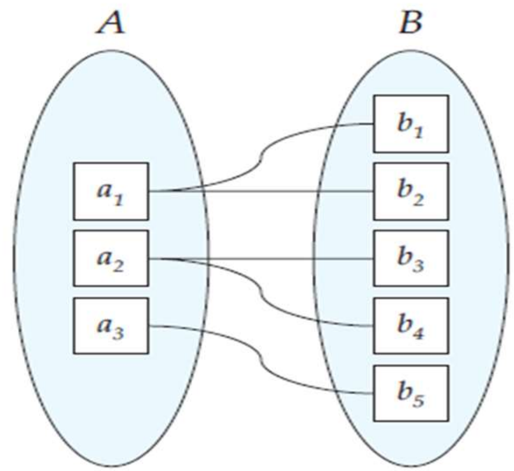
Four possibilities-

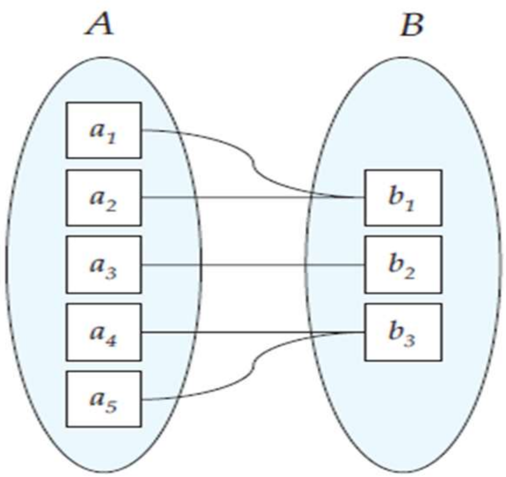
one to one, one to many, many to one and many to many



