



UNIT:2

Database Design

Prepared By: Prof. Meghna Bhatt

TOPICS TO BE DISCUSSED....

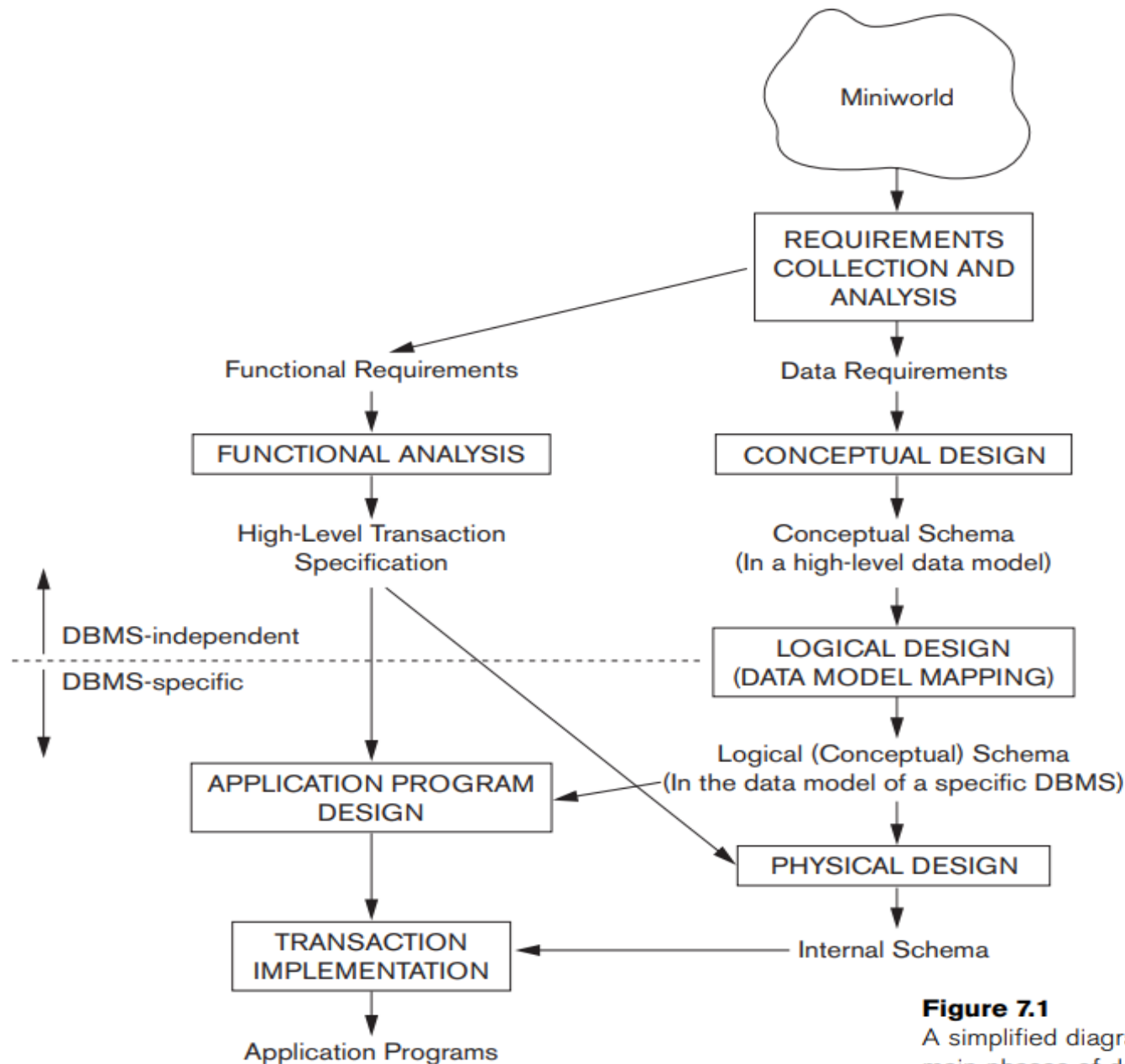
- ❖ The basics of E-R Model
- ❖ Components of E-R Diagram
- ❖ Entity
- ❖ Attribute and Relationships
- ❖ Mapping Cardinality
- ❖ Binary and Ternary relationship
- ❖ E-R Diagram Notations
- ❖ The concept of weak entity set
- ❖ E-R diagram illustrations
- ❖ Extended E-R features- specialization & generalization
- ❖ Normalization Process:
- ❖ Overview of functional dependency, types of
- ❖ functional dependencies, decomposition process for a
- ❖ relation, Normalization – 1NF, 2NF and 3NF.



INTRODUCTION

- ❖ Database application refers to a particular database and the associated programs that implement the database queries and updates.
- ❖ The application designing has 2 phases :
 - ❖ Application Programs
 - ❖ Data Base Designing
- ❖ A major part of the database application will require the design, implementation, and testing of these application programs.



**Figure 7.1**

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

HISTORY

- ❖ Peter Chen developed ERDs in the 1970s .
- ❖ Peter Chen was a computer scientist who worked on improving database design.
- ❖ He published his proposal for entity relationship modeling in a 1976 paper titled "The Entity-Relationship Model: Toward a Unified View of Data".
- ❖ His entity relationship model was a way to visualize a database that unified other existing models into a single understanding .



WHAT IS ER MODEL?

- ❖ The entity-relationship (ER) model, which is a popular high-level conceptual data model.
- ❖ ER Model represents logical structure of databases with the help of diagram.
- ❖ ER diagrams are created based on three basic concepts:
 1. Entities
 2. Attributes
 3. Relationships
- ❖ ER models are normally represented by ER-diagrams.

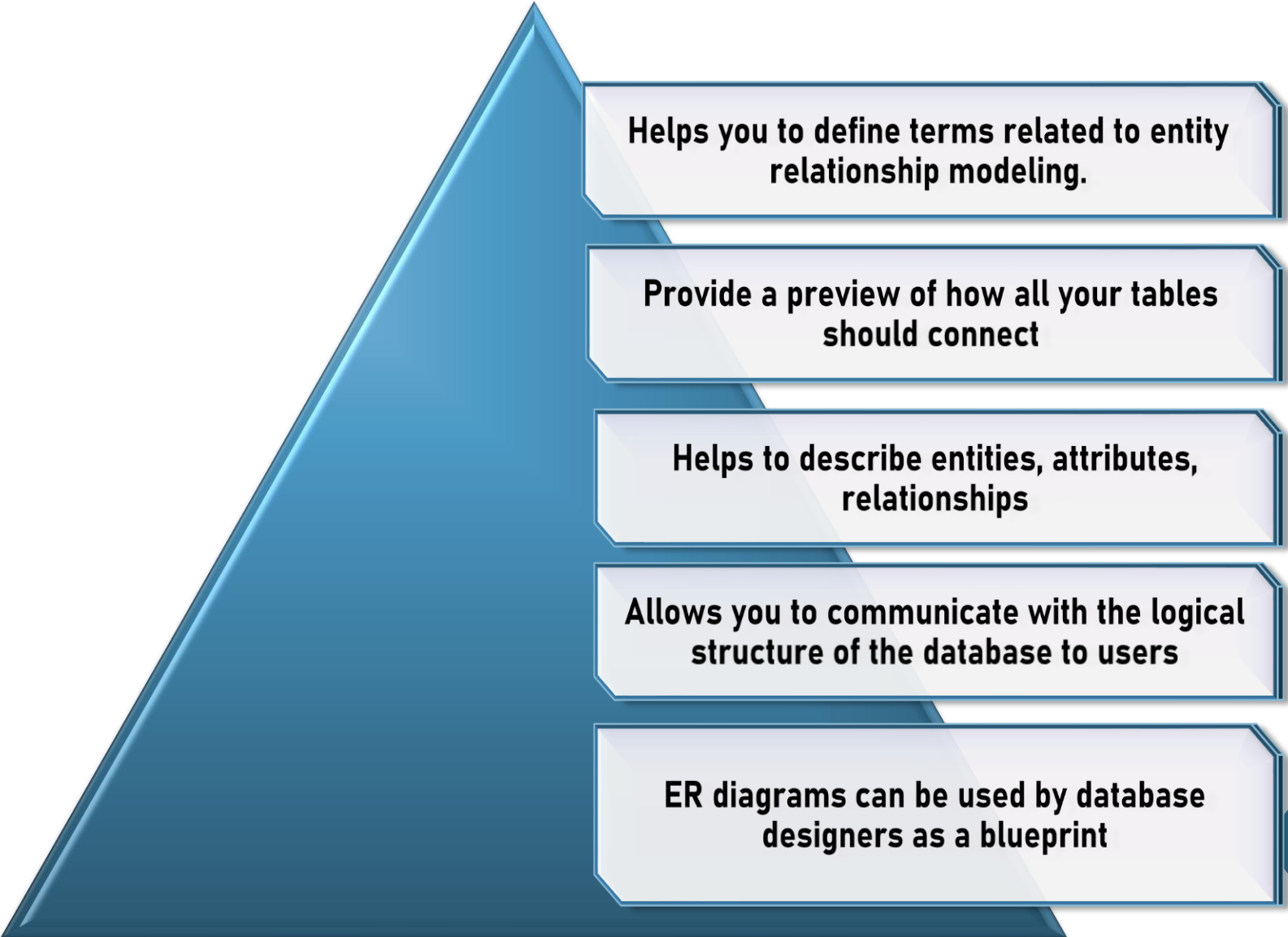


WHAT IS ER-DIAGRAM?

- ❖ **E-R diagram:** (Entity-Relationship diagram)
- ❖ It is **graphical (pictorial) representation** of database.
- ❖ ER diagram allows you to draw Database Design.
- ❖ It is Widely used in Database Design.
- ❖ It is a GUI representation of the logical structure of a Database.
- ❖ It uses different types of **symbols** to represent different **objects** of database.



WHY WE USE ER-DIAGRAM?



Helps you to define terms related to entity relationship modeling.

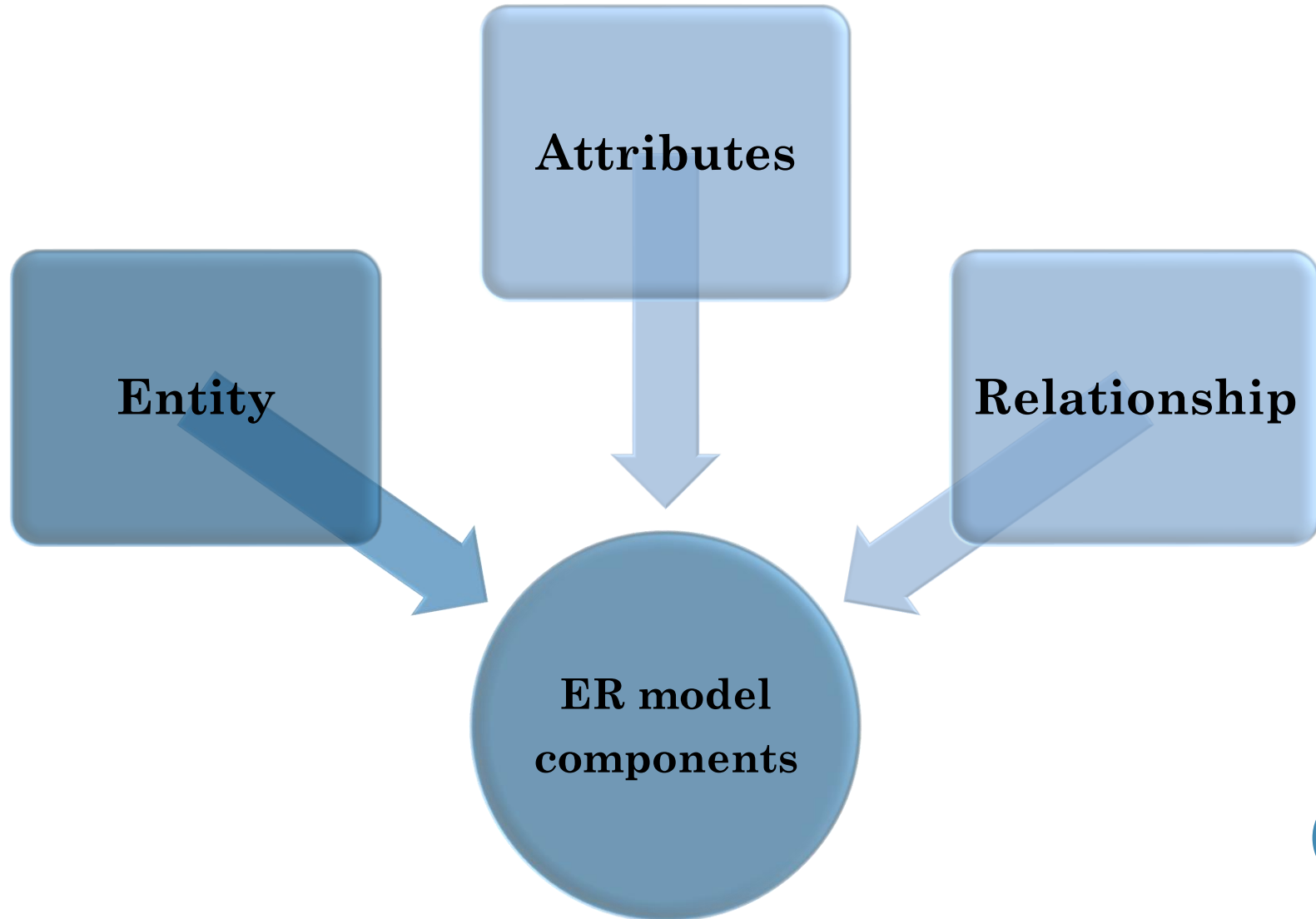
Provide a preview of how all your tables should connect

Helps to describe entities, attributes, relationships

Allows you to communicate with the logical structure of the database to users

ER diagrams can be used by database designers as a blueprint

COMPONENTS OF ER MODEL



COMPONENTS OF THE ER DIAGRAM



Entity Name

Entity

Person, place, object, event or concept about which data is to be maintained

Example: Car, Student



Jack

Attribute Name

Attribute

Property or characteristic of an entity

Example: Color of car Entity Name of Student Entity



Relation

Verb Phrase

Association between the instances of one or more entity types

Example: Blue Car Belongs to Student Jack



ENTITY



ENTITY

- ❖ An entity is a **person**, a **place** or an **object**.
- ❖ An entity is represented by a **rectangle** which contains the name of an entity.
- ❖ Entities of a college database are:
 - ❖ Student
 - ❖ Professor/Faculty
 - ❖ Course
 - ❖ Department
 - ❖ Result
 - ❖ Class
 - ❖ Subject



Entity
Name

Symbol



Student



ENTITY

Student

Entity Type

Roll_no	Student_name	Age	Mobile_no
1	Andrew	18	7089117222
2	Angel	19	8709054568
3	Priya	20	9864257315
4	Analisa	21	9847852156

Entity

ENTITY CLASSIFICATION

Tangible Entity

- ❖ Tangible Entities are those entities which exist in the real world physically.

Examples

- ❖ Person / student/ employee
- ❖ Car
- ❖ Product
- ❖ building
- ❖ Locker in bank
- ❖ Painting
- ❖ Pen
- ❖ Watch



ENTITY CLASSIFICATION

Intangible Entity:

- ❖ Entities that exist only logically and have no physical existence.

Examples

- ❖ Bank Account
- ❖ Order
- ❖ Location
- ❖ Balance in bank account



ENTITY SET

- ❖ It is a **set (group) of entities** of same type.
- ❖ **Example: Entity Sets Customer and Loan**

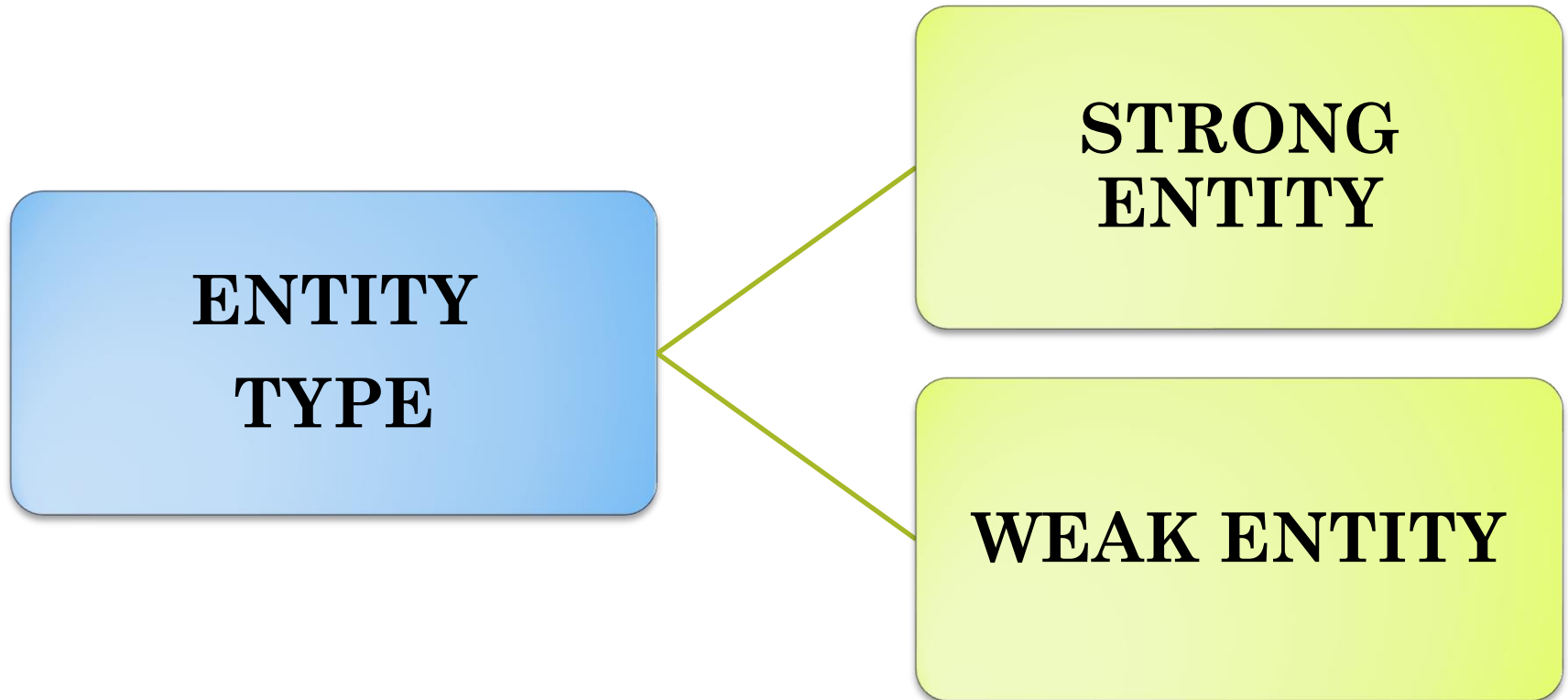
321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rye
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

customer

L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-19	500
L-11	900
L-16	1300

loan

TYPES OF ENTITY

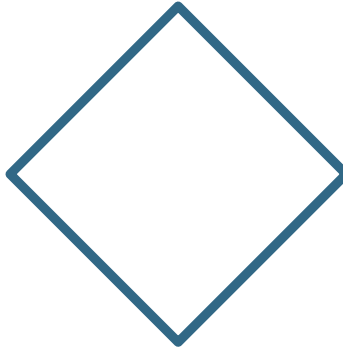


STRONG ENTITY TYPE

- ❖ Strong entity are those entity types which has a **key attribute**.
- ❖ The **primary key** helps in identifying each entity uniquely.
- ❖ Strong entity is represented by a rectangle.



- ❖ The relationship of two strong entities is represented by a single diamond.



- ❖ Various strong entities, when combined together, create a strong entity set.