One to One

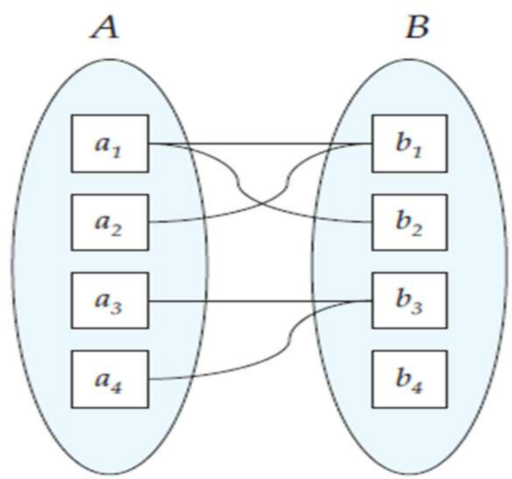
One to Many

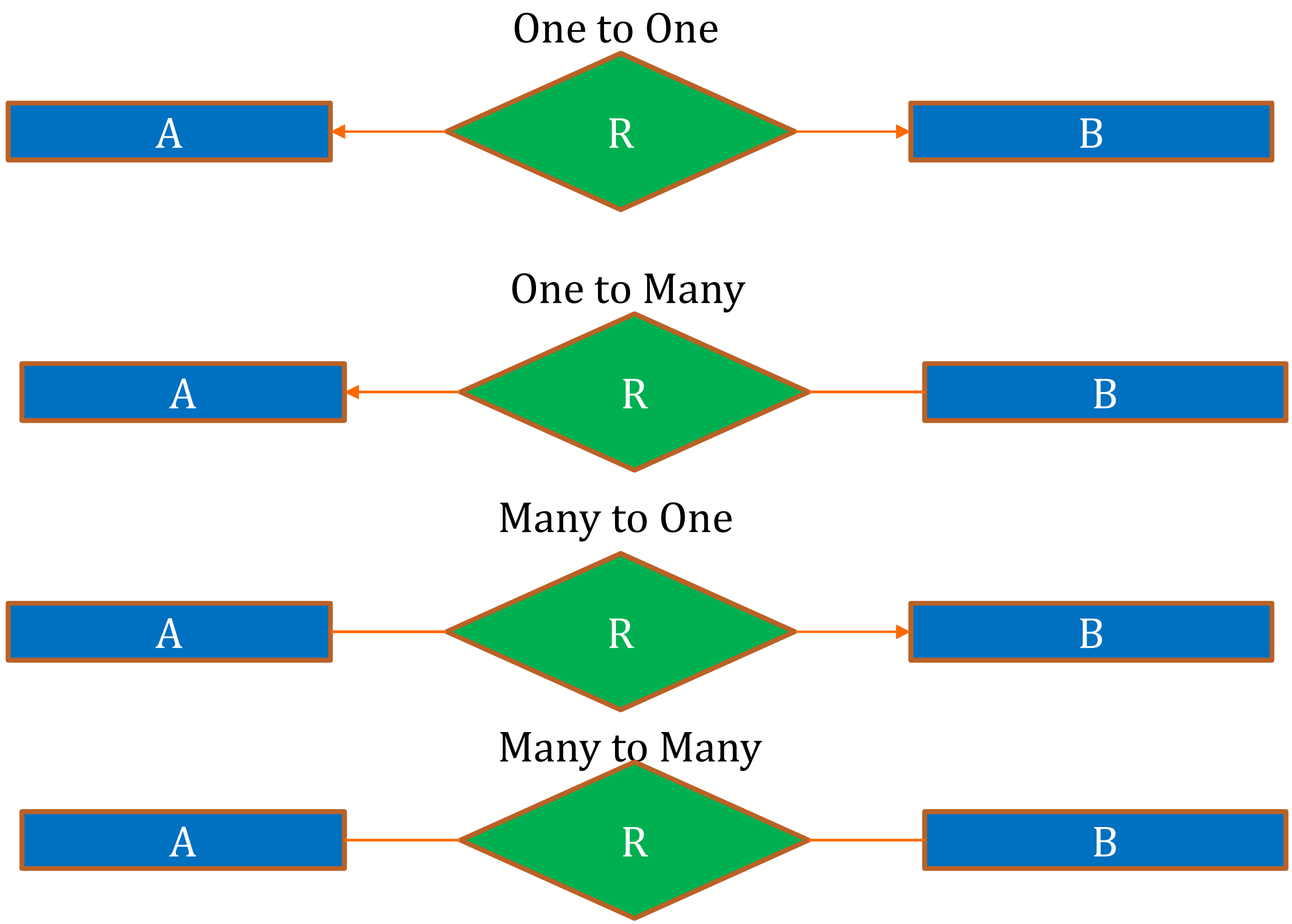




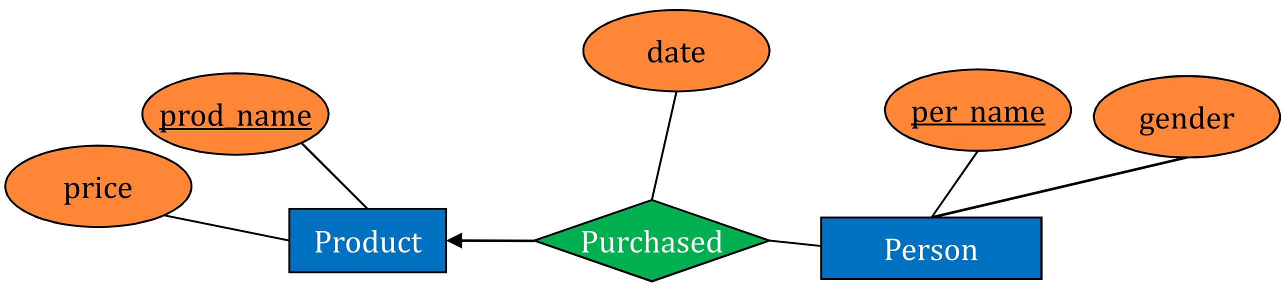
Many to One

Many to Many



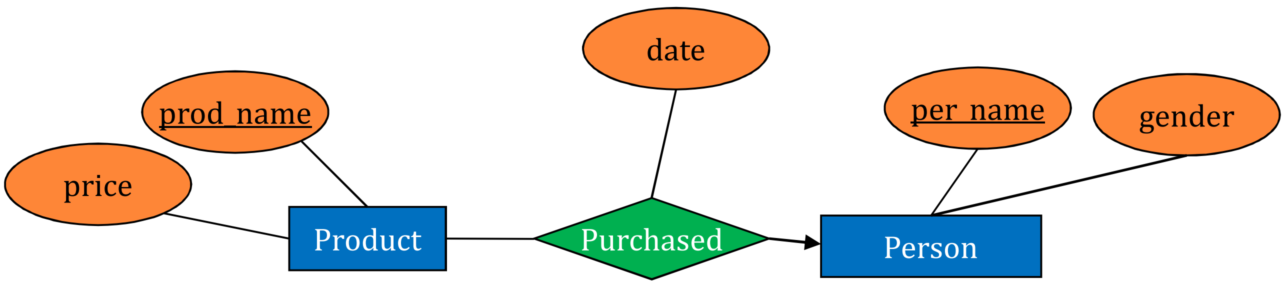

One to One
A
R
B
One to Many
A
R
B
Many to One
A
R
B
Many to Many