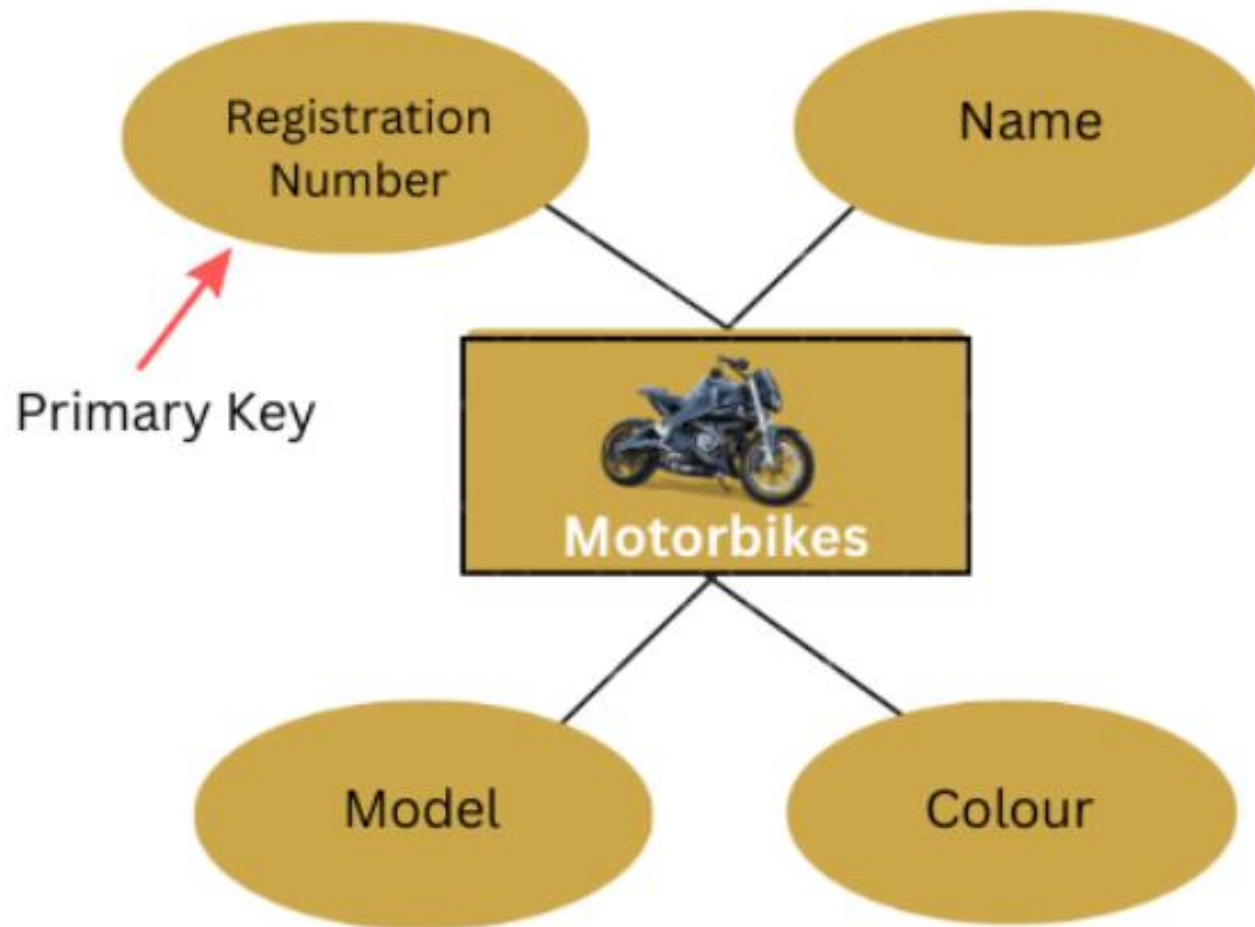
STRONG ENTITY



Motorbikes

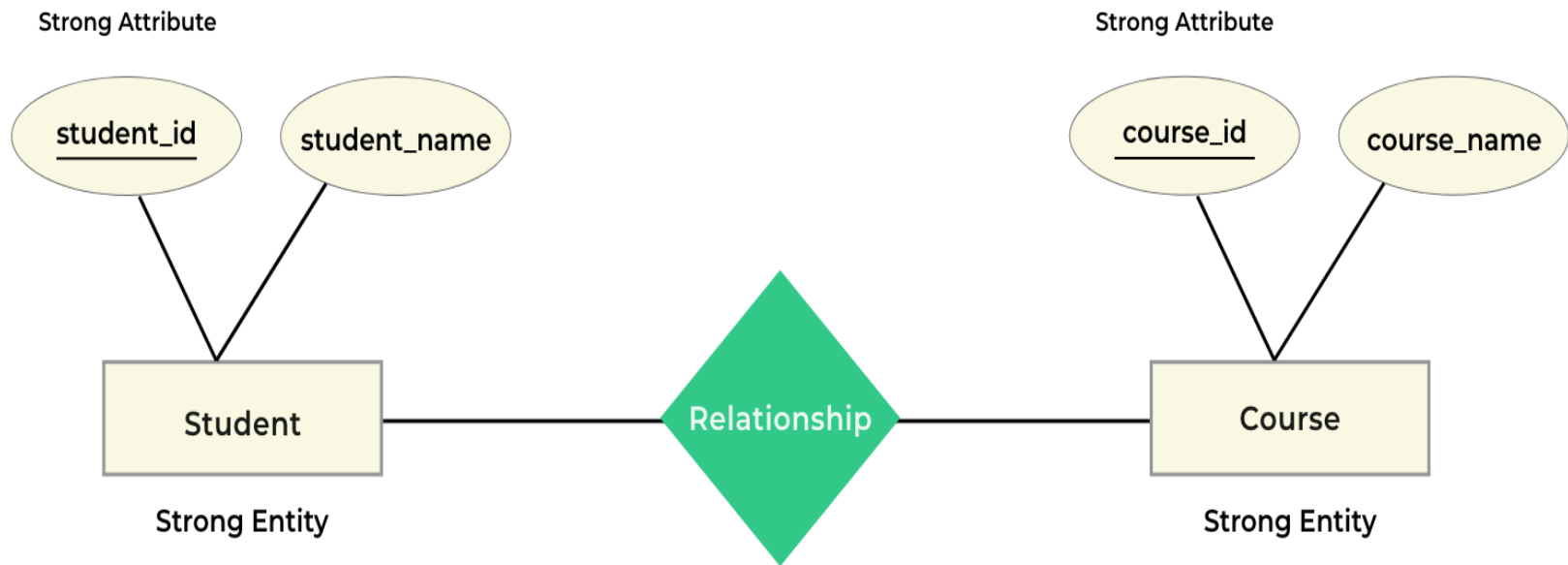
	REGISTRATION NUMBER	NAME	MODEL	COLOUR
Entity 1 →	UP515	Yamaha	R15S	Black
Entity 2 →	UP340	Honda	SP125	Red
Entity 3 →	UP414	KTM	RC8	Orange

ER DIAGRAM OF STRONG ENTITY



STRONG ENTITY EXAMPLE

Strong Entity



WEAK ENTITY

- ❖ A weak entity is an entity set that **do not have sufficient attributes for unique identification** of its records.
- ❖ Weak entity **do not have primary key.**
- ❖ A weak entity is depends on strong entity to ensure its existence.
- ❖ A set of weak entity is known as weak entity set.

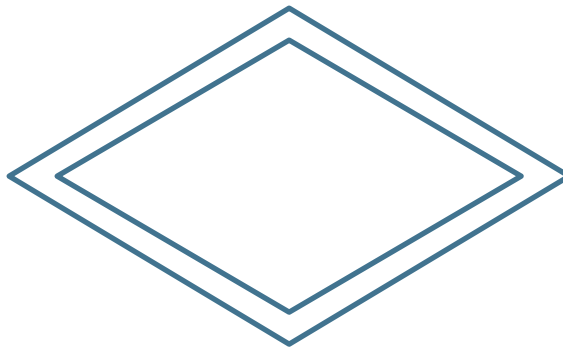


WEAK ENTITY

- ❖ A weak entity is represented by a double rectangle.



- ❖ It instead has a partial discriminator key.



- ❖ A partial key (discriminator key) of weak entity is marks as



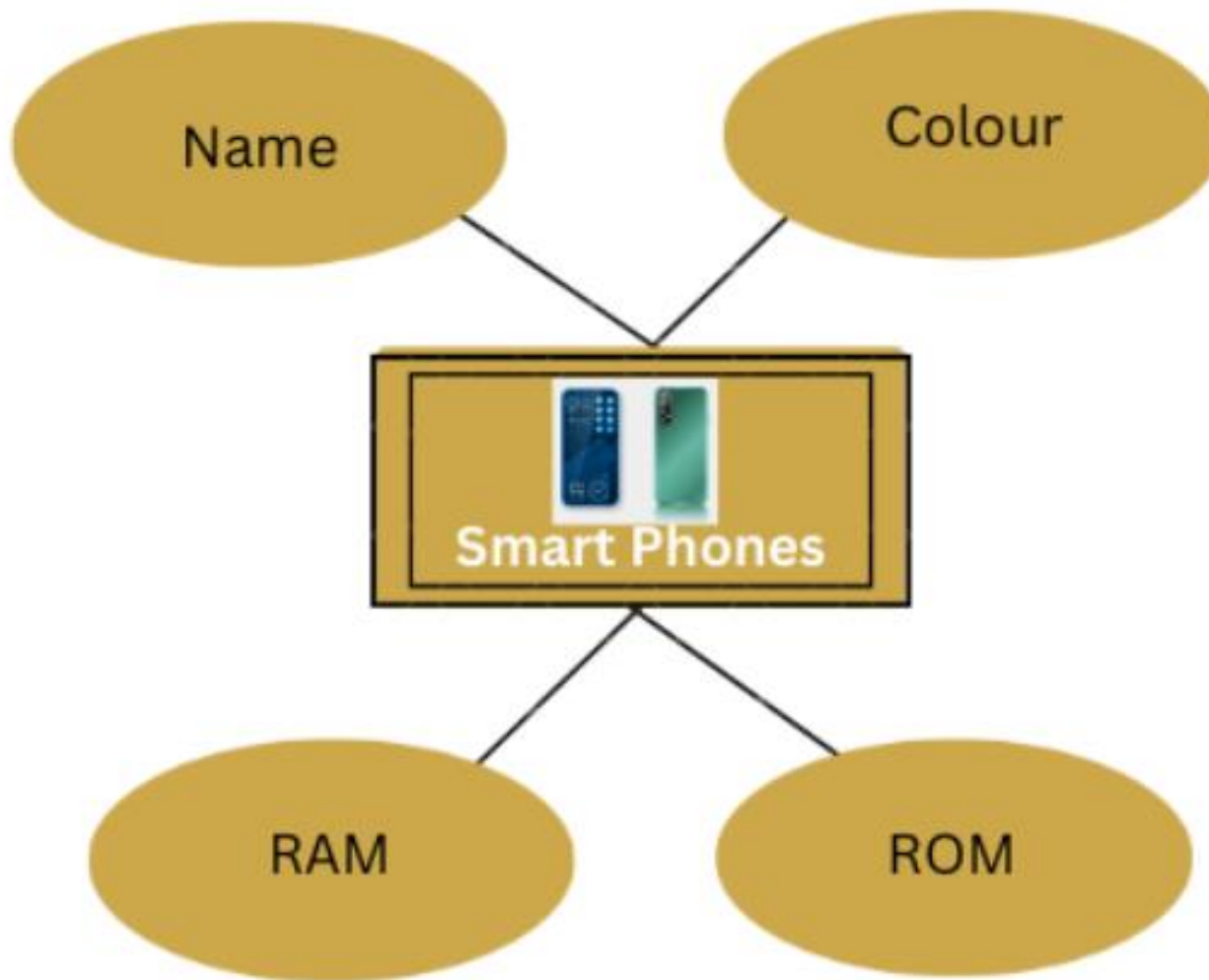
EXAMPLE OF WEAK ENTITY



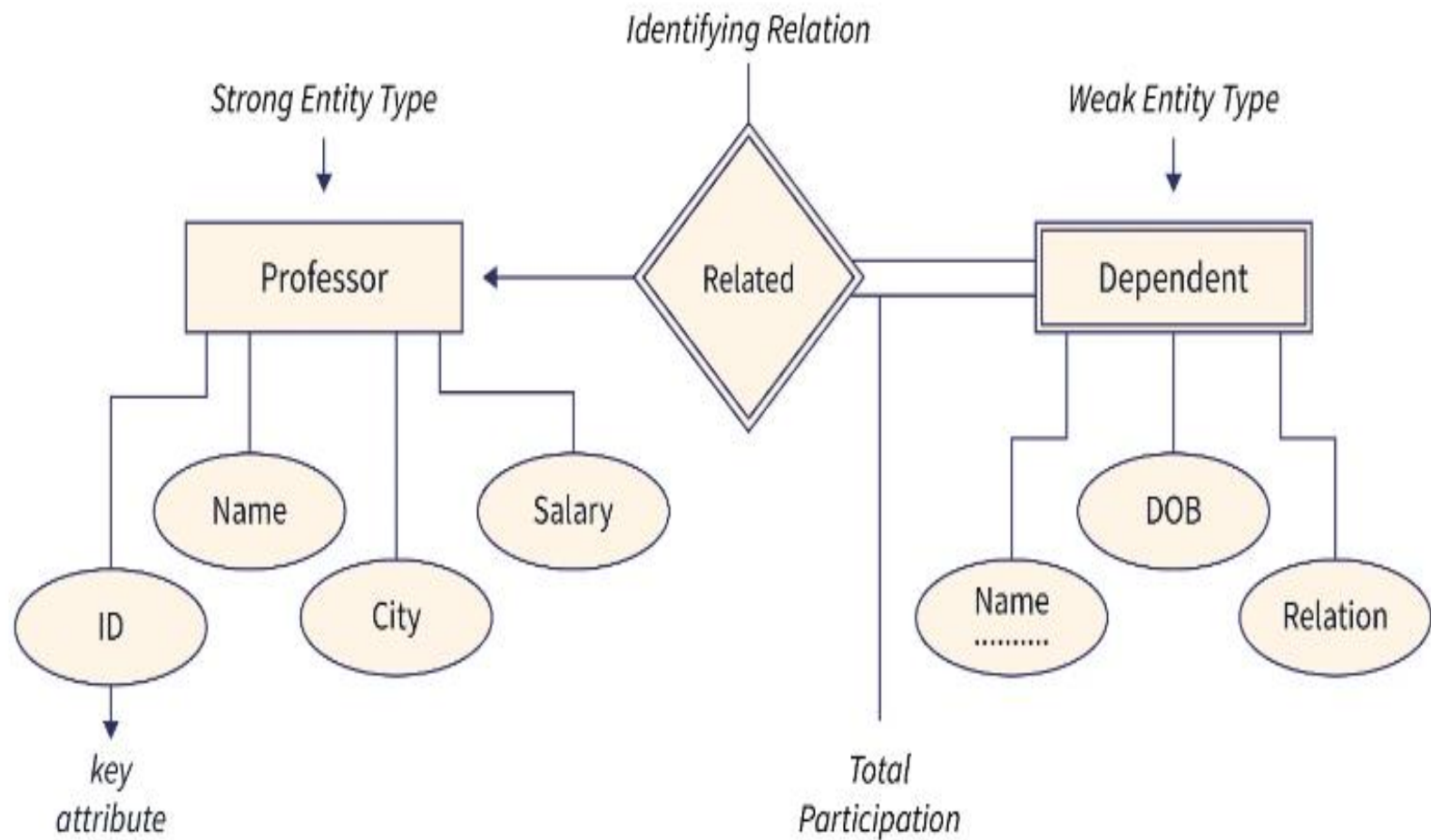
Smart Phones

	NAME	COLOUR	RAM	ROM
Entity 1 →	Samsung	Sea Green	4GB	64GB
Entity 2 →	Vivo	White	6GB	128GB
Entity 3 →	Redmi	Black	4GB	128GB

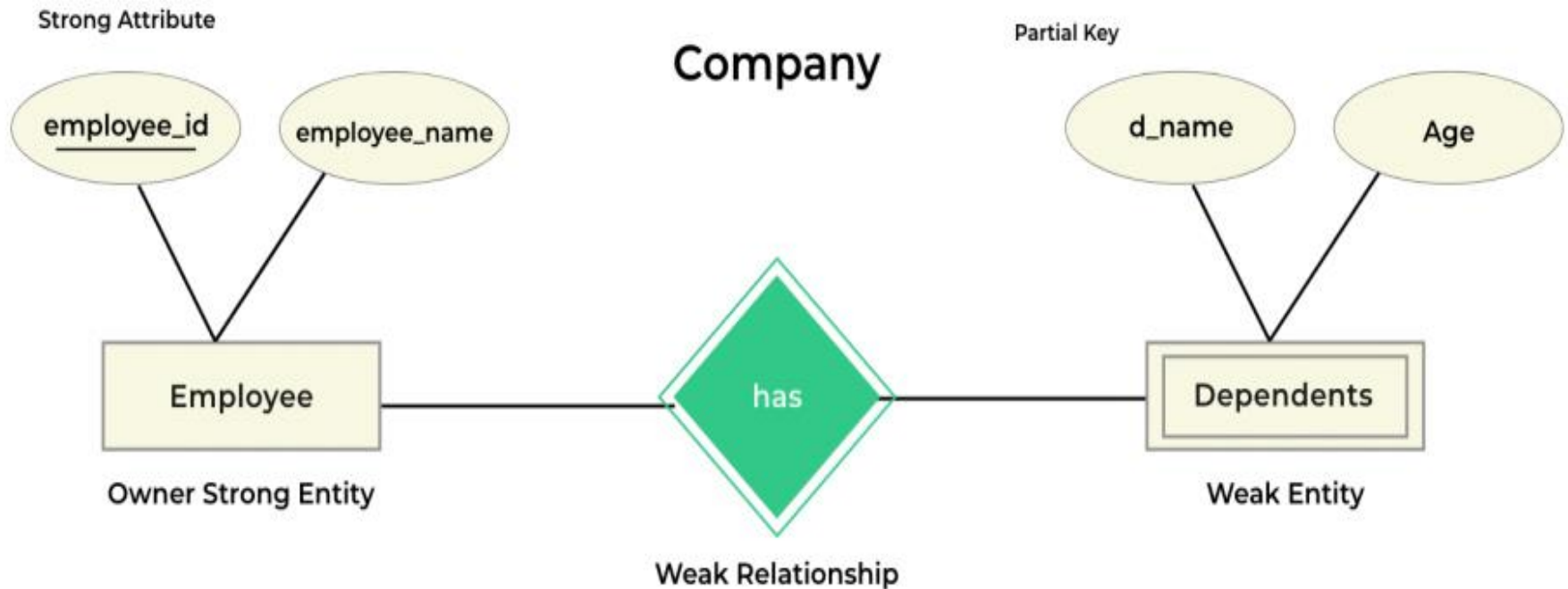
ER DIAGRAM OF WEAK ENTITY



WEAK ENTITY EXAMPLE : 1



WEAK ENTITY EXAMPLE : 2



Initialisation

Strong Entities	Employee
Weak Entities	Dependents
Strong Attributes	employee_id
Partial Key	d_name

== Total Participation

— Partial Participation

Weak Entity

Weak Relationship

Strong Entity

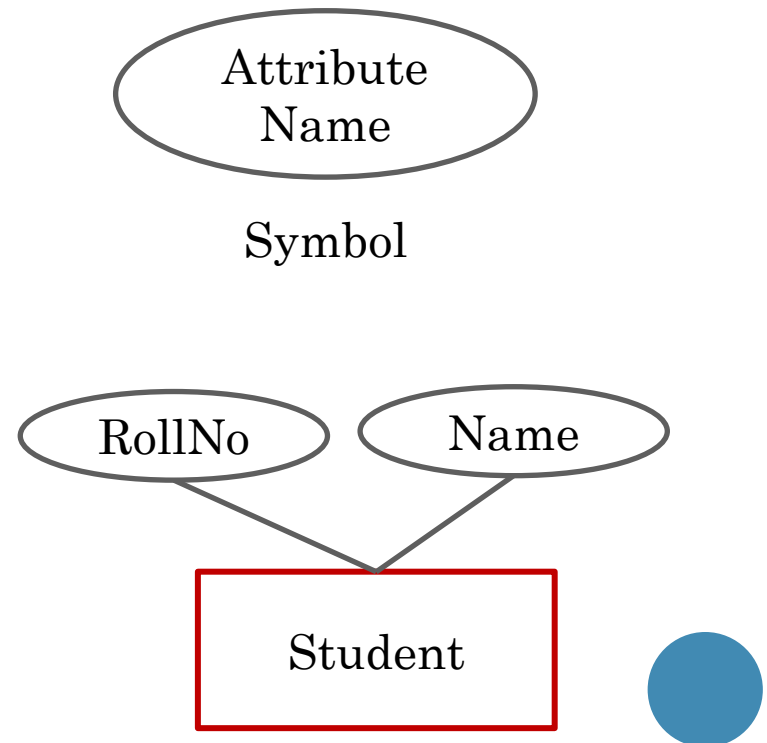
Attribute

ATTRIBUTES

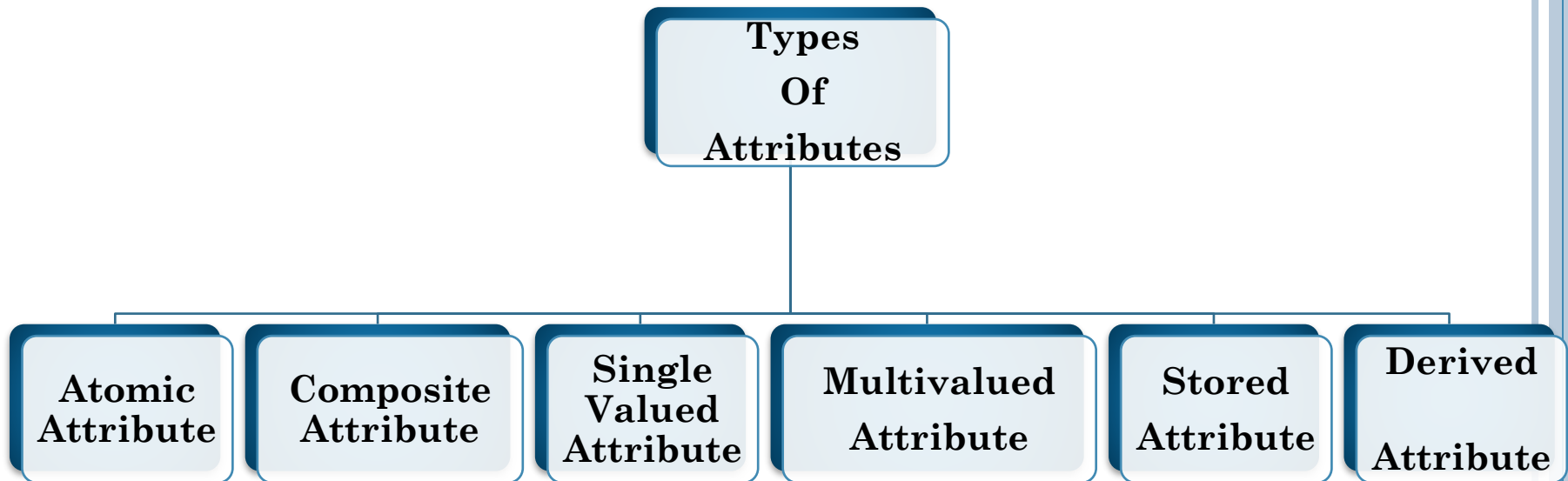


ATTRIBUTES

- ❖ Attribute is **properties** or details about an entity.
- ❖ An attribute is represented by an **oval** containing name of an attribute.
- ❖ Attributes of Student are:
 - ❖ Roll No
 - ❖ Student Name
 - ❖ Branch
 - ❖ Semester
 - ❖ Address
 - ❖ Mobile No
 - ❖ Age
 - ❖ SPI
 - ❖ Backlogs



TYPES OF ATTRIBUTES:



ATOMIC & COMPOSITE ATTRIBUTE

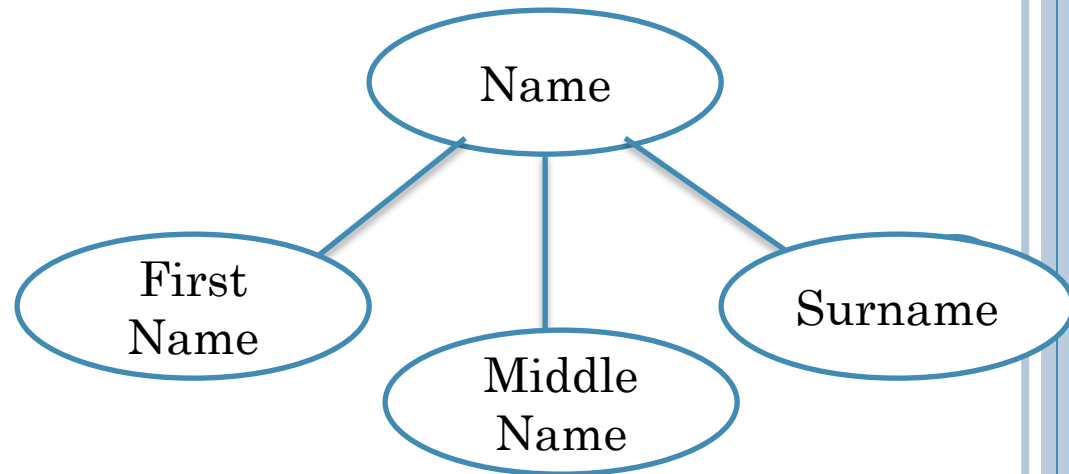
Atomic Attribute

- ❖ Attributes that are not divisible are called simple or atomic attributes.
- ❖ E.g: **Roll no**
- ❖ Symbol



Composite Attribute

- ❖ Composite attributes can be divided into smaller subparts.
- ❖ E.g. : **Name** (First name , middle name, surname)
- ❖ Symbol



SINGLE-VALUED & MULTIVALUE ATTRIBUTE

Single Valued Attribute

- ❖ Attributes have a single value for a particular entity
- ❖ E.g.: **Roll No**
- ❖ Symbol

Roll No

Multi-valued Attribute

- ❖ Attributes have a more than 1 value for a particular entity.
- ❖ E.g. : **Mobile No**
- ❖ Symbol

Mobile
No

STORED VALUE & DERIVED VALUE ATTRIBUTE

Stored Value Attribute

- ❖ Attributes whose values are stored manually in database.
- ❖ E.g.: **Birth Date**
- ❖ Symbol

Birth Date



Derived value Attribute

- ❖ Attribute whose value is **derived** or **calculated** from other attributes
- ❖ E.g. : **Age**
- ❖ Symbol

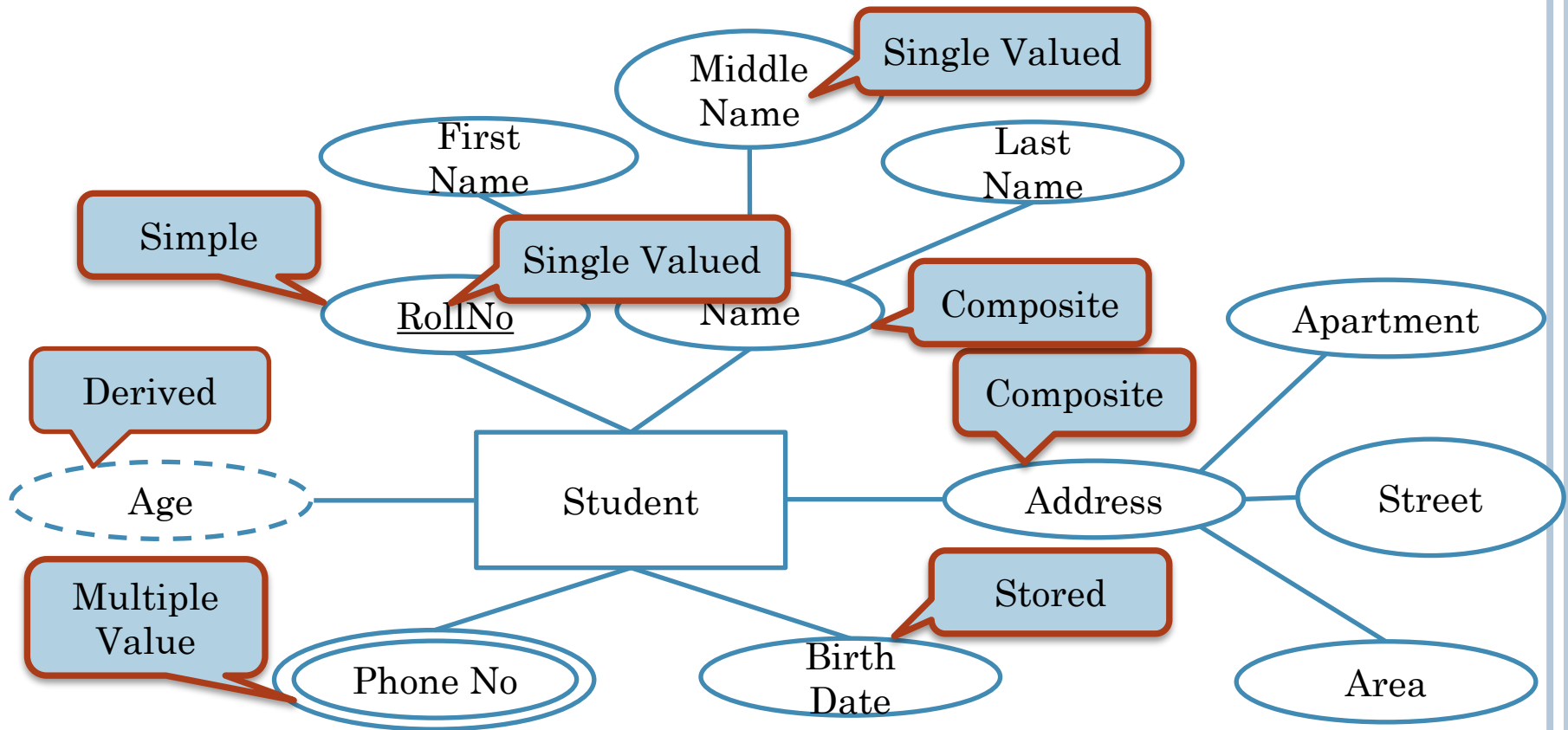
Age



Exercise :
1

IDENTIFY (FIND OUT) THE FOLLOWING FROM THE GIVEN E-R DIAGRAM (WITH SYMBOL).

(A) SIMPLE ATTRIBUTE (B) COMPOSITE ATTRIBUTE
(C) SINGLE VALUED ATTRIBUTE (D) MULTI VALUED
(E) STORED ATTRIBUTE (F) DERIVED ATTRIBUTE



Exercise : 2

IDENTIFY (FIND OUT) THE FOLLOWING FROM THE GIVEN E-R DIAGRAM (WITH SYMBOL).

(A) SIMPLE ATTRIBUTE

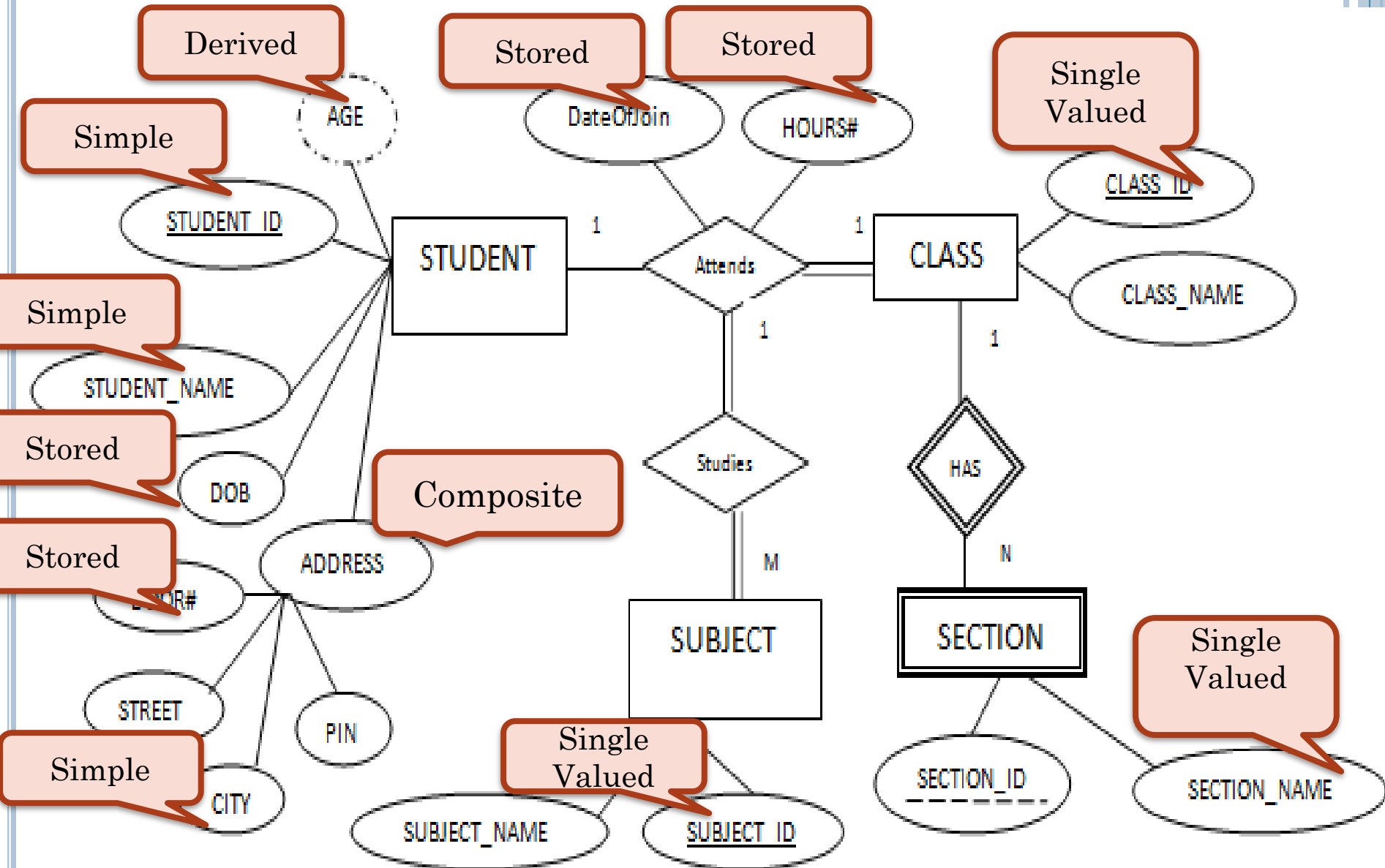
(B) COMPOSITE ATTRIBUTE

(C) SINGLE VALUED ATTRIBUTE

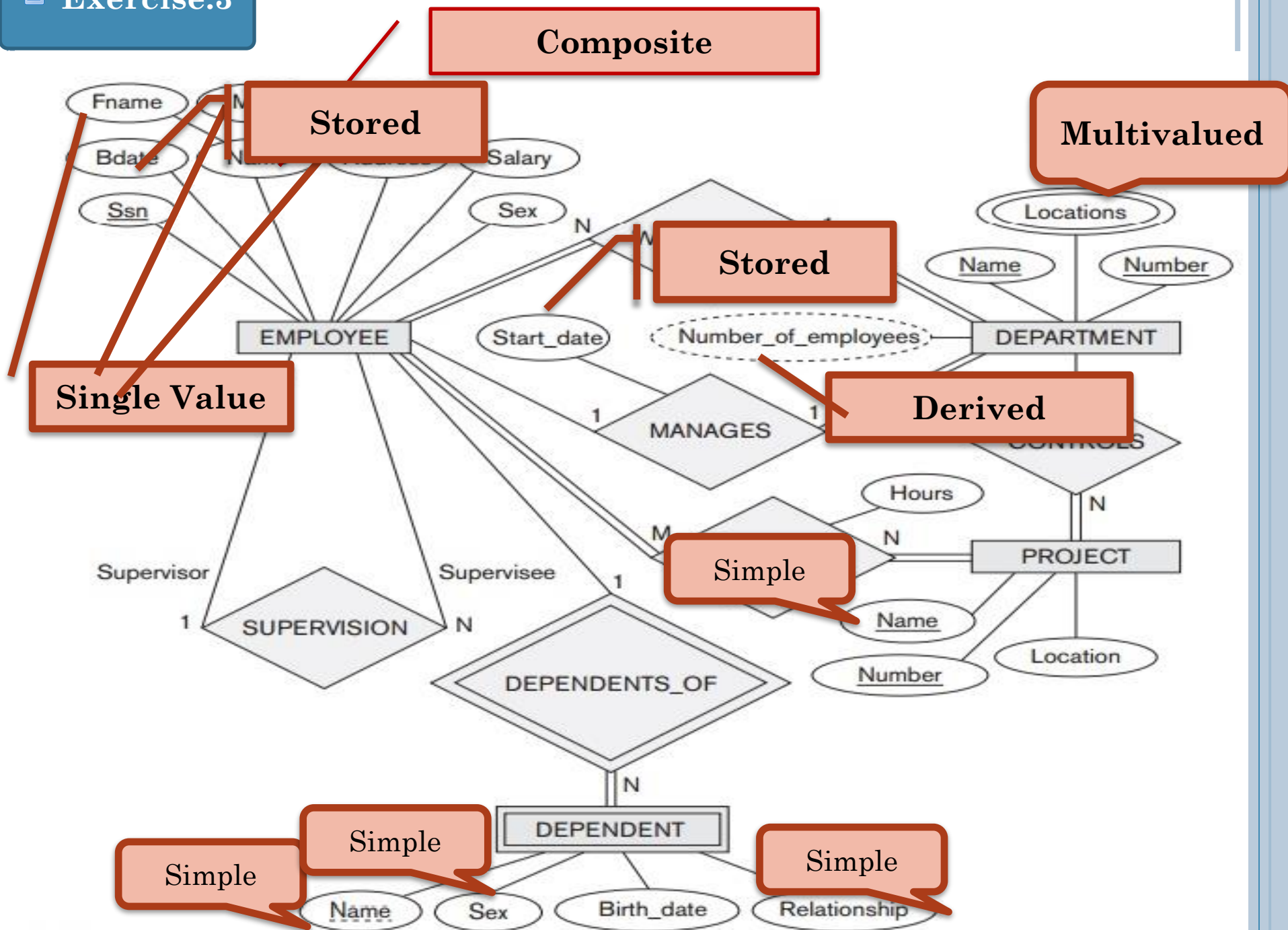
(D) MULTI VALUED

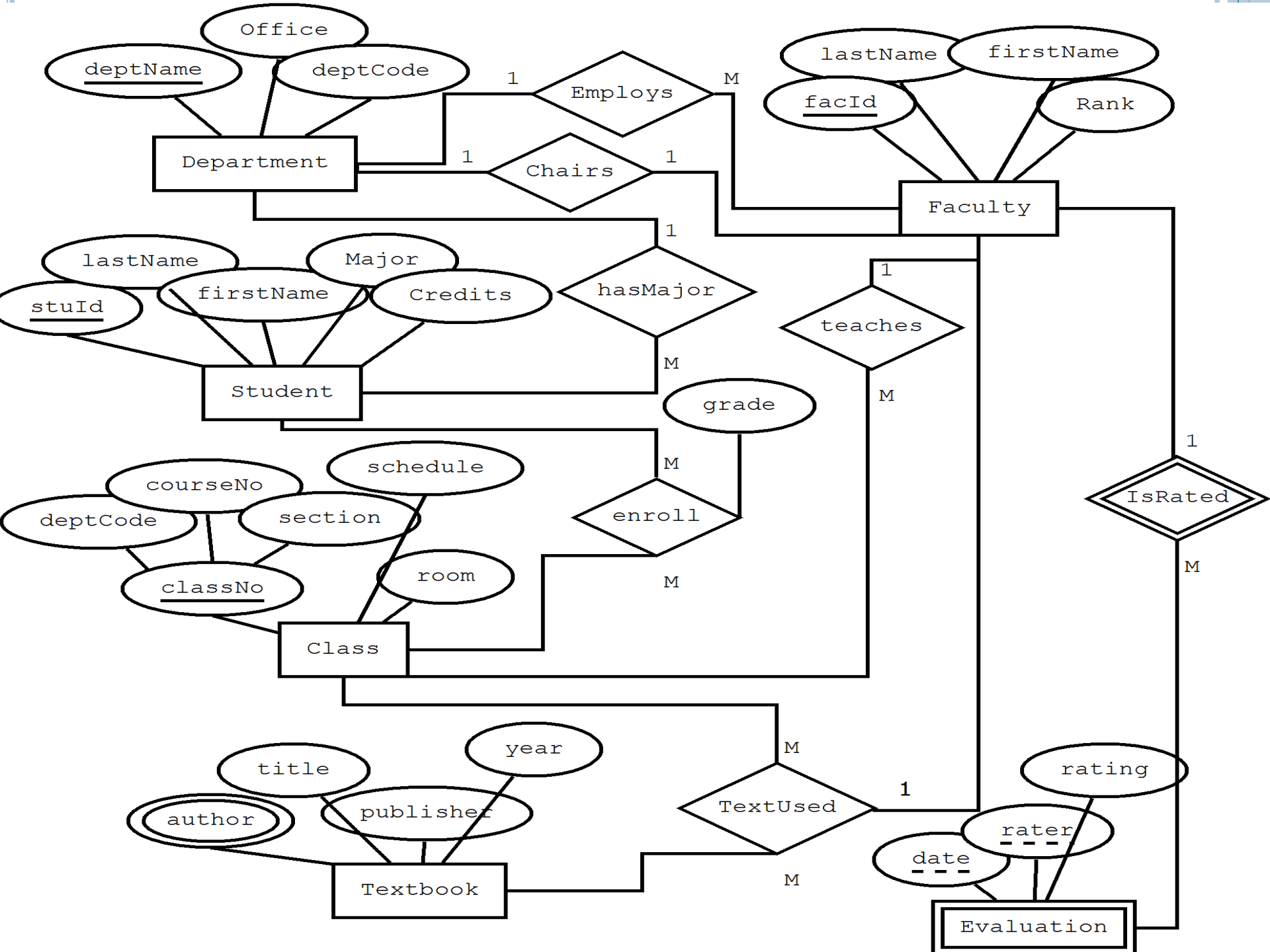
(E) STORED ATTRIBUTE

(F) DERIVED ATTRIBUTE



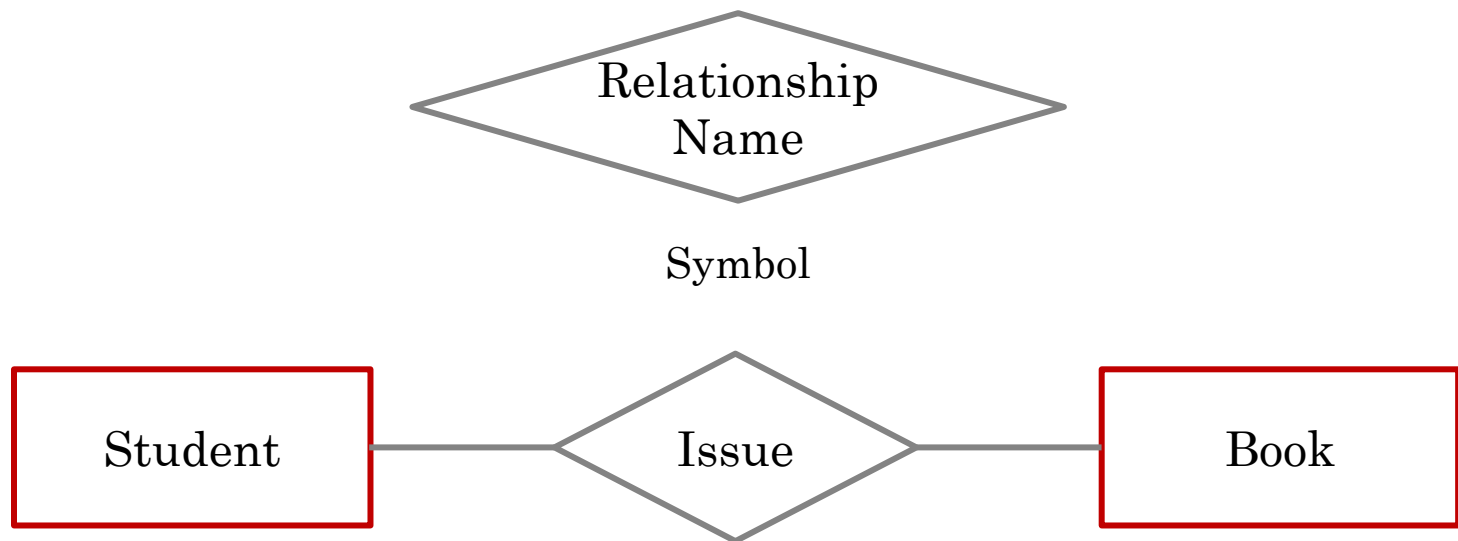
Exercise:3



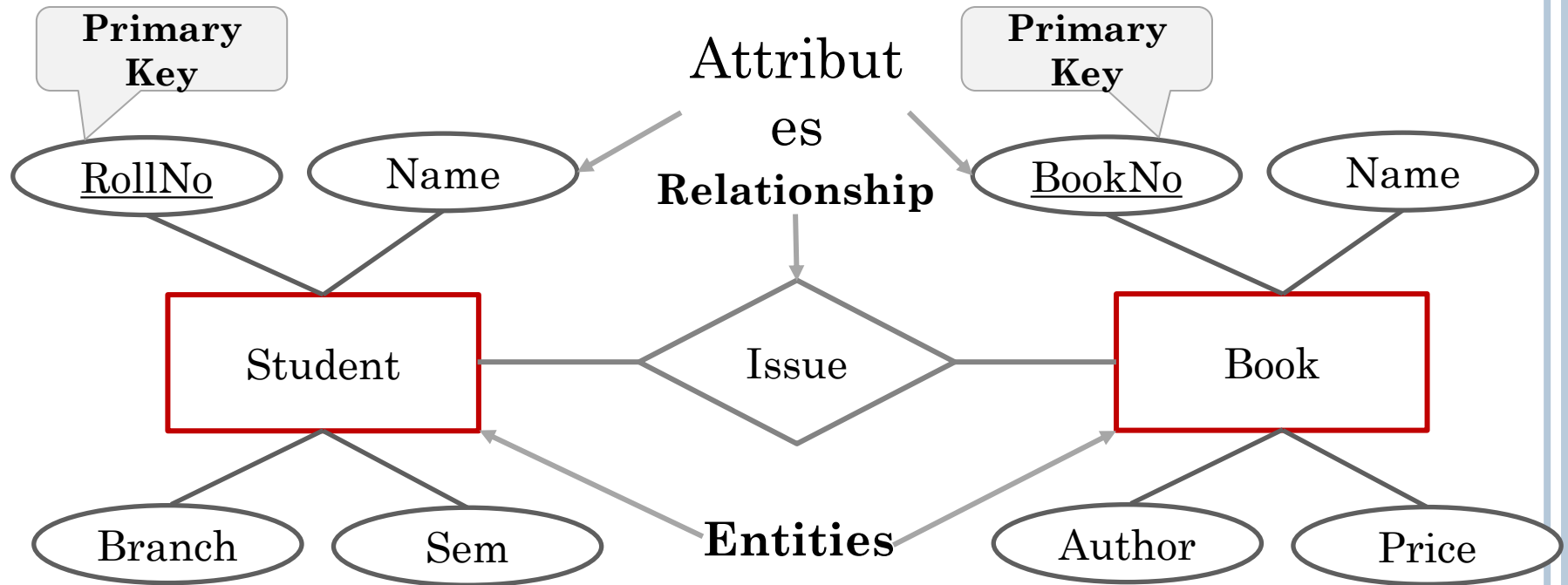


RELATIONSHIP

- ❖ Relationship is an **association (connection)** between several entities.
- ❖ It should be placed between two entities and a line connecting it to an entity.
- ❖ A relationship is represented by a **diamond** containing relationship's name.