A
R
B

EXAMPLE: ONE TO MANY

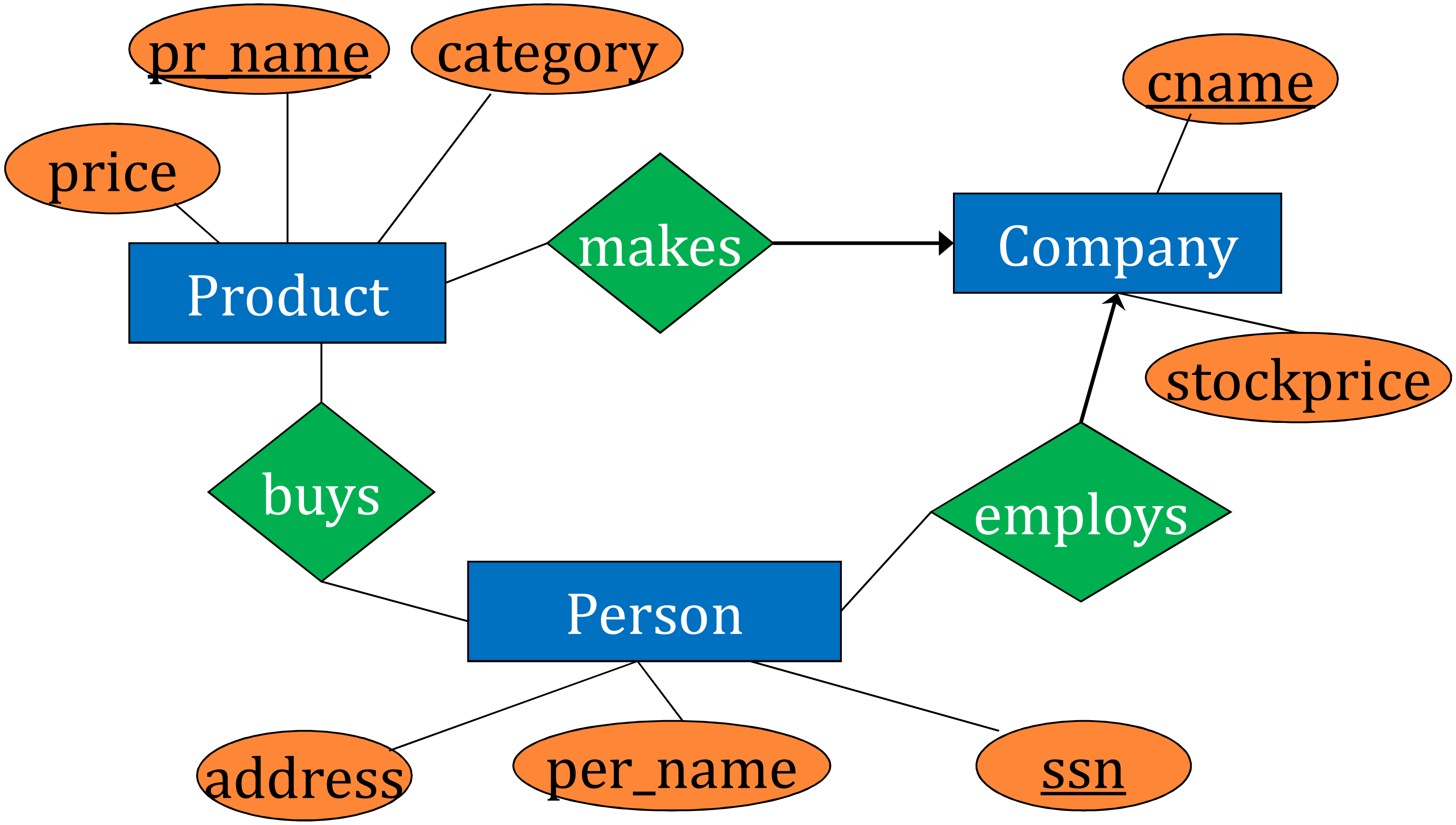


One person can be associated with atmost one product and one product can be associated with any (0 or more) number of persons through Purchased relationship

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?