E-R DIAGRAM OF A LIBRARY SYSTEM



- ❖ Each and every entity **must have one primary key** attribute.
- ❖ **Relationship between 2 entities** is called **binary relationship**.

BINARY RELATIONSHIP

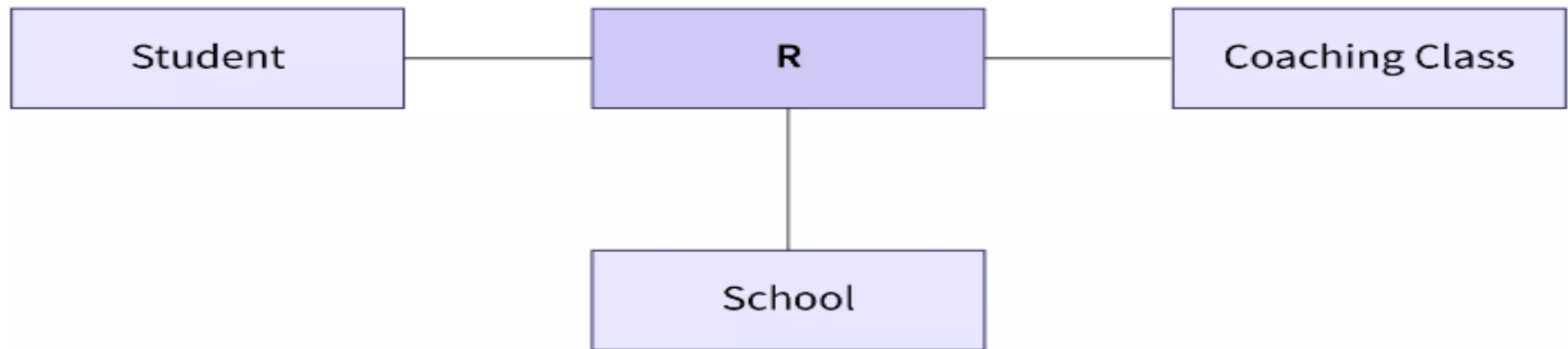


- ❖ In a binary relationship, there are two entities involved.
- ❖ There are two entities that are participating in the relationship.
- ❖ The degree of relationship is 2. This is the most common type of relationship.

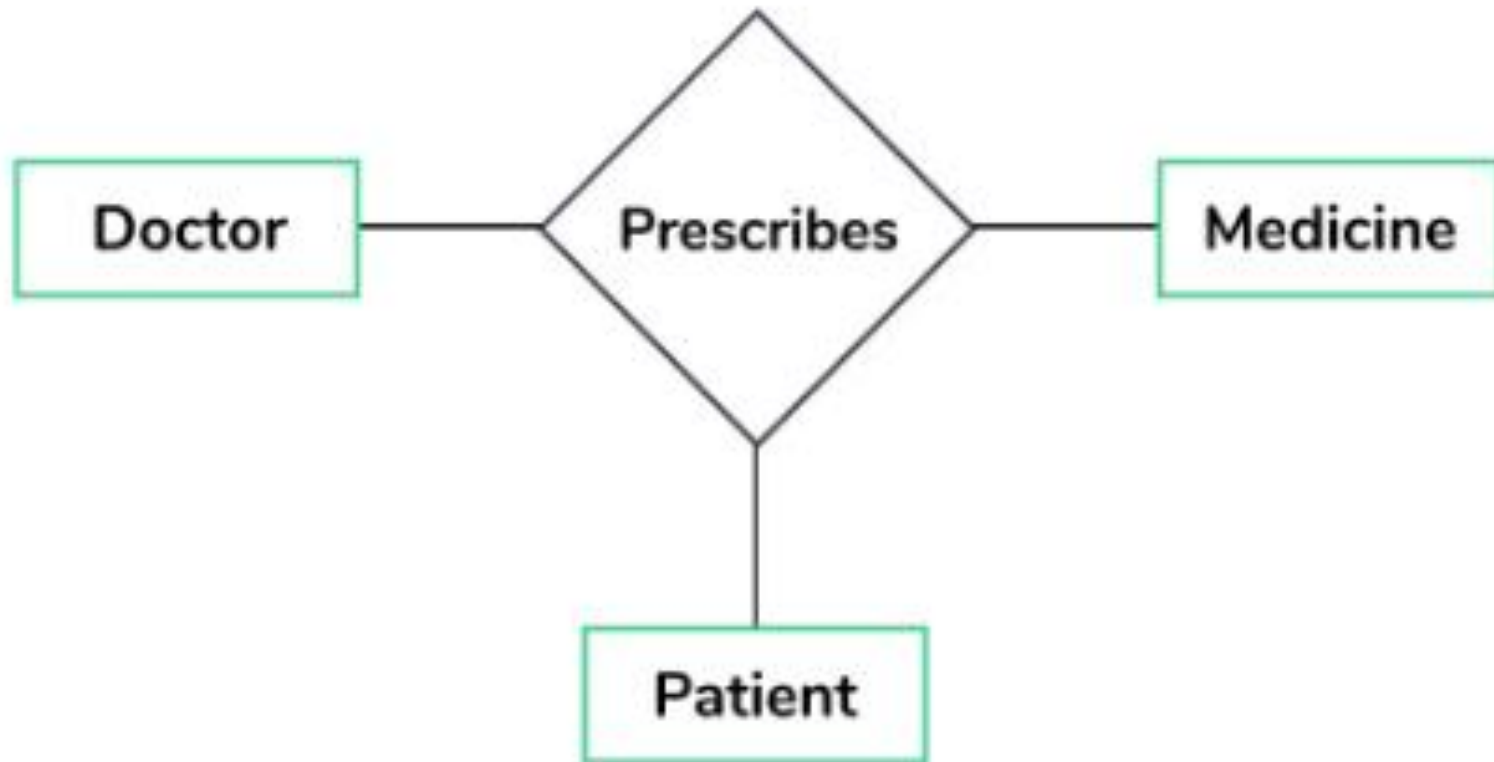


TERNARY RELATIONSHIP

- ❖ A relationship represents the association between two or more entities.
- ❖ When there are exactly three entity sets participating in a relationship then such type of relationship is called ternary relationship.

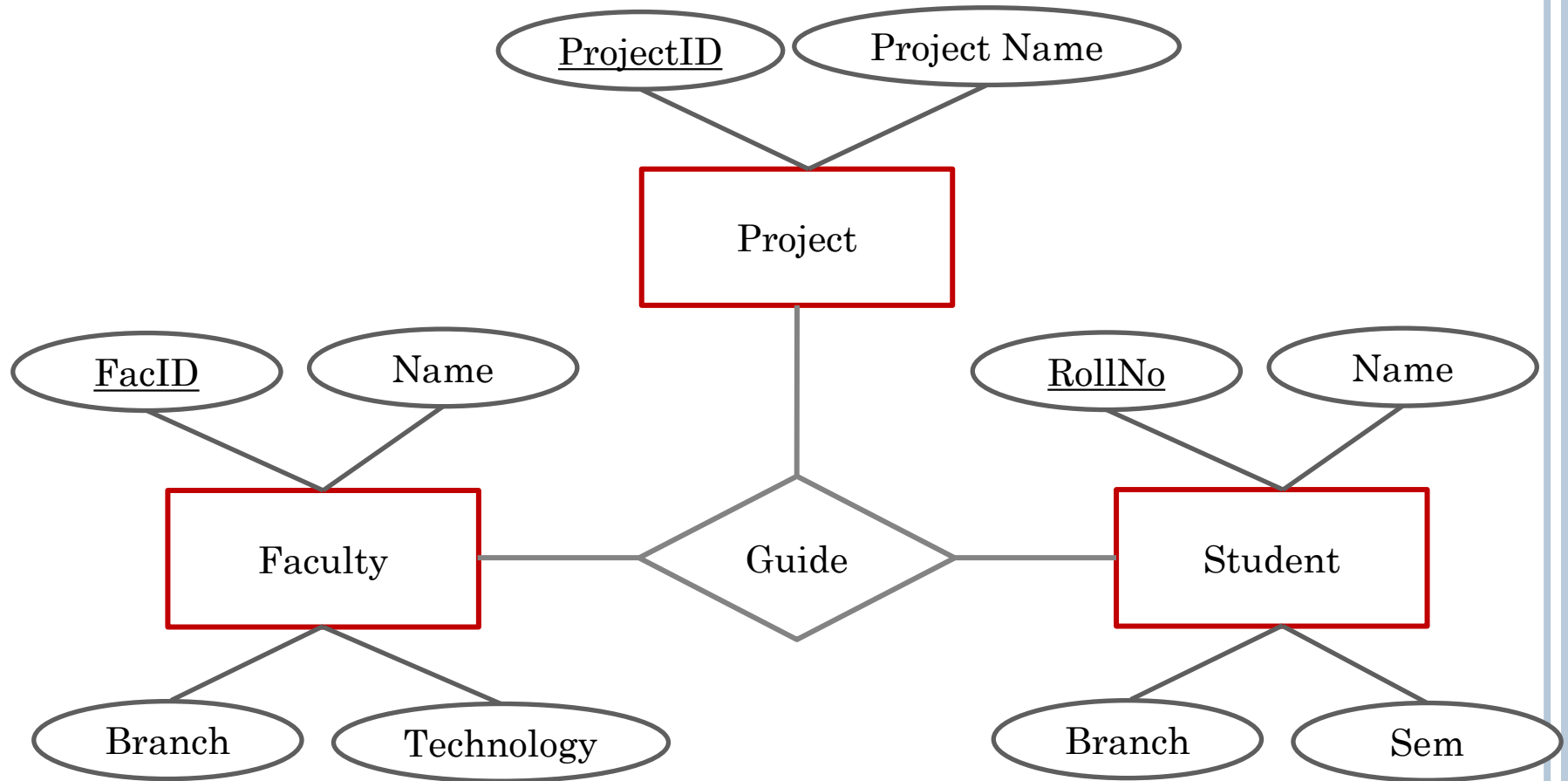


TERNARY RELATIONSHIP



In the real world, a patient goes to a doctor and doctor prescribes the medicine to the patient, **three entities Doctor, patient and medicine are involved in the relationship “prescribes”**

TERNARY RELATIONSHIP



Relationship between 3 entities is called **ternary relationship**.

MAPPING CARDINALITY



MAPPING CARDINALITY (CARDINALITY CONSTRAINTS) DIFFERENT TYPES OF RELATIONSHIPS

- ❖ It represents the **number of entities of another entity set** which are **connected to an entity** using a relationship.
- ❖ It is most useful in describing binary relationship sets.
- ❖ Cardinality is possible with **key field** in entity sets.
- ❖ For a binary relationship set the mapping cardinality must be one of the following types:
 1. One to One
 2. One to Many
 3. Many to One
 4. Many to Many



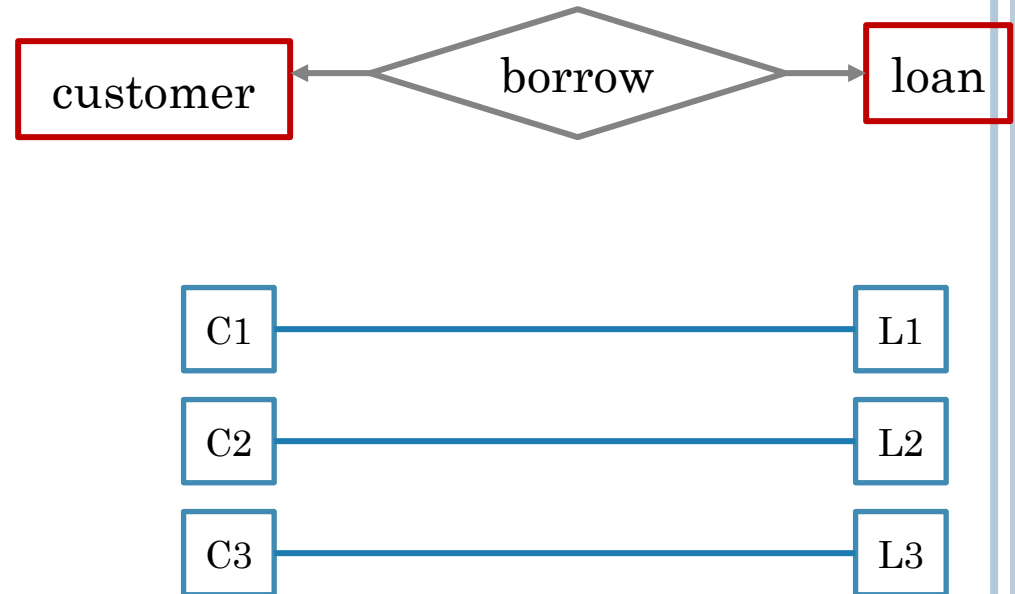
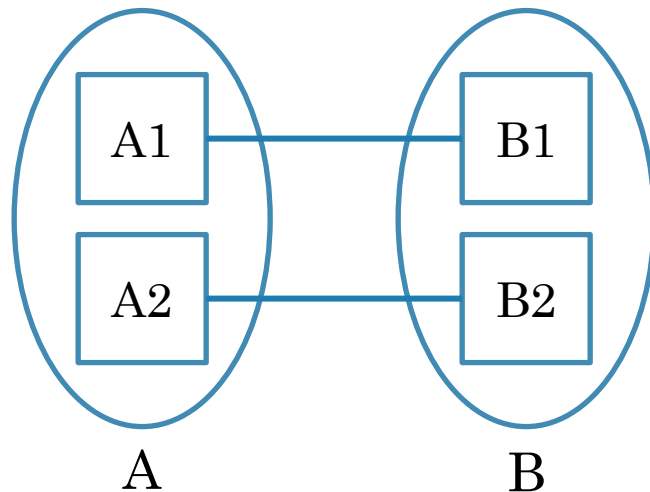
KEY FIELD

- ❖ A key field that used to create cardinality / relationship between two or more entities.
- ❖ The basic rules for key field are:
 - ❖ Key field in entity set must give unique identification for any entity.
 - ❖ Key field in entity set must not be blank in entity.



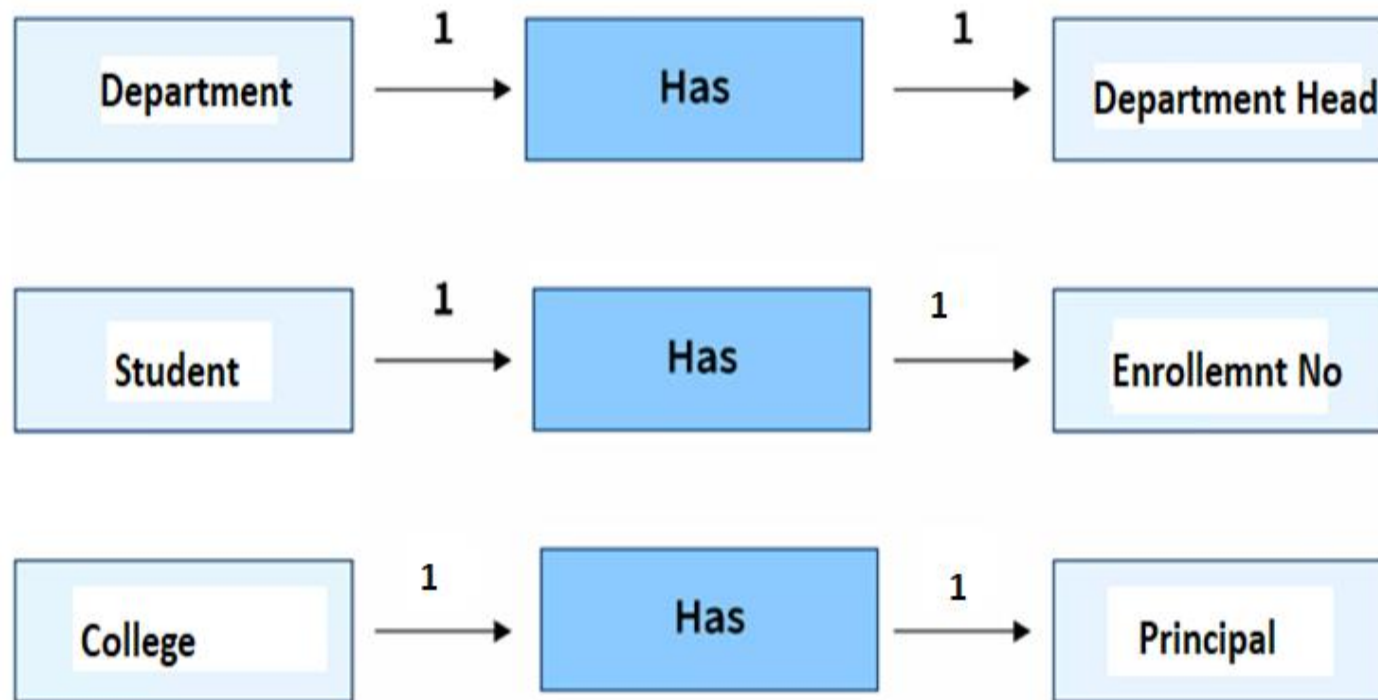
ONE-TO-ONE RELATIONSHIP (1 – 1)

- ❖ An entity in **A** is associated with only one entity in **B** and an entity in **B** is associated with only one entity in **A**.

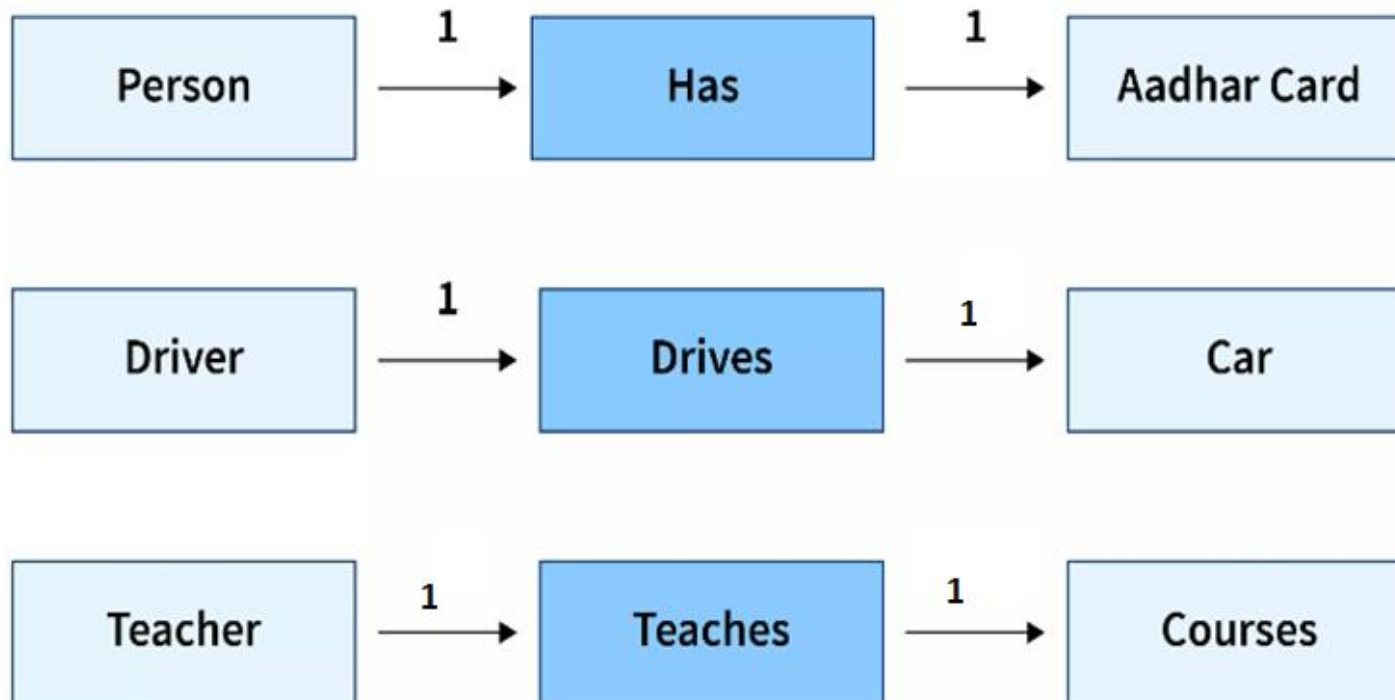


- ❖ **Example:** A **customer** is connected with only one loan using the relationship borrower and a **loan** is connected with only one customer using borrower.

ONE TO ONE RELATIONSHIP (1 – 1)

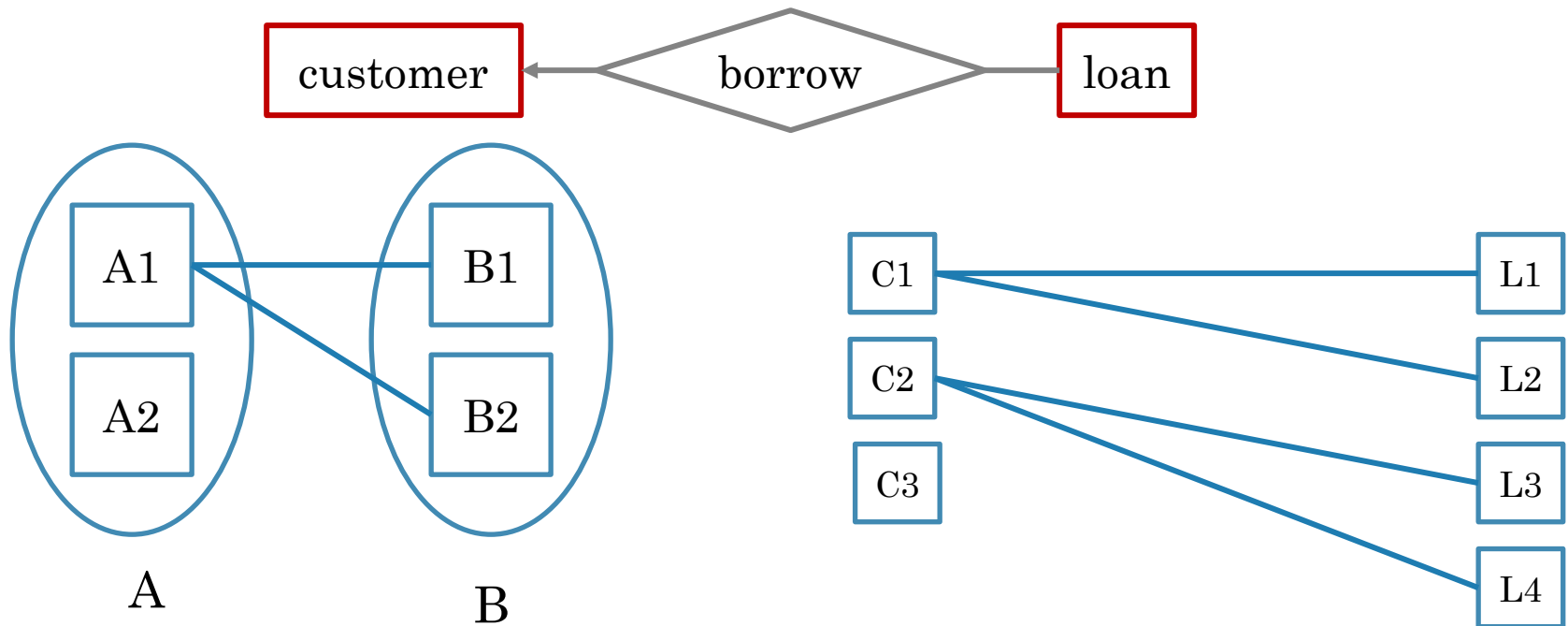


ONE TO ONE RELATIONSHIP (1 – 1)



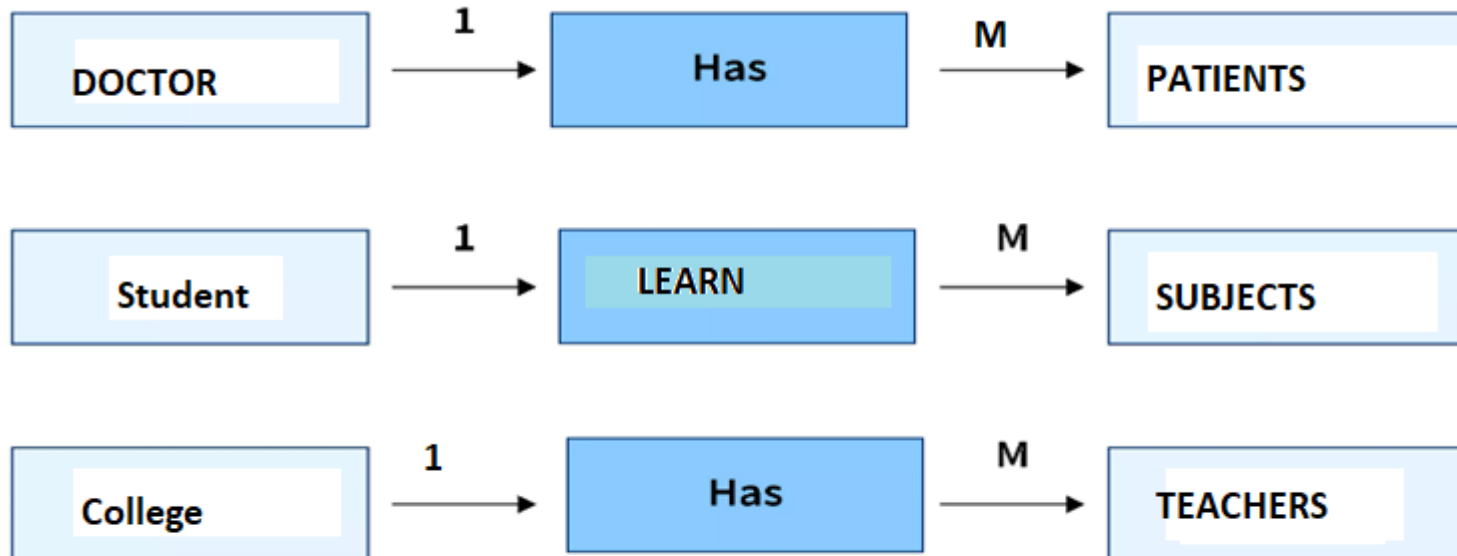
ONE-TO-MANY RELATIONSHIP (1 – M)

- ❖ An entity in **A** is associated with more than one entities in **B** and an entity in **B** is associated with only one entity in **A**.

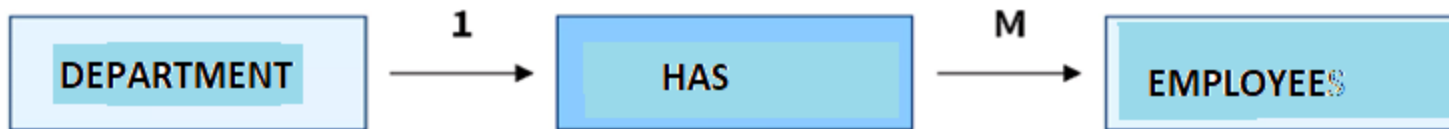
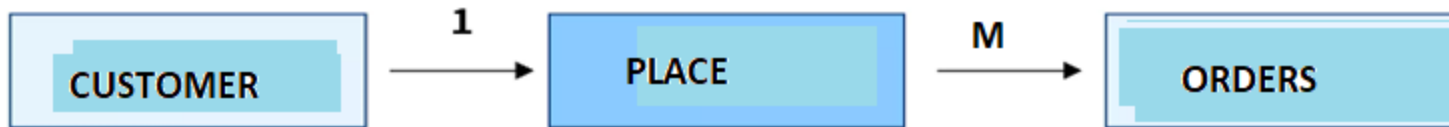


- ❖ **Example:** A **loan** is connected with only one customer using borrower and a **customer** is connected with more than one **loans** using borrower.

ONE-TO-MANY RELATIONSHIP (1 – M)

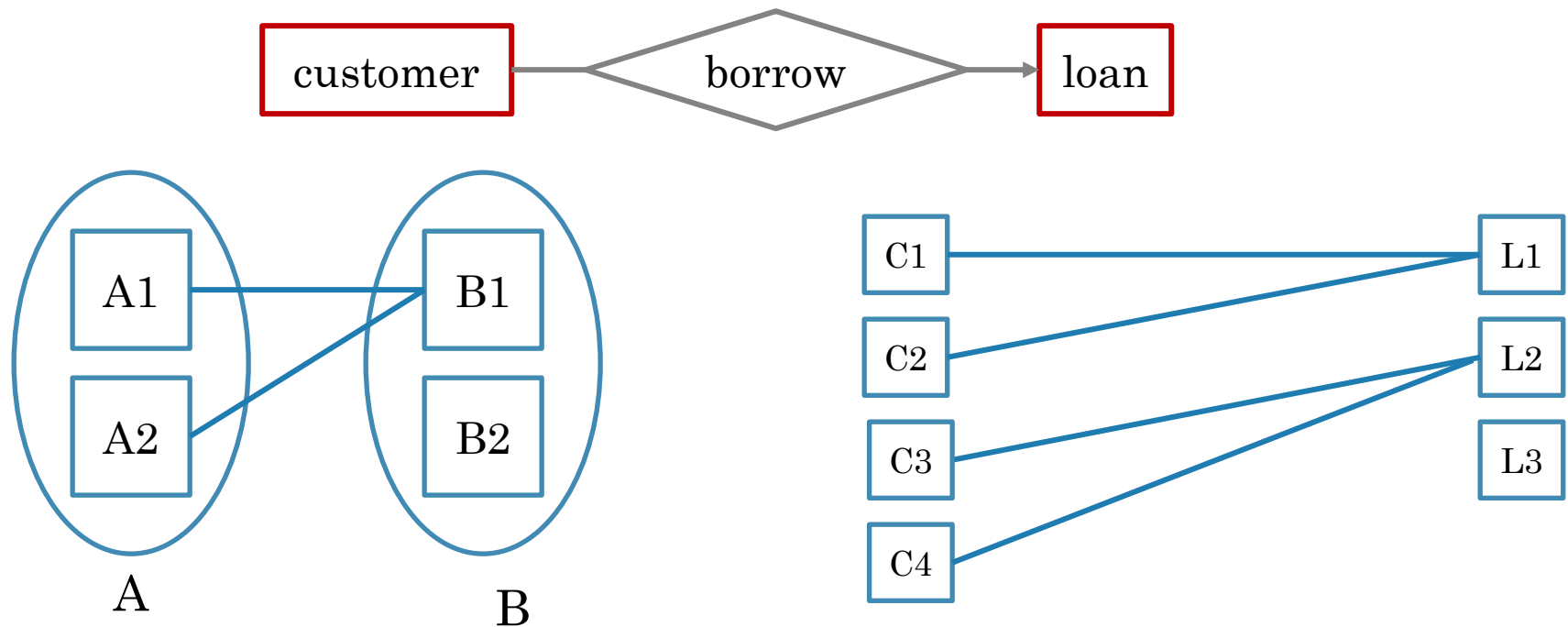


ONE-TO-MANY RELATIONSHIP (1 – M)



MANY-TO-ONE RELATIONSHIP (N/M – 1)

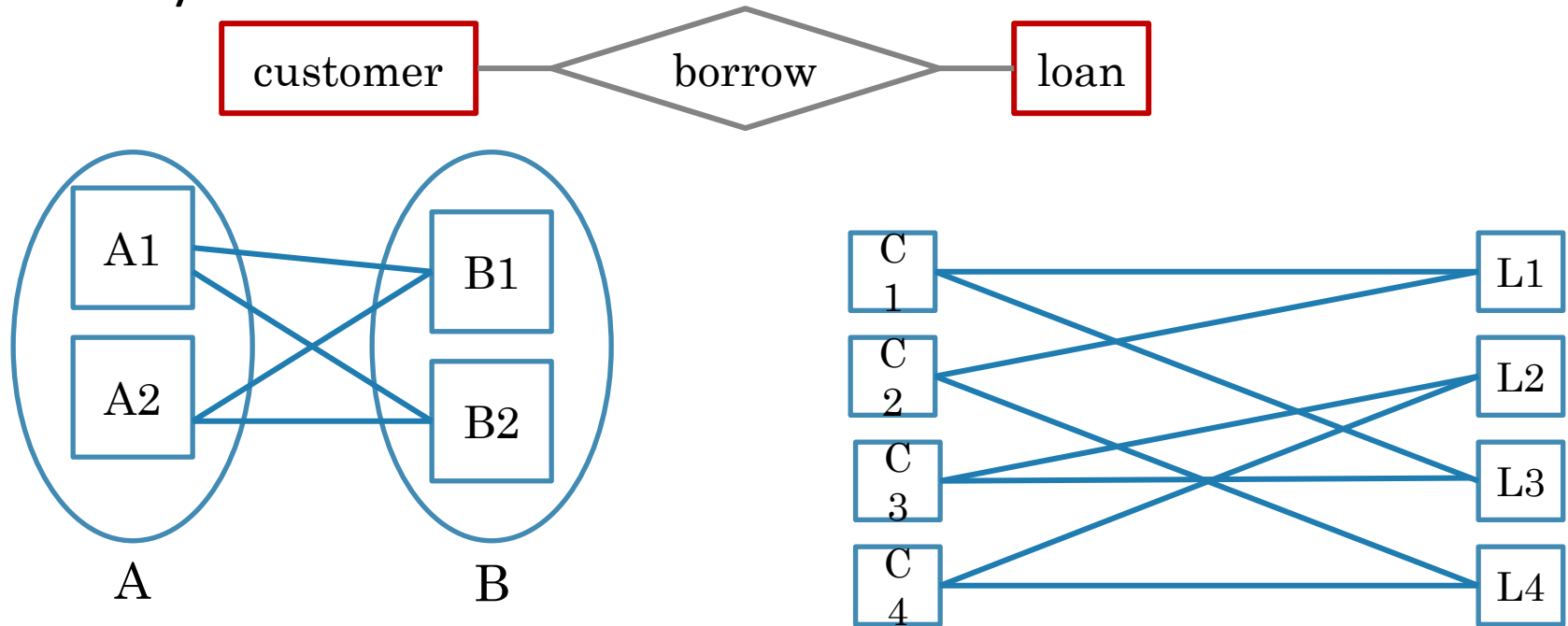
- ❖ An entity in **A** is associated with only one entity in **B** and an entity in **B** is associated with more than one entities in **A**.



- ❖ **Example:** A **loan** is connected with more than one customer using borrower and a **customer** is connected with only one **loan** using borrower.

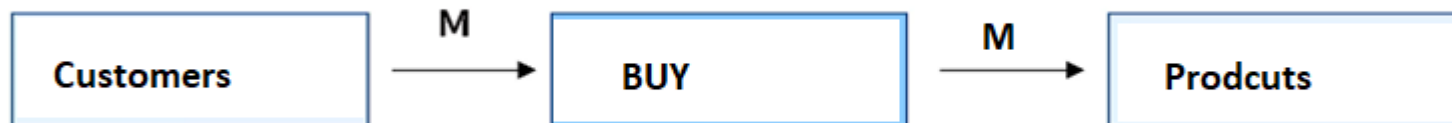
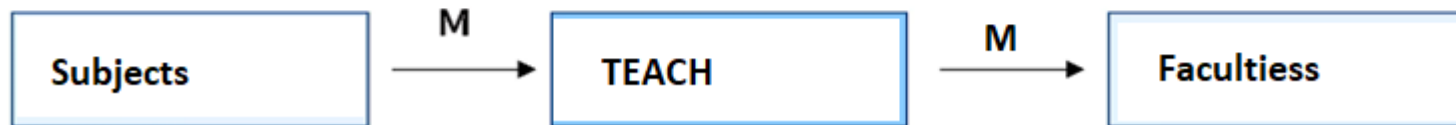
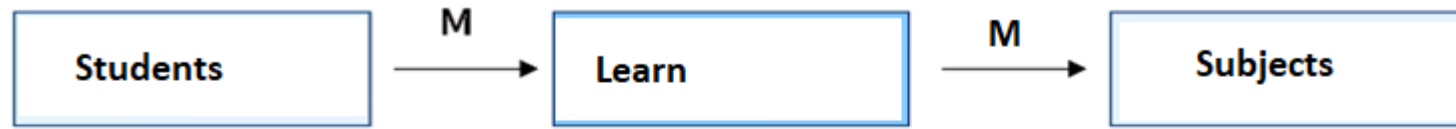
MANY-TO-MANY RELATIONSHIP (N – N)

- ❖ An entity in **A** is associated with more than one entities in **B** and an entity in **B** is associated with more than one entities in **A**.



- ❖ **Example:** A **customer** is connected with more than one **loan** using borrower and a **loan** is connected with more than one **customer** using borrower.


MANY-TO-MANY RELATIONSHIP (N – N)/(M-M)



ER DIAGRAM ILLUSTRATION



ER DIAGRAM ILLUSTRATION

- ❖ Bank have Customer.
 - ❖ Banks are identified by a name, code, address of main office.
 - ❖ Banks have branches.
 - ❖ Branches are identified by a branch_no., branch_name, address.
 - ❖ Customers are identified by name, cust-id, phone number, address.
 - ❖ Customer can have one or more accounts.
 - ❖ Accounts are identified by account_no., acc_type, balance.
 - ❖ Customer can avail loans.
 - ❖ Loans are identified by loan_id, loan_type and amount.
 - ❖ Account and loans are related to bank's branch.
- 

ENTITIES AND THEIR ATTRIBUTES

- ❖ Bank Entity : Attributes of Bank Entity are Bank Name, Code and Address.

Code is Primary Key for Bank Entity.

- ❖ Customer Entity : Attributes of Customer Entity are Customer_id, Name, Phone Number and Address.

Customer_id is Primary Key for Customer Entity.



ENTITIES AND THEIR ATTRIBUTES

- ❖ Branch Entity : Attributes of Branch Entity are Branch_id, Name and Address.

Branch_id is Primary Key for Branch Entity.

- ❖ Account Entity : Attributes of Account Entity are Account_number, Account_Type and Balance.

Account_number is Primary Key for Account Entity.

- ❖ Loan Entity : Attributes of Loan Entity are Loan_id, Loan_Type and Amount.

Loan_id is Primary Key for Loan Entity.



RELATIONSHIPS ARE

Bank has Branches => 1 : N

One Bank can have many Branches but one Branch can not belong to many Banks, so the relationship between Bank and Branch is one to many relationship.

Branch maintain Accounts => 1 : N

One Branch can have many Accounts but one Account can not belong to many Branches, so the relationship between Branch and Account is one to many relationship.

Branch offer Loans => 1 : N

One Branch can have many Loans but one Loan can not belong to many Branches, so the relationship between Branch and Loan is one to many relationship.



RELATIONSHIPS ARE

Account held by Customers => M : N

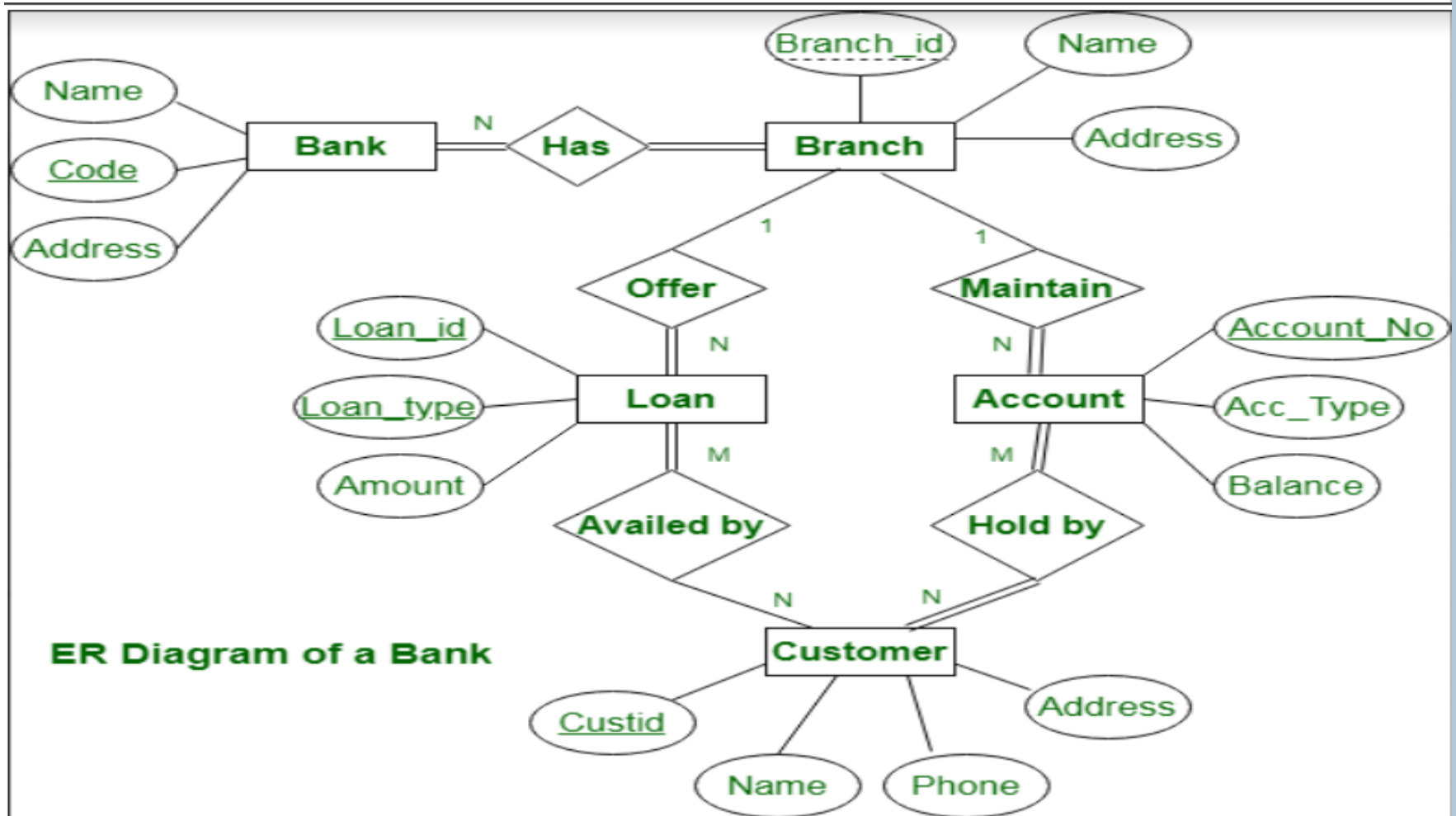
One Customer can have more than one Accounts and also One Account can be held by one or more Customers, so the relationship between Account and Customers is many to many relationship.