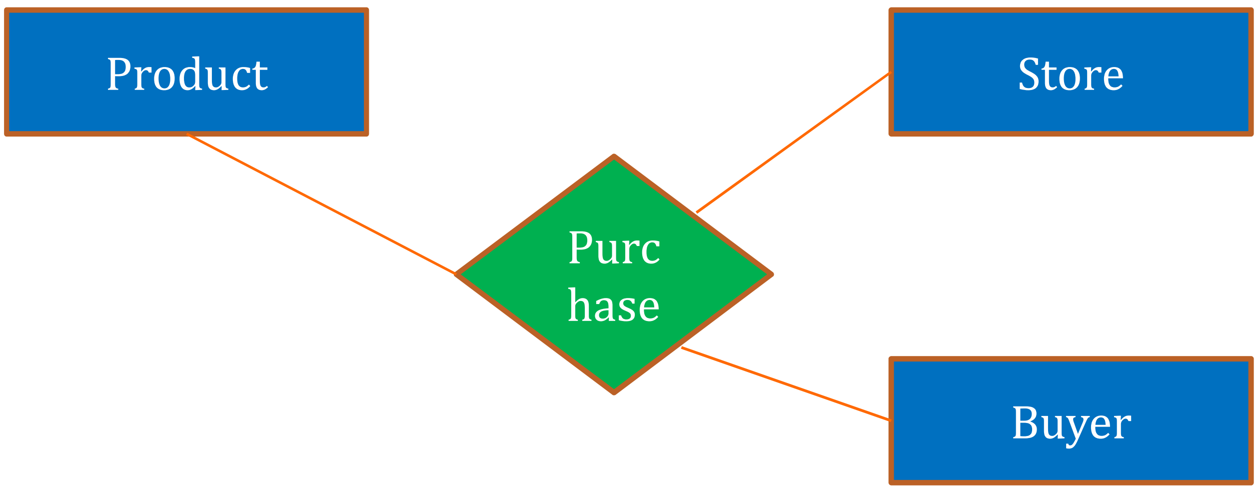
Degree of a Relationship

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

Binary relationships are very common and widely used

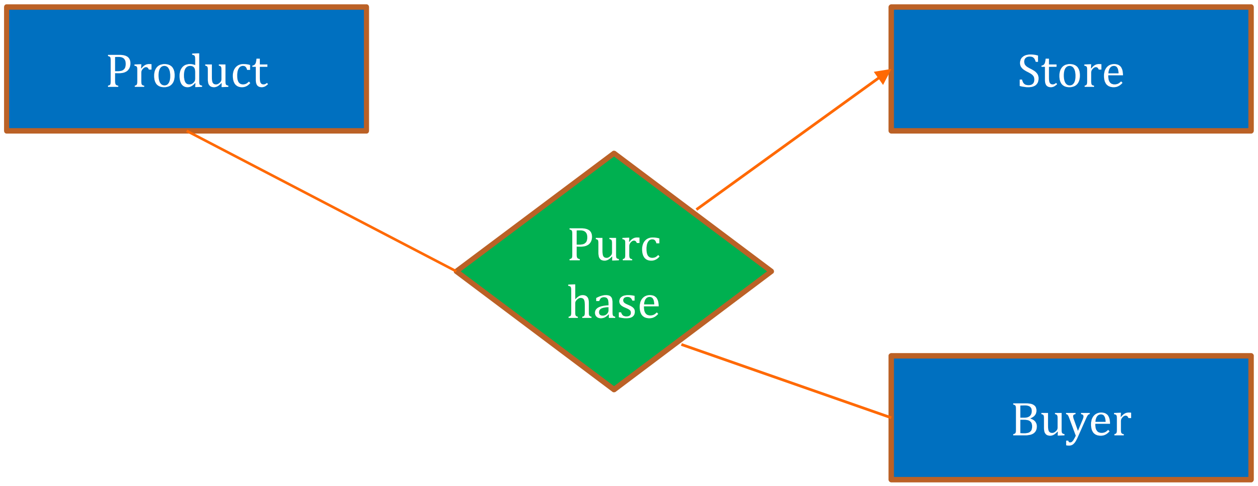
Multi-way Relationship

Modeling a purchase relationship between product, store and buyers



MULTI-WAY RELATIONSHIP

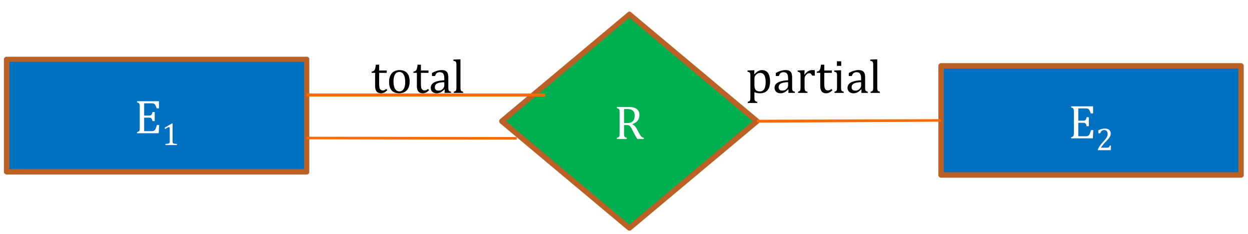
What is the meaning of the following relationship?



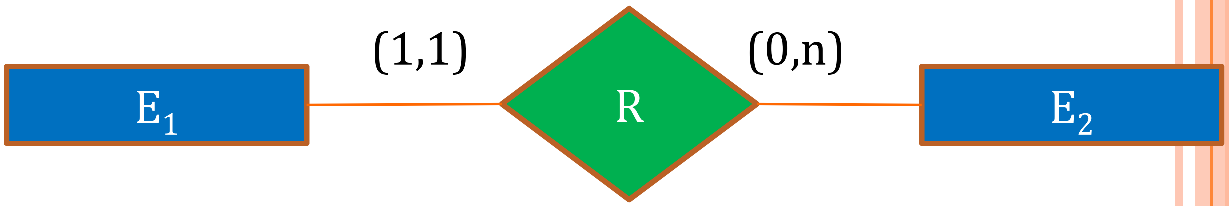
For each unique combination of product and buyer, there will be atmost one store associated