Loan availed by Customer => M : N

(Assume loan can be jointly held by many Customers).
One Customer can have more than one Loans and also One Loan can be availed by one or more Customers, so the relationship between Loan and Customers is many to many relationship.



RELATIONSHIPS ARE :

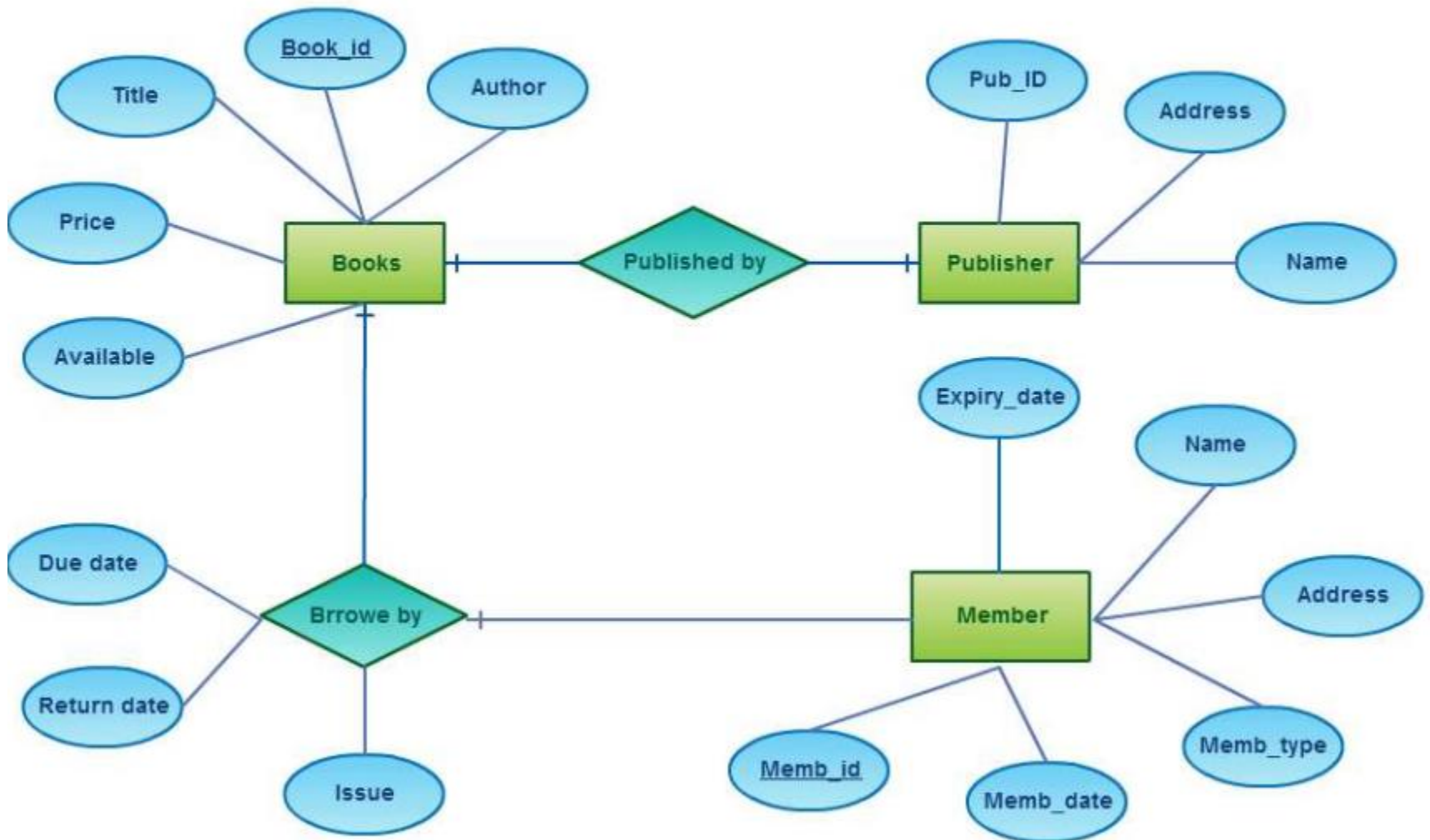


EXAMPLE OF ER DIAGRAM

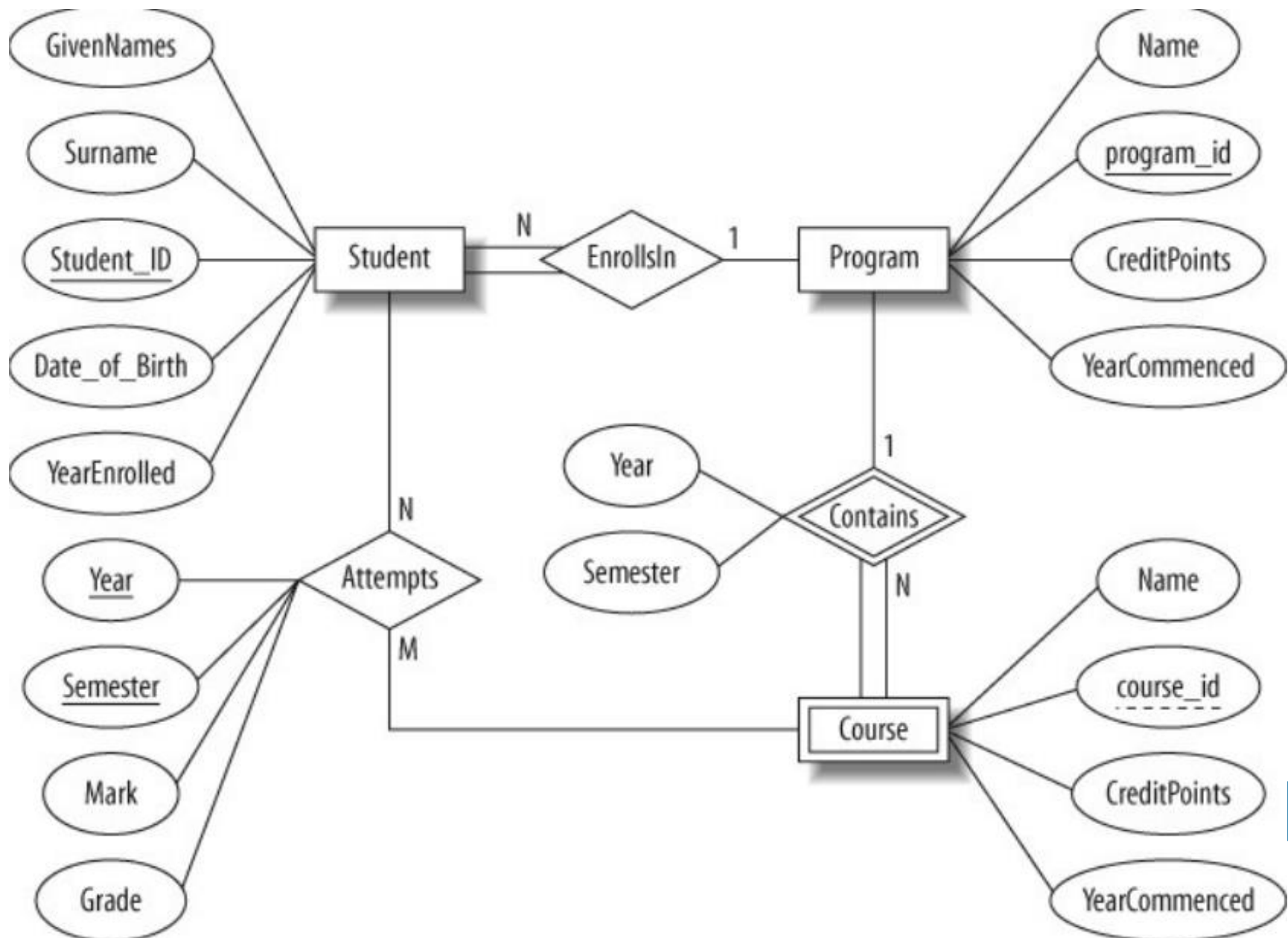


LIBRARY MANAGEMENT SYSTEM

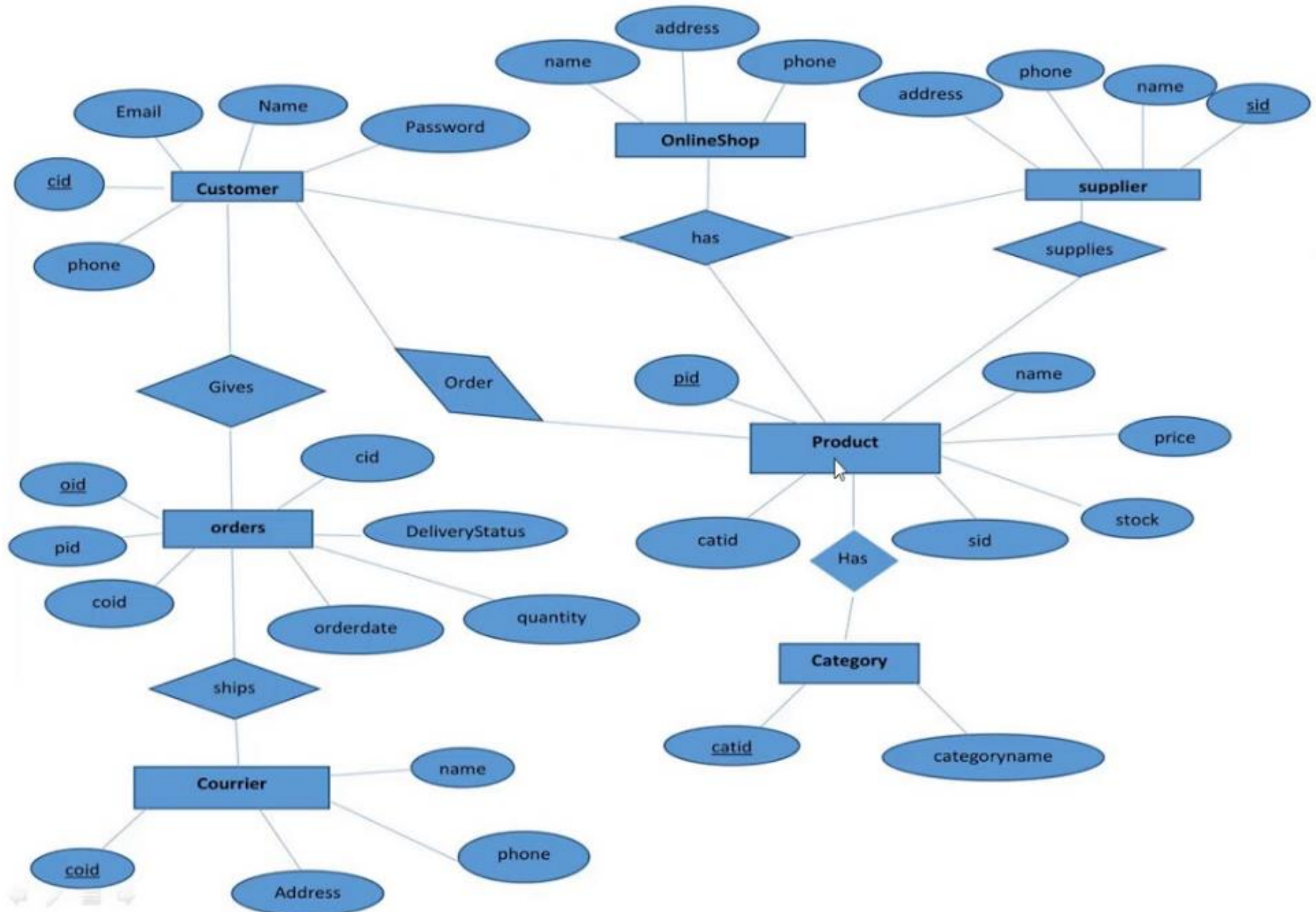
E-R Diagram of Library Management System



STUDENT MANAGEMENT SYSTEM



HOSPITAL MANAGEMENT SYSTEM



EXTENDED E-R FEATURES



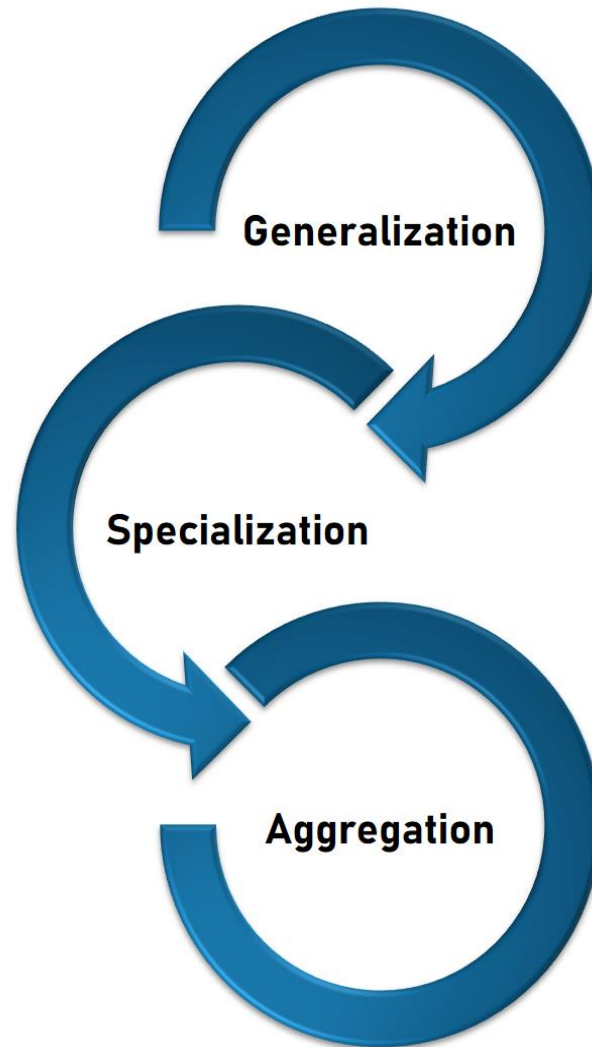
EXTENDED E-R MODEL

- ❖ ER model is supported with the additional semantic concepts is called as Extended / Enhanced Entity Relationship Model .
- ❖ Its also called EER.
- ❖ EER creates a design more accurate to database schemas.
- ❖ It reflects the data properties and constraints more precisely.
- ❖ It is used to represent a collection of objects that is union of objects of different of different entity types.



EXTENDED E-R MODEL CONCEPTS

- ❖ Three new concepts were added to the existing ER model:



SUPER CLASS AND SUBCLASS

❖ Super class

- ❖ An entity type that represents a general concept at a high level, is called superclass.

❖ Subclass

- ❖ An entity type that represents a specific concept at lower levels, is called subclass.
- ❖ The subclass is said to inherit from superclass.
- ❖ When a subclass inherits from one or more super classes, it inherits all their attributes

❖ The symbol used for specialization/ Generalization is:



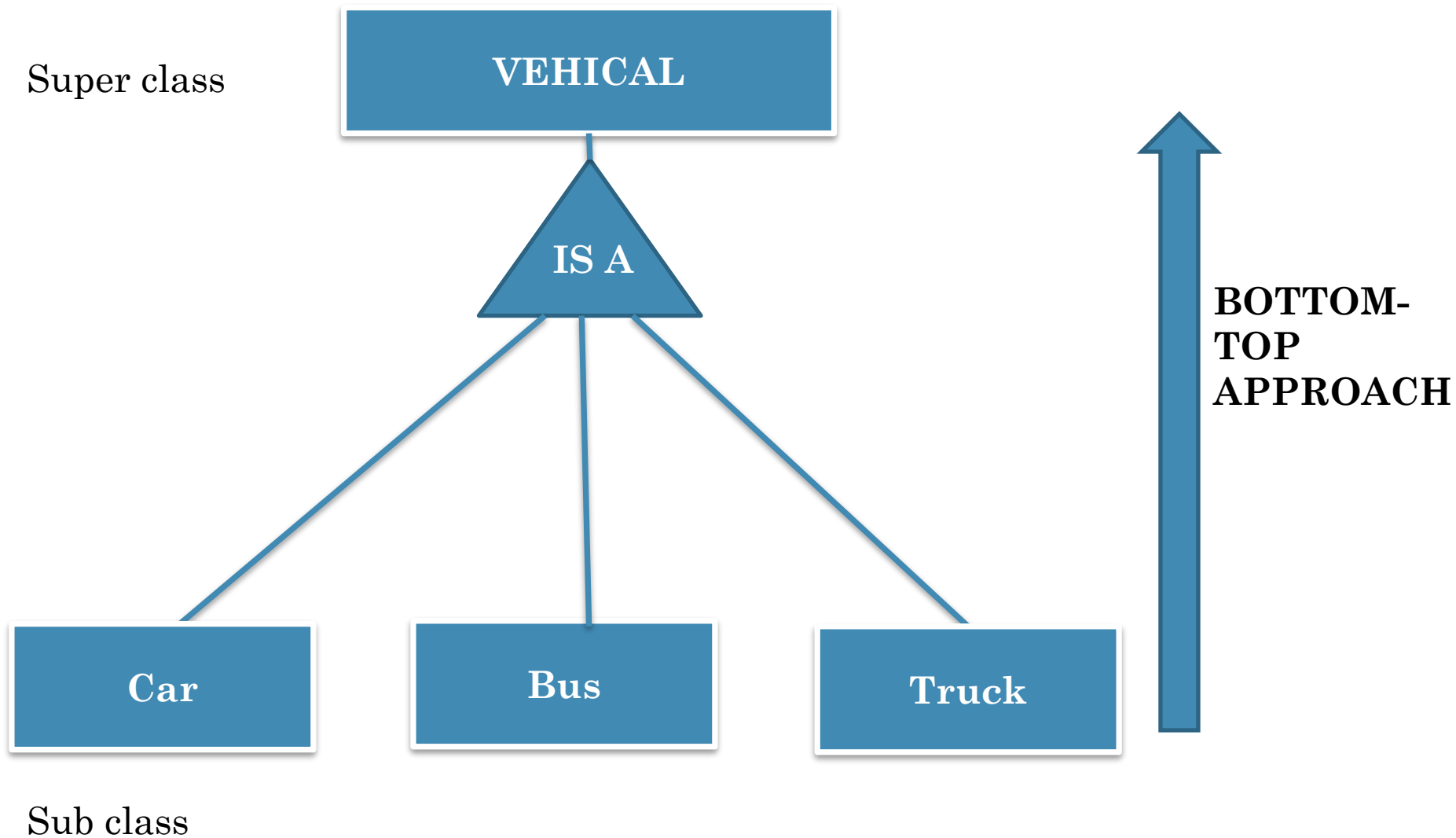
IS A

EXTENDED E-R FEATURES - GENERALIZATION

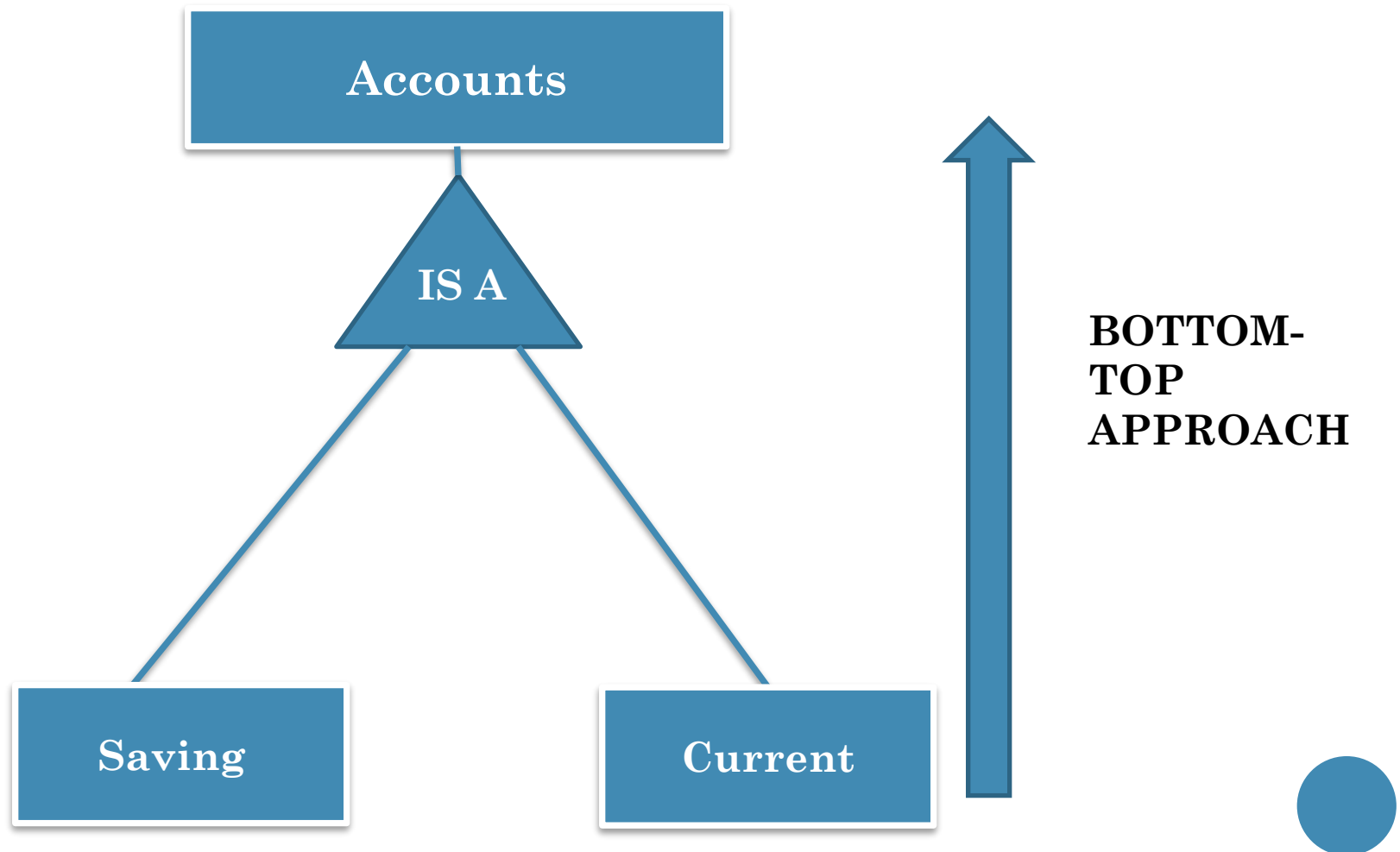
- ❖ It's a bottom-up approach
- ❖ Extract common properties from a set of entities and creates generalized entities from it.
- ❖ In generalization, two or more entities of lower level combine to form a higher level entity if they have some **attributes in common**.
- ❖ This approach creates a more generic entity, known as a superclass, by combining entities with similar features.
- ❖ sub-classes are combined to form a super-class.