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



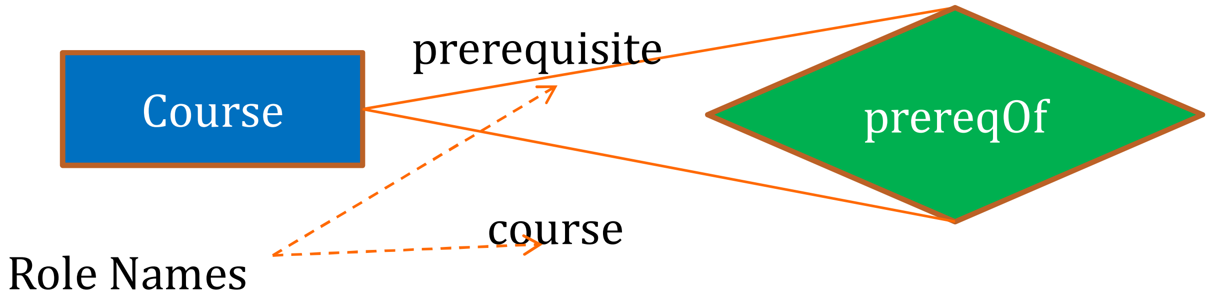
STRUCTURAL CONSTRAINTS



- 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

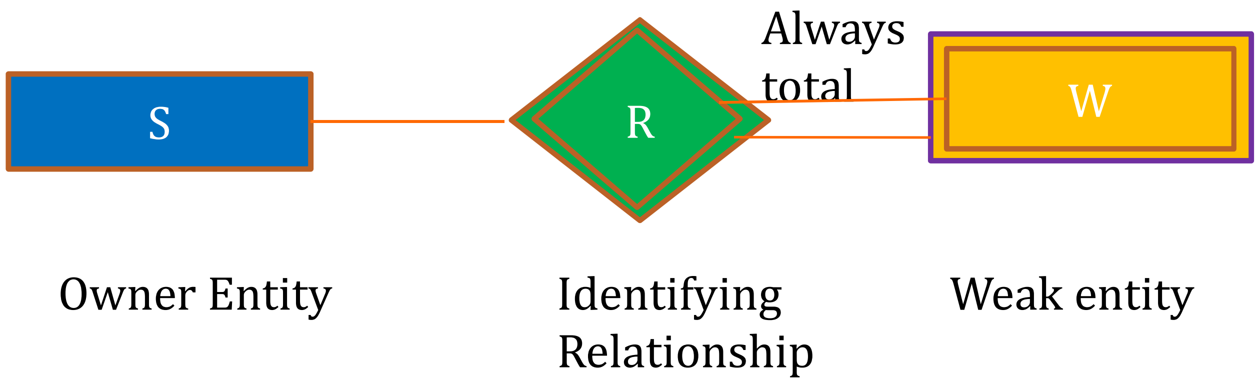
RECURSIVE RELATIONSHIP AND ROLE NAME

- 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

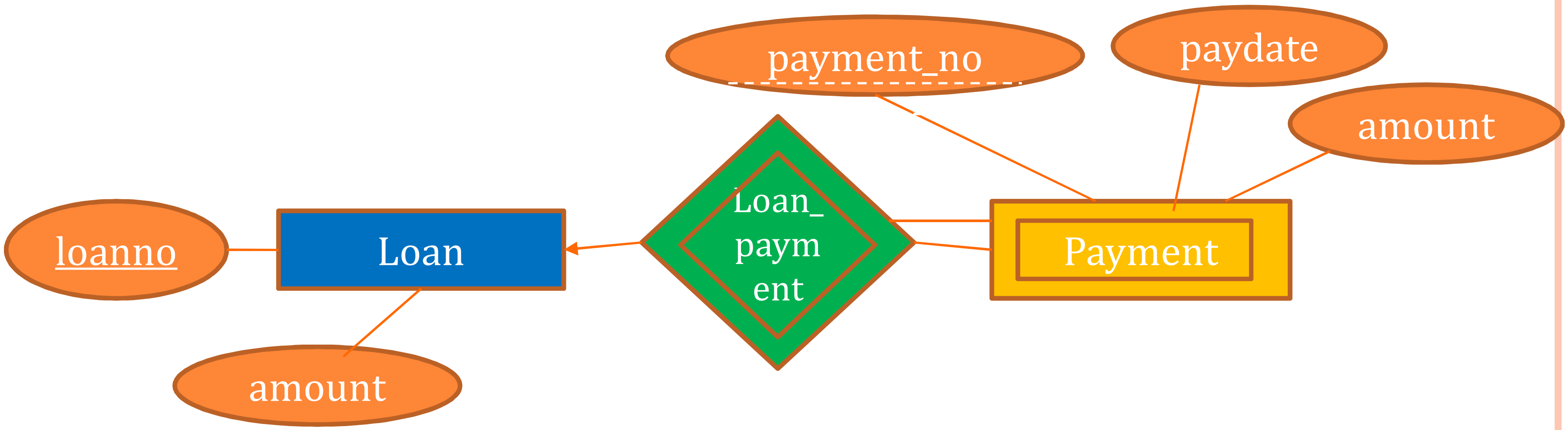


WEAK ENTITY SET

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



Weak Entity Set Example



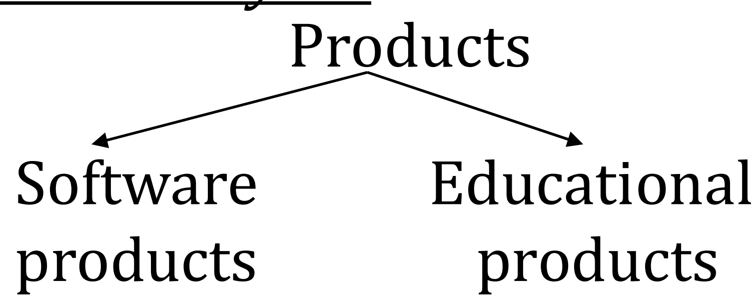
MODELING SUBGROUPINGS

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

  Define a new entity set?

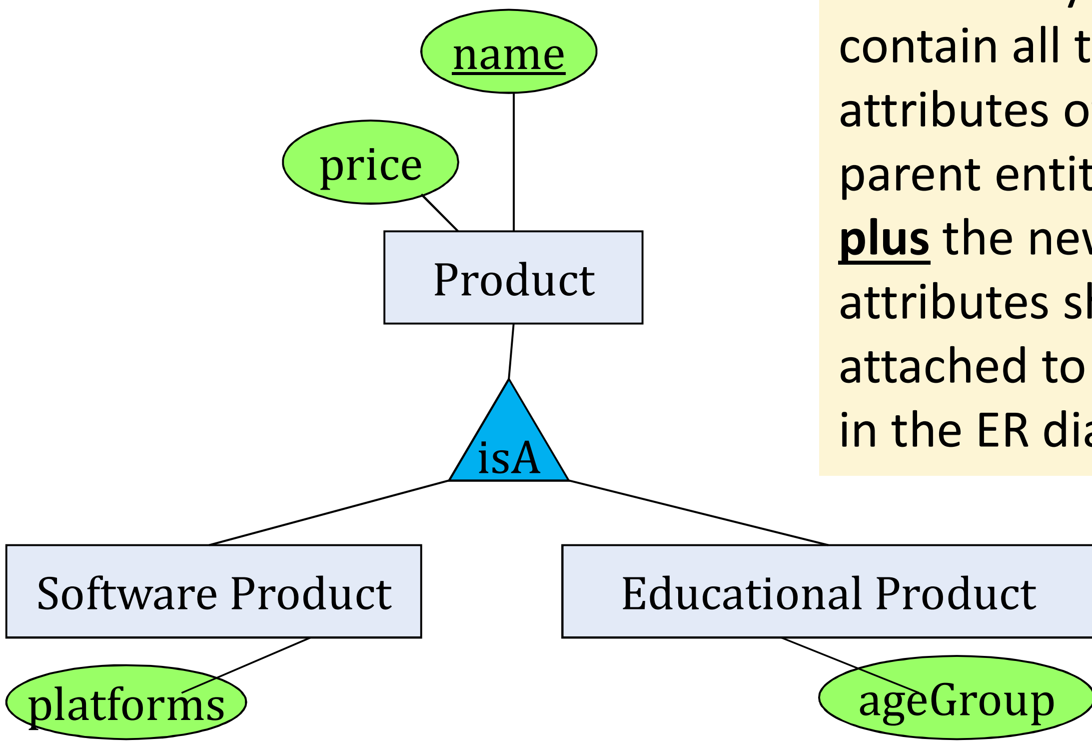
  - *But what if we want to maintain connection to current entity set?*