EXTENDED E-R FEATURES - **GENERALIZATION**



EXTENDED E-R FEATURES - **GENERALIZATION**

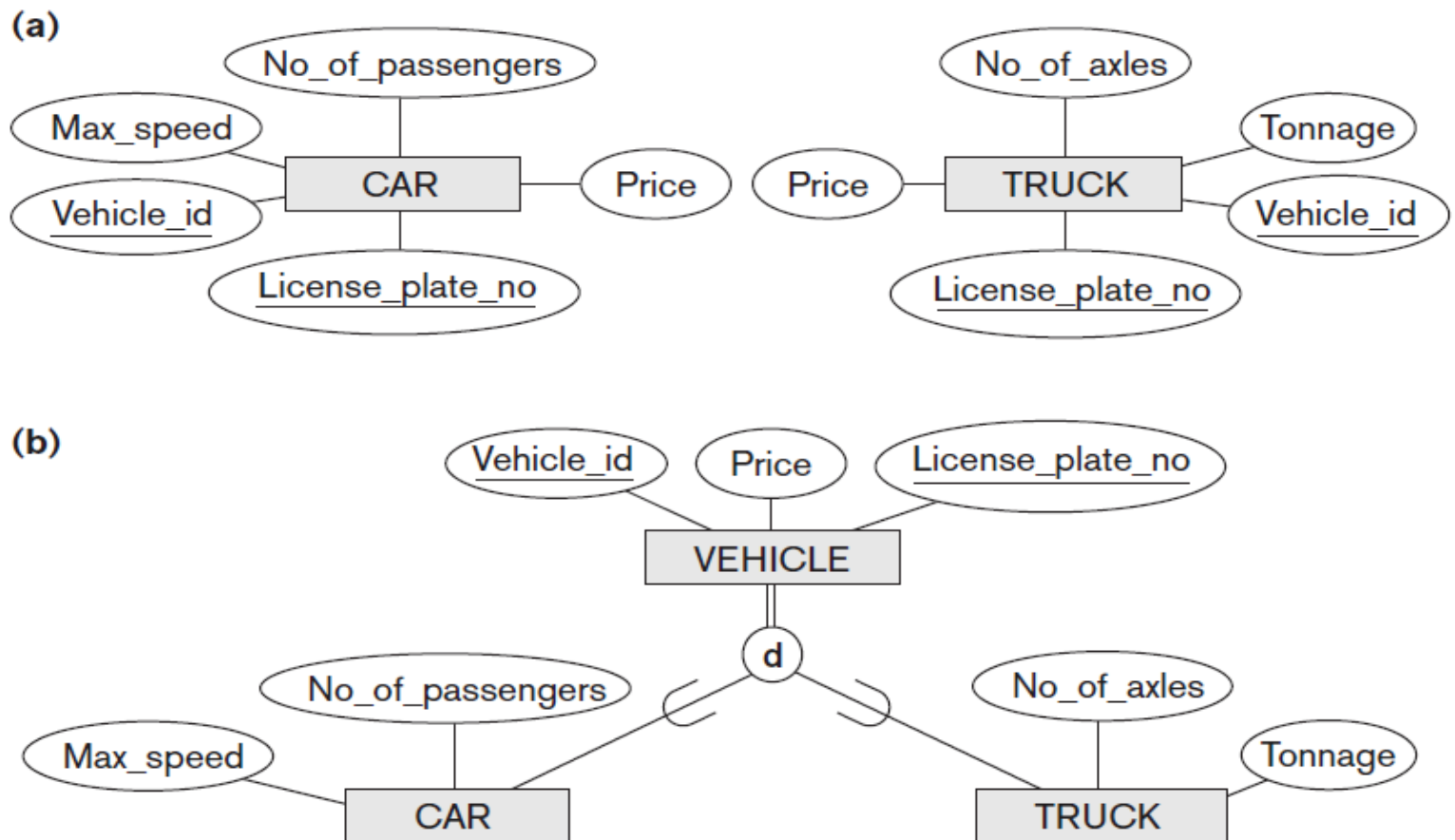


EXTENDED E-R FEATURES - GENERALIZATION


Figure 4.3

Generalization. (a) Two entity types, CAR and TRUCK.

(b) Generalizing CAR and TRUCK into the superclass VEHICLE.



EXTENDED E-R FEATURES - SPECIALIZATION

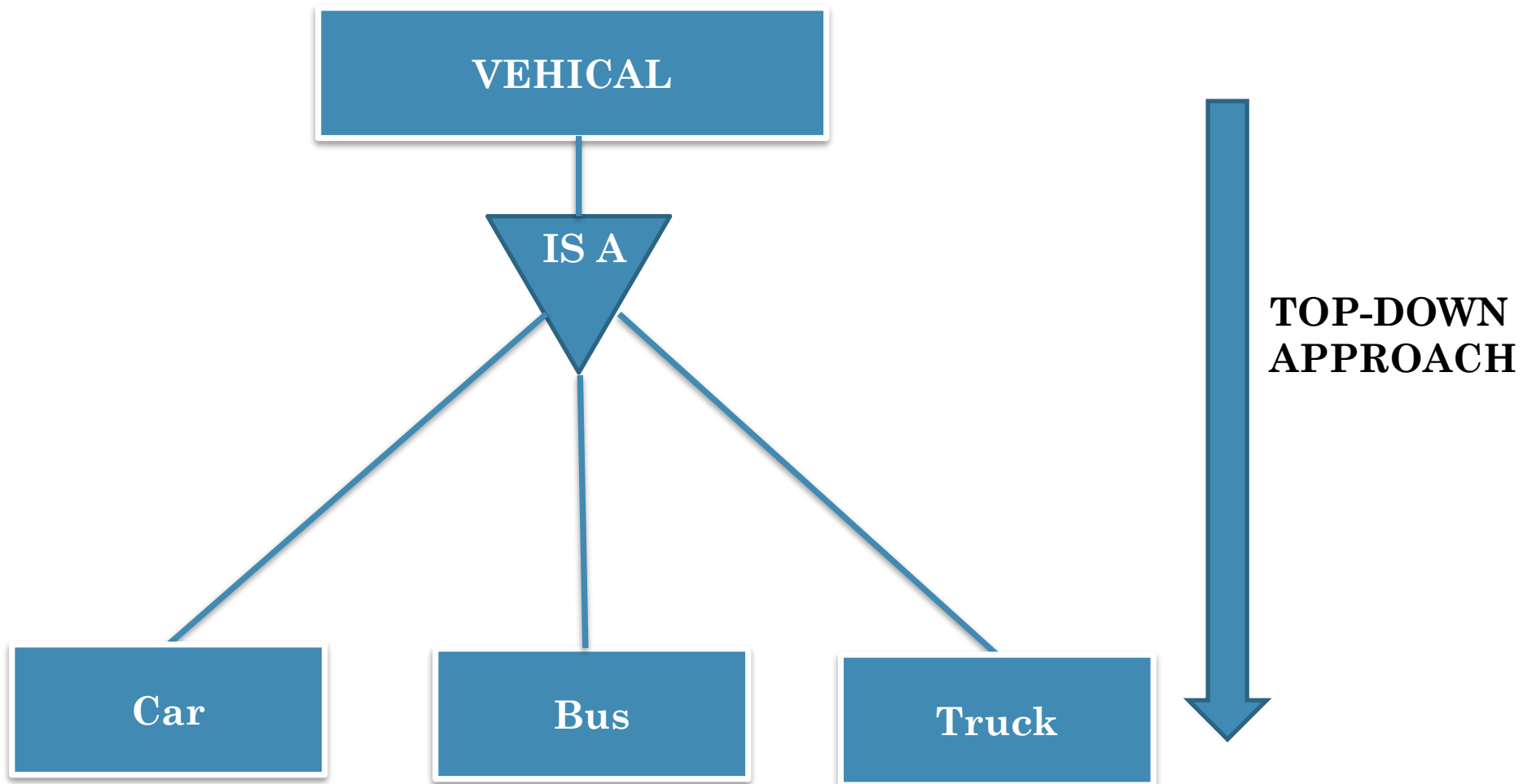
- ❖ Specialization is opposite of Generalization.
 - ❖ In Specialization, an entity is broken down into sub-entities based on their characteristics.
 - ❖ Specialization is a "Top-down approach" where higher level entity is specialized into two or more lower level entities.
 - ❖ Specialization is used to identify the subset of an entity set that shares some **distinguishing characteristics**.
 - ❖ Specialization can be repeatedly applied to refine the design of schema.
- 

EXTENDED E-R FEATURES - SPECIALIZATION

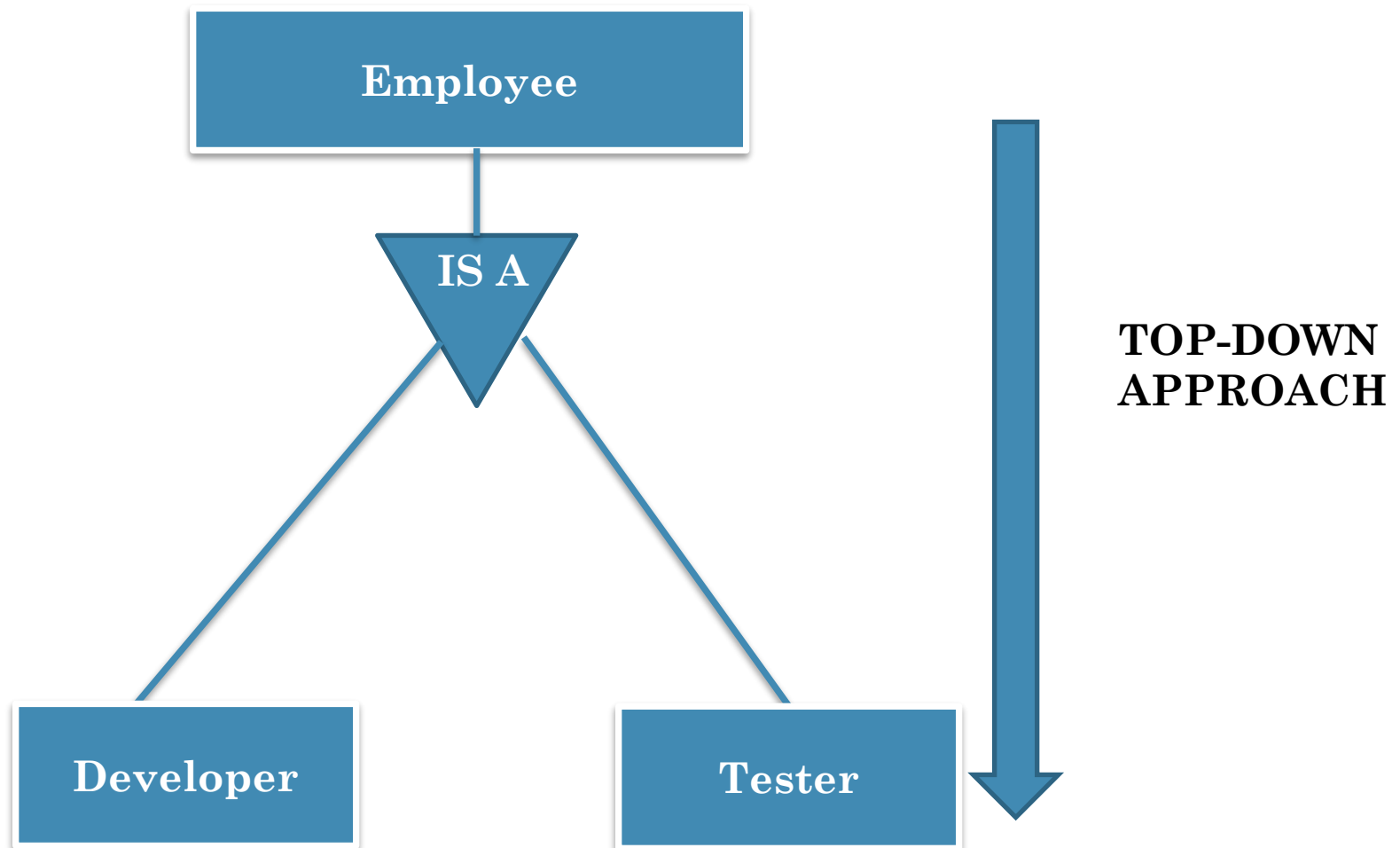
- ❖ For example, the set of subclasses {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of the superclass EMPLOYEE that distinguishes among employee entities based on the job type of each employee.



EXTENDED E-R FEATURES - SPECIALIZATION



EXTENDED E-R FEATURES - SPECIALIZATION



In the above example, Employee can be specialized as Developer or Tester, based on what role they play in an Organization

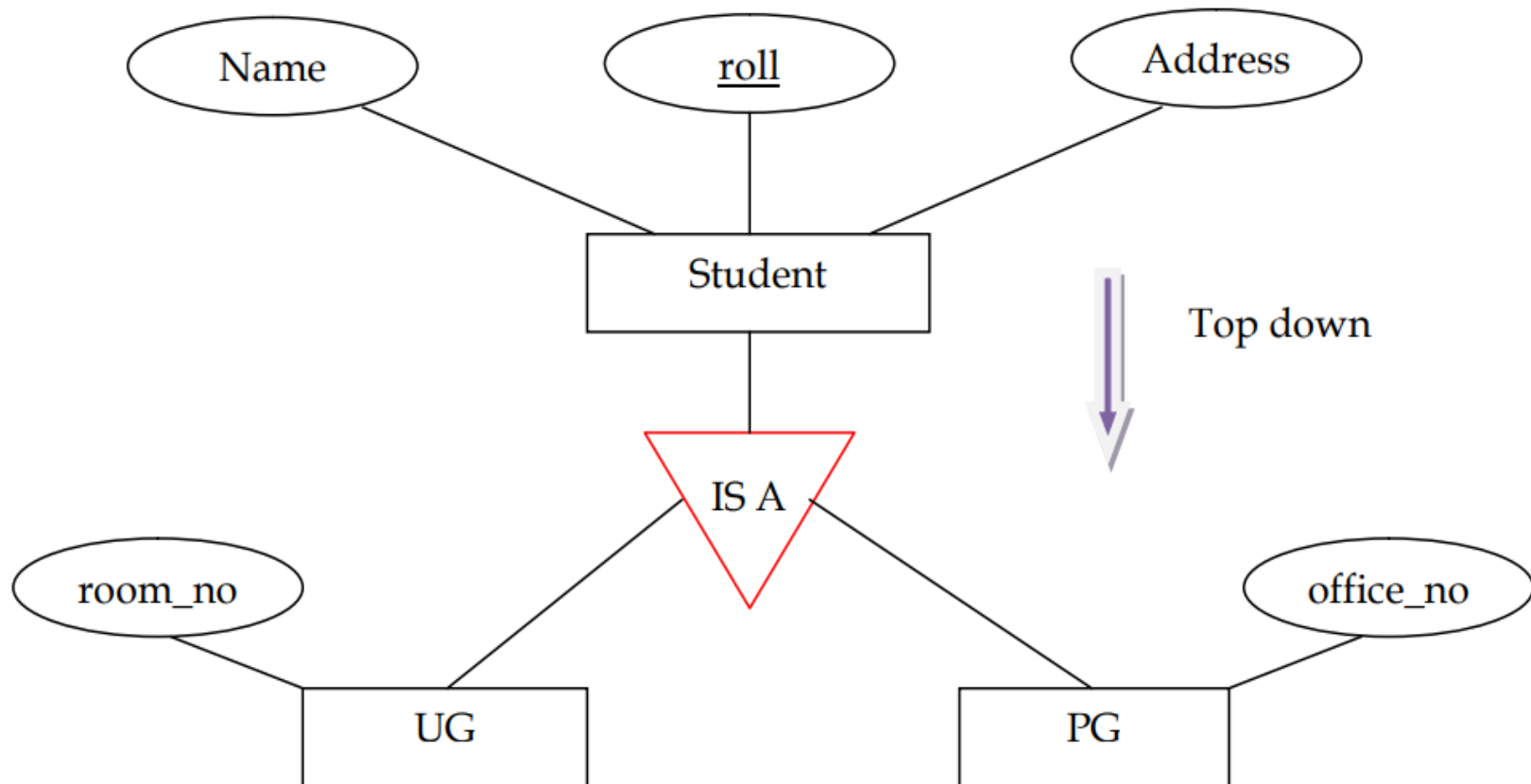
EXTENDED E-R FEATURES - SPECIALIZATION

- ❖ In university, student belongs to 2 categories “Undergraduate” & “Postgraduate”.
- ❖ Both categories of students may have common attributes as well as unique attributes.

UG Entity Attributes	PG entity Attributes
roll, Name, Address, room_no	roll, Name, Address, office_no



EXTENDED E-R FEATURES - SPECIALIZATION



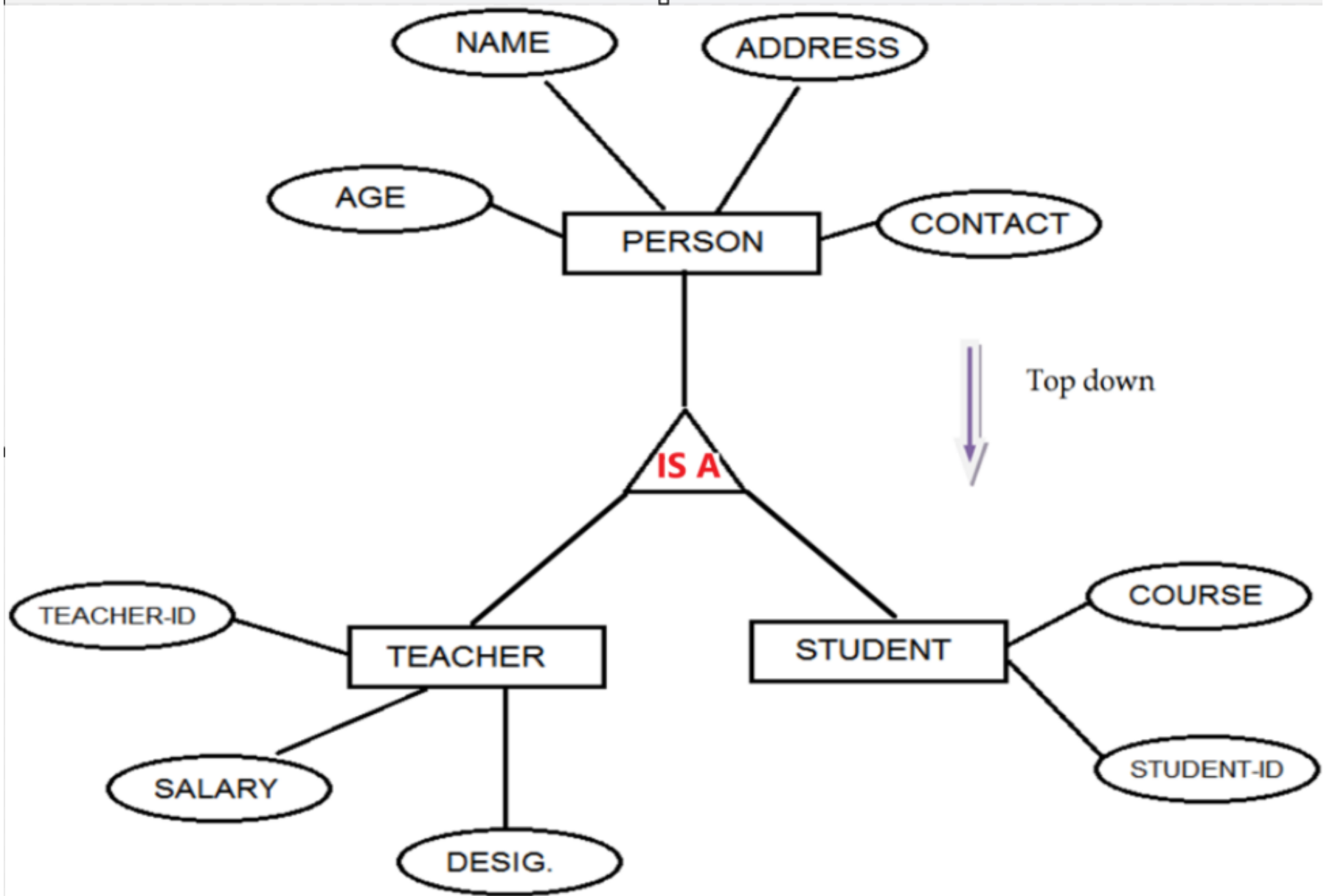
EXTENDED E-R FEATURES - SPECIALIZATION

- ❖ In university, student belongs to 2 categories “Undergraduate” & “Postgraduate”.
- ❖ Both categories of students may have common attributes as well as unique attributes.