- <u>Better: define a *sub-entity set*</u>
  - *Ex:*



We can define **subgroups** in ER Diagram

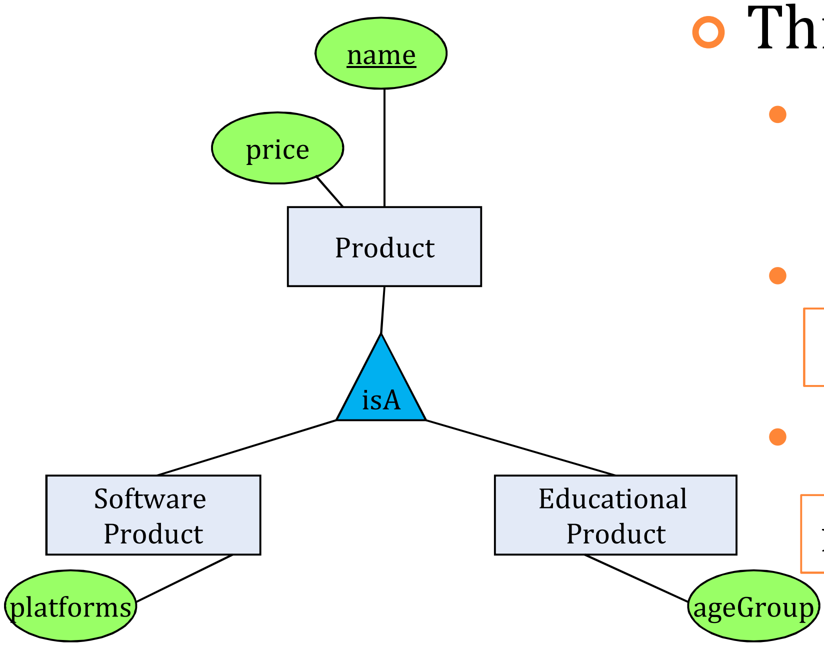
Modeling Subgroups



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

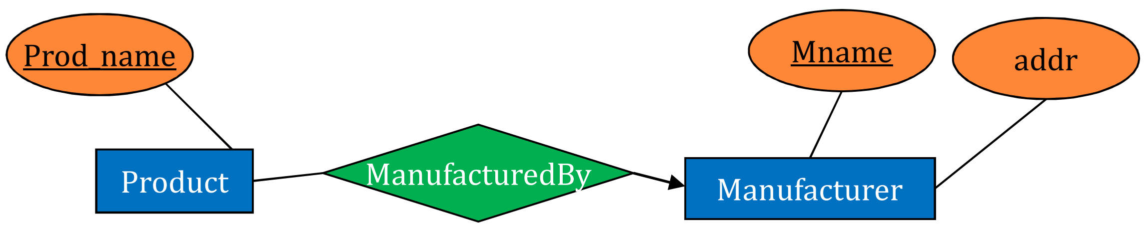
Some Design Tips

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

Avoiding Redundancy

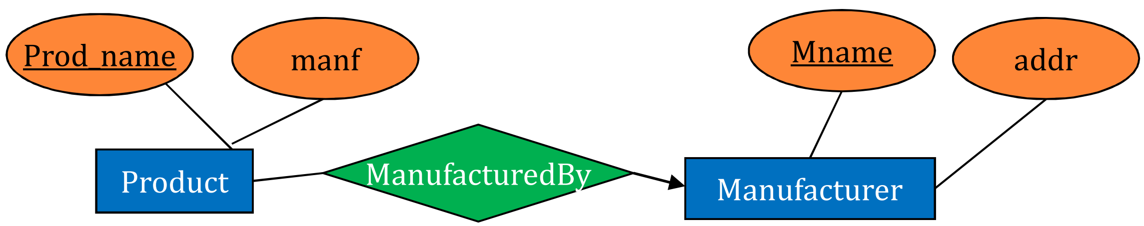
- Redundancy = saying the same thing in two (or more) different ways.
- Wastes space and (more importantly) encourages inconsistency.
- Two representations of the same fact become inconsistent if we change one and forget to change the other.

EXAMPLE: GOOD



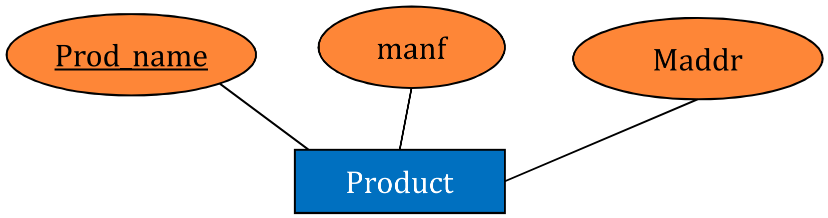
This design gives the address of each manufacturer exactly once.

EXAMPLE: BAD



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

Example: Bad

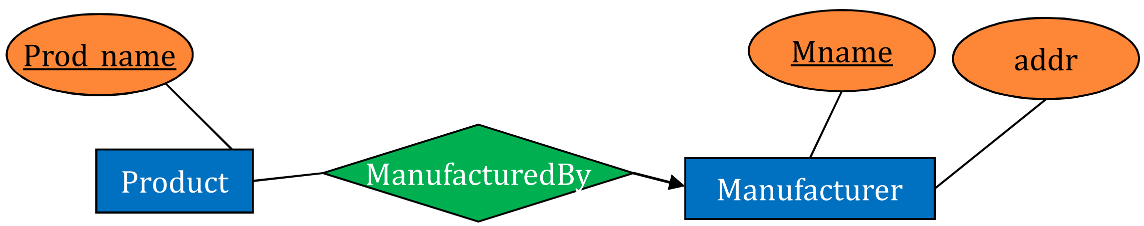


This design repeats the manufacturer's address once for each product and loses the address if there are temporarily no product for a manufacturer.

Entity Sets vs Attributes

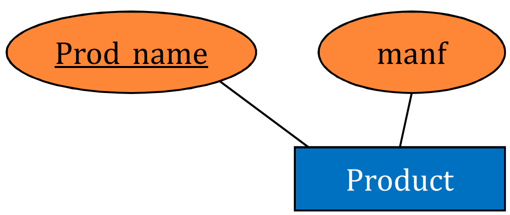
- An entity set should satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one nonkey attribute or
  - It is the "many" in a many-one or many-many relationship.

EXAMPLE: GOOD



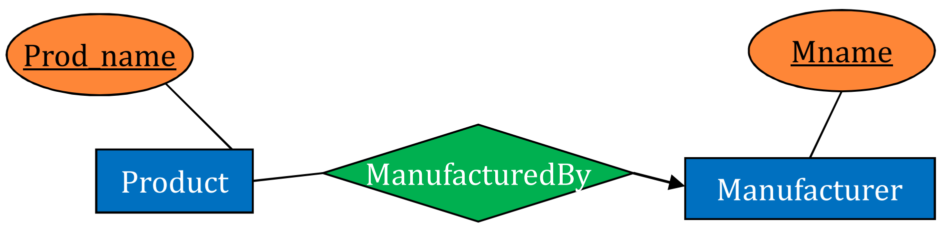
Manufacturer deserves to be an entity set because of the nonkey attribute addr
Product deserves to be an entity set because it is the "many" of the many-to-one relationship ManufacturedBy

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.

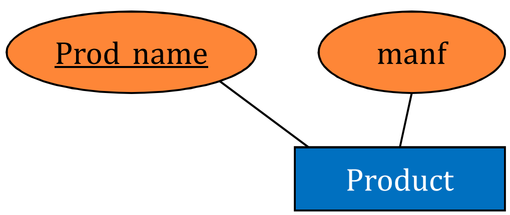
When do we need weak entity set

- 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

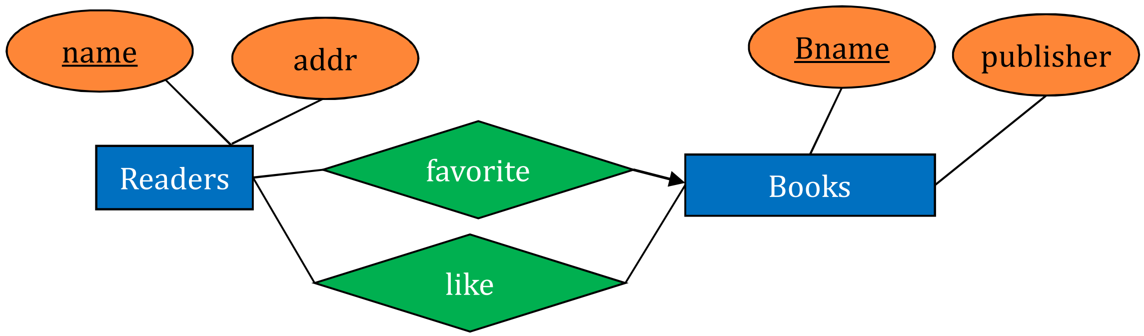
- Entity set -> relation.
- Attributes -> attributes.
- Relationships -> relations whose attributes are only:
  - The keys of the connected entity sets.
  - Attributes of the relationship itself

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).

Is it a good idea to combine Readers (name, addr) and Likes (name, Bname)?

Risk with many-many relationships

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

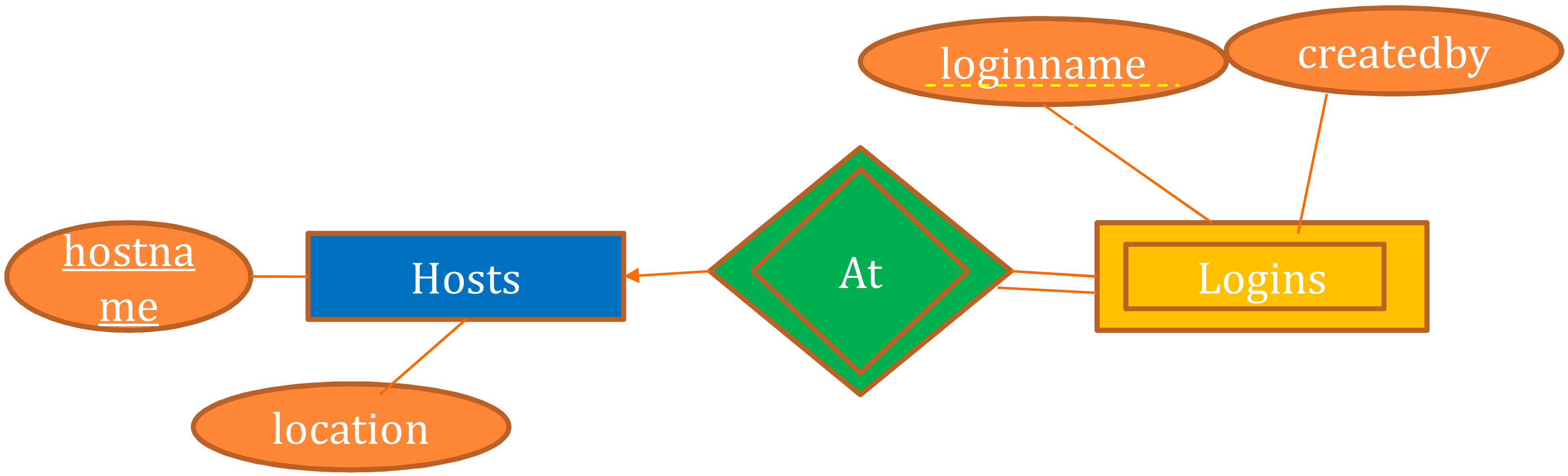
Name	Addr	Bname
Rabi	Patna	The Post Office
Rabi	Patna	Gitanjali

Redundancy

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 (a.k.a. identifying relationship) is redundant and yields no relation (unless it has attributes)

EXAMPLE: WEAK ENTITY SET TO RELATION



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

At becomes part of Logins