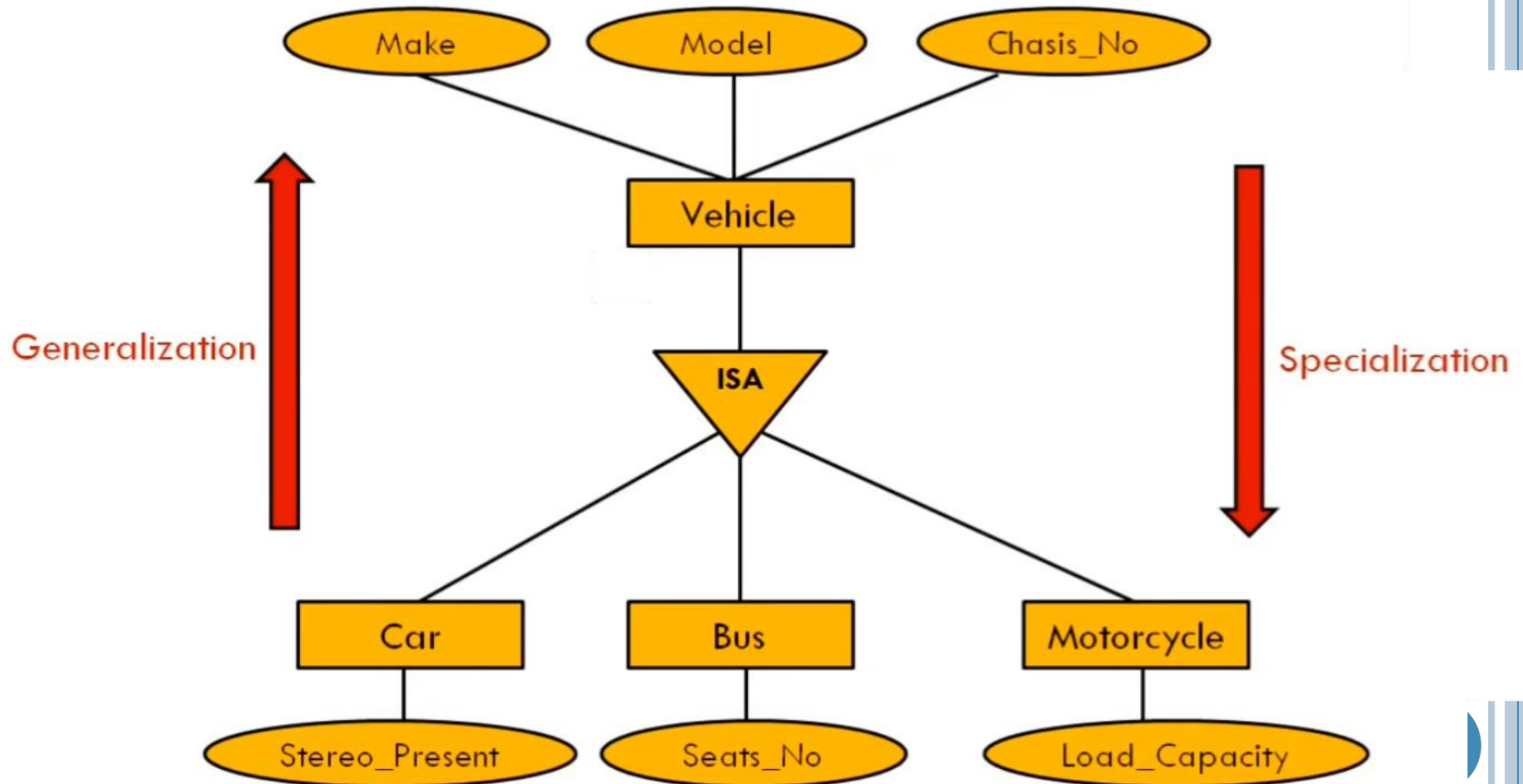
Teacher	Student
Teacher- id, Name, address, age, design, salary, contact no,	student- id, Name, address, age, course, contact no,



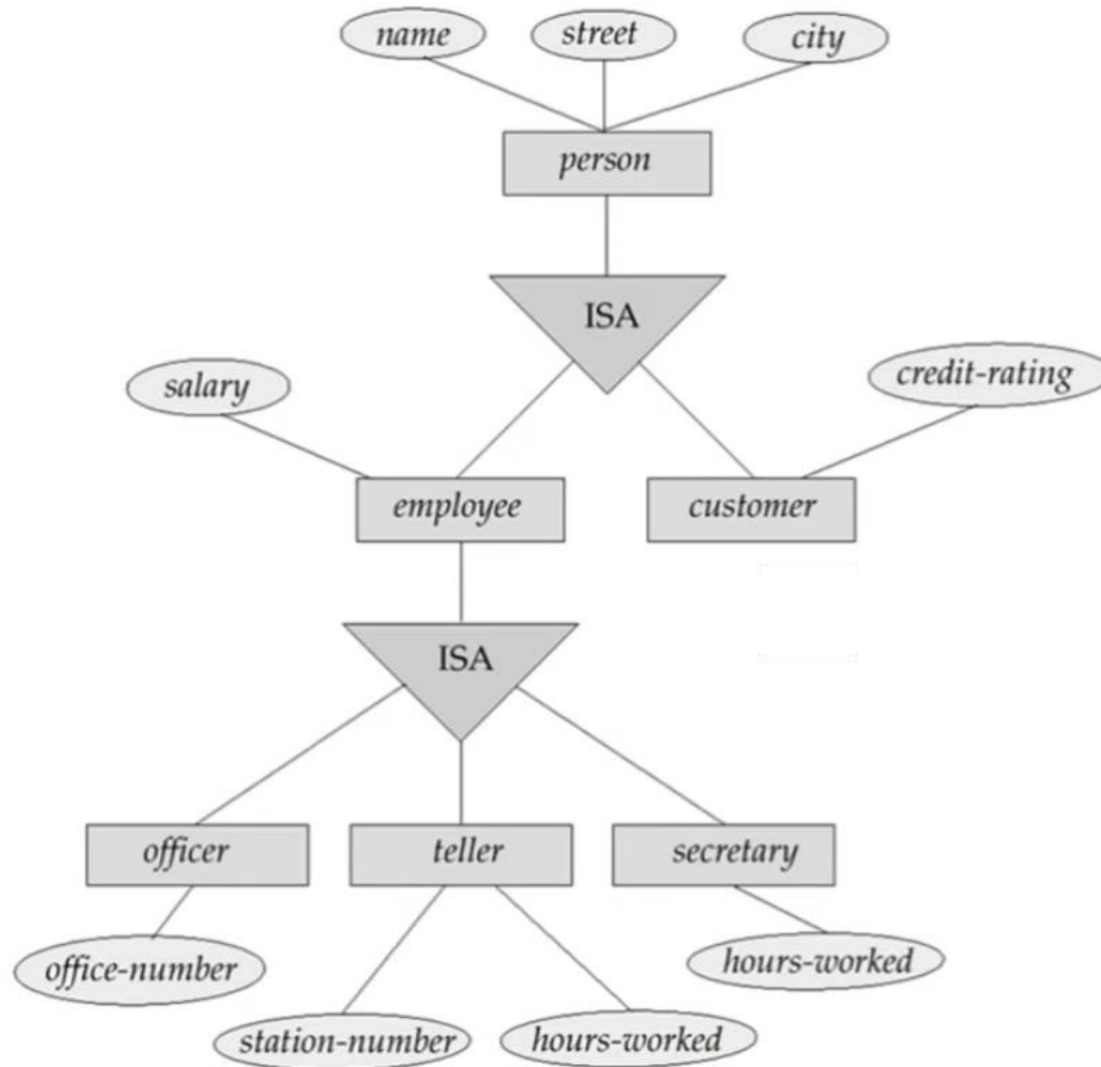
EXTENDED E-R FEATURES - SPECIALIZATION



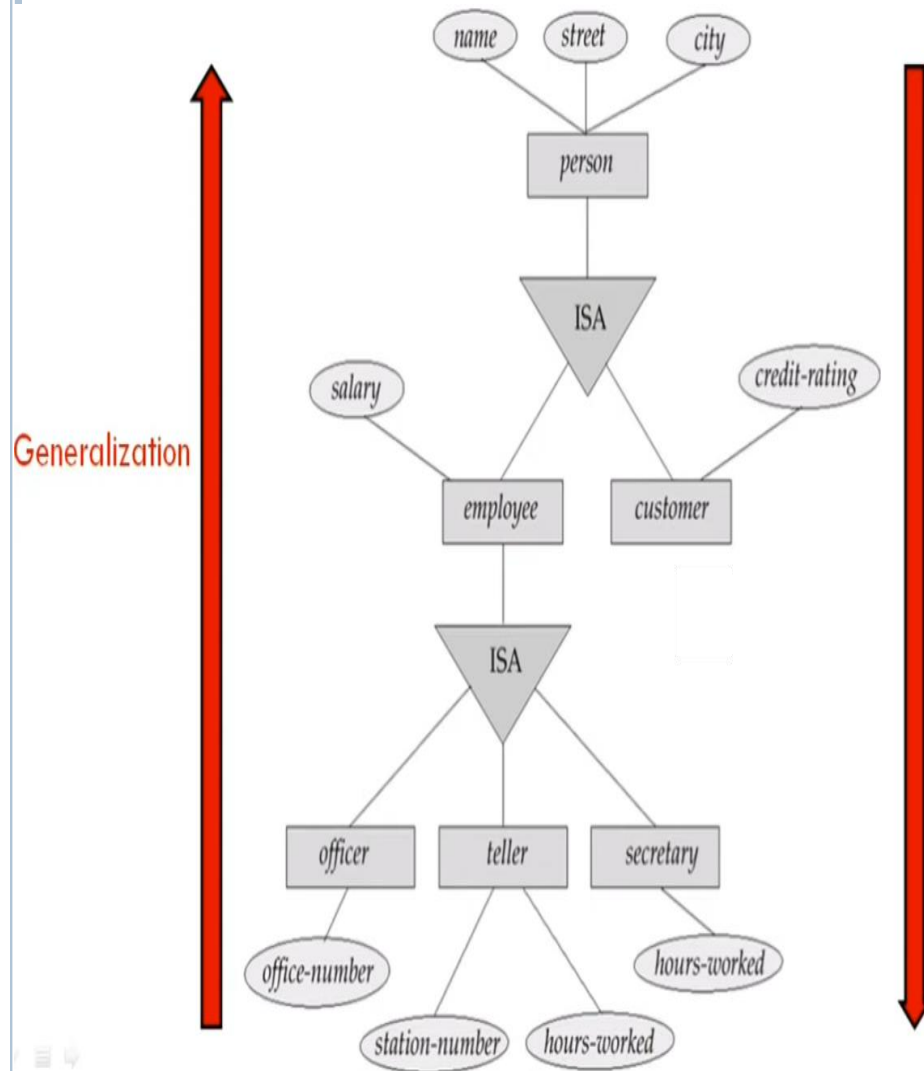
EXAMPLE :



EXAMPLE :



HOW TO DERIVED SCHEMA?



Customer:

(name, street, city, credit_Rating)

Officer:

(name, street, city, officer-number)

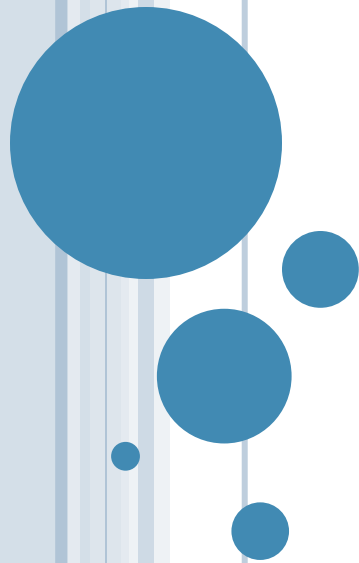
teller:

(name, street, city, station number, Hours-worked)

secretary:

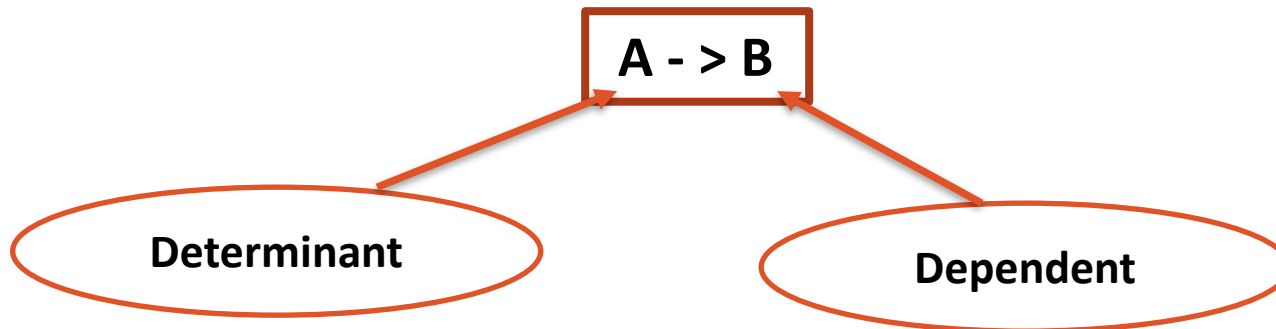
(name, street, city, hours-worked)

FUNCTIONAL DEPENDENCY



INTRODUCTION

- ❖ The functional dependency is a relationship that exists between two attributes.
- ❖ It helps in preventing data redundancy and gets to know about bad designs.
- ❖ Functional Dependency is represented by \rightarrow (arrow sign)
- ❖ let us consider P is a relation with attributes **A** and **B**.



B	\rightarrow	Functionally Dependent on A
A	\rightarrow	Determinant Set
B	\rightarrow	Dependent Attribute



EXAMPLE : TABLE DEPARTMENT

Dept_no is primary key

<u>Deptno</u>	Dept_name
011	Marketing
022	HR
033	Finance
044	Accounting
055	Sales
066	Telecom

Deptno - > Dept_name



EXAMPLE : TABLE DEPARTMENT

Employee_Id	Employee_Name	Department	Salary
1	Ryan	Mechanical	\$5000
2	Justin	Biotechnology	\$5000
3	Andrew	Computer Science	\$8000
4	Felix	Human Resource	\$10000

Employee_id is primary key

Employee_id - > {Employee_name, Department, Salary}



TYPES OF FUNCTIONAL DEPENDENCY

- ❖ Following are the types of functional dependency
 - ❖ Full Functional Dependency
 - ❖ Partial Functional Dependency
 - ❖ Transitive Functional Dependency
 - ❖ Trivial Functional Dependency
 - ❖ Nontrivial Functional Dependency



FULL FUNCTIONAL DEPENDENCY

- ❖ An attribute is fully functional dependent on another attribute, if it is Functionally Dependent on that attribute and not on any of its proper subset(attribute).

supplier_id	item_id	price
1	1	540
2	1	545
1	2	200
2	2	201
1	1	540
2	2	201

{supplier_id, item_id} → price



PARTIAL FUNCTIONAL DEPENDENCY

- ❖ Partial Dependency occurs when a nonprime attribute is functionally dependent on part of a candidate key.

Name	Roll_no	Course
Ravi	2	DBMS
Tim	3	OS
John	5	Java

Name and roll_no alone are able to uniquely identify a course



PARTIAL FUNCTIONAL DEPENDENCY

❖ Example : 2

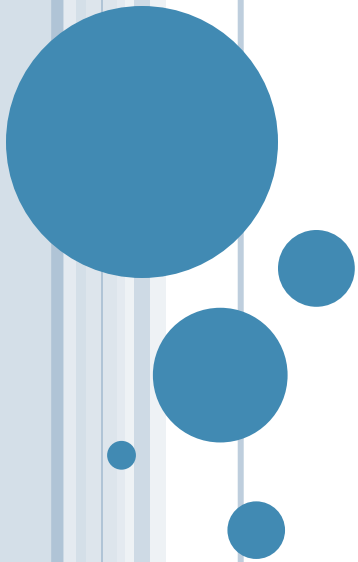
StudentID	ProjectNo	StudentName	ProjectName	StudentID
S01	199	Katie	Geo Location	S01
S02	120	Ollie	Cluster Exploration	S02

The **StudentName** can be determined by **StudentID** that makes the relation Partial Dependent.

The **ProjectName** can be determined by **ProjectID**, which that the relation Partial Dependent.



NORMALIZATION



WHAT IS NORMALIZATION?

- ❖ Normalization is the process of organizing the data in the database.
- ❖ In the normalization process your data of a single table will be distributed in multiple tables without losing any data.
- ❖ Normalization is used for mainly two purposes:
 - ❖ Eliminating redundant(useless) data.
 - ❖ Ensuring data dependencies make sense i.e data is logically stored.



ROW LEVEL DUPLICATION

<u>STUDID</u>	SNM	AGE
1	Raj	22
2	Meet	23
1	Raj	22

Row Level Duplication can be achieved with the help of defining Primary Key



COLUMN/FILED LEVEL DUPLICATION

<u>STU DID</u>	SNM	CID	CNM	FID	FNM	SALARY
1	Raj	C1	Java	F1	John	4000
2	Meet	C2	DBMS	F2	Smith	5000
3	Mahek	C1	Java	F1	John	4000
4	Nisha	C1	Java	F1	John	4000
5	Ruchit	C3	C++	F3	Scott	3000
....

Row level delicacy is not found but have some columns that have same values

COLUMN/FILED LEVEL DUPLICATION

<u>STU DID</u>	SNM	CID	CNM	FID	FNM	SALARY
1	Raj	C1	Java	F1	John	4000
2	Meet	C2	DBMS	F2	Smith	5000
3	Mahek	C1	Java	F1	John	4000
4	Nisha	C1	Java	F1	John	4000
5	Ruchit	C3	C++	F3	Scott	3000
....

❖ field level duplication always caused 3 major problems:

- ❖ Insertion Anomaly
- ❖ Updation Anomaly
- ❖ Deletion Anomaly



INSERT ANOMALY

❖ Consider a relation

❖ emp_dept (E#, Ename, Address, D#, Dname, Dmgr#) **E# as a primary key**

<u>E#</u>	Ename	Address	D#	Dname	Dmgr#
1	Raj	Rajkot	1	CE	1
2	Meet	Surat	1	CE	1

Want to insert new department detail (IT)

- Suppose a **new department (IT) has been started** by the organization but **initially there is no employee** appointed for that department.
- We **want to insert that department detail** in emp_dept table.
- But the **tuple for this department cannot be inserted** into this table as the **E# will have NULL value, which is not allowed** because **E# is primary key**.
- This kind of problem in the relation where some tuple cannot be inserted is known as insert anomaly.

WHAT IS INSERT ANOMALY?

- ❖ An insert anomaly occurs when **certain attributes cannot be inserted** into the database **without the presence of another attribute**.



DELETE ANOMALY

❖ Consider a relation

❖ emp_dept (E#, Ename, Address, D#, Dname, Dmgr#) **E# as a primary key**

<u>E#</u>	Ename	Address	D#	Dname	Dmgr#
1	Raj	Rajkot	1	CE	1
2	Meet	Surat	1	IT	2
3	EKTA	RAJKOT	1	CE	1
4	NIHAR	SURAT	1	CE	1

Want to delete Meet employee's detail

- Now consider **there is only one employee in some department (IT)** and that **employee leaves the organization**.
- So we **need to delete tuple of that employee (Meet)**.
- But in **addition to that information** about the **department also deleted**.
- This kind of problem in the relation where deletion of some tuples can lead to loss of some other data not intended to be removed is known as delete anomaly.

WHAT IS DELETE ANOMALY?

- ❖ A delete anomaly exists when **certain attributes are lost because of the deletion of another attribute.**



UPDATE ANOMALY

❖ Consider a relation

❖ emp_dept (E#, Ename, Address, D#, Dname, Dmgr#) **E# as a primary key**

<u>E#</u>	Ename	Address	D#	Dname	Dmgr#
1	Raj	Rajkot	1	CE	M1
2	Meet	Surat	2	IT	M2
3	Jay	Rajkot	2	C.E	M2

Want to update CE department's manager

- Suppose the **manager of a (CE) department has changed**, this requires that the **Dmgr# in all the tuples corresponding to that department must be changed** to reflect the new status.
- If we **fail to update all the tuples of given department**, then two **different records of employee working in the same department might show different Dmgr#** lead to inconsistency in the database.



WHAT IS UPDATE ANOMALY?

- ❖ An update anomaly exists when **one or more records (instance) of duplicated data is updated, but not all.**



HOW TO DEAL WITH INSERT ANOMALY

<u>EmpID</u>	EmpName	Address	DeptID	DeptName	DeptMngr
E1	Raj	Rajkot	D1	C.E.	Patel
E2	Samir	Rajkot	D2	Civil	Shah
E3	Meet	Baroda	D1	Computer	Patel
E4	Deepak	Surat	D1	C.E	Patel
E5	Suresh	Surat	D3	Electrical	Joshi
null	null	null	D4	Chemical	null

Do not allow to insert new department “Chemical” until an employee is assign to it.

<u>EmpID</u>	EmpName	Address	DeptID
E1	Raj	Rajkot	D1
E2	Samir	Rajkot	D2
E3	Meet	Baroda	D1
E4	Deepak	Surat	D1
E5	Suresh	Surat	D3

<u>DeptID</u>	DeptName	DeptMngr
D1	Computer	Patel
D2	Civil	Shah
D3	Electrical	Joshi
D4	Chemical	null

HOW TO DEAL WITH DELETE ANOMALY

<u>EmpID</u>	EmpName	Address	DeptID	DeptName	DeptMgr
E2	Samir	Rajkot	D2	Civil	Shah
E3	Meet	Baroda	D1	Computer	Patel
E4	Deepak	Surat	D1	C.E	Patel
E5	Suresh	Surat	D3	Electrical	Joshi

If we delete Employee having ID “E2” then Civil department will also delete because there is only one record of Civil dept.

<u>EmpID</u>	EmpName	Address	DeptID
E2	Samir	Rajkot	D2
E3	Meet	Baroda	D1
E4	Deepak	Surat	D1
E5	Suresh	Surat	D3

<u>DeptID</u>	DeptName	DeptMgr
D1	Computer	Patel
D2	Civil	Shah
D3	Electrical	Joshi

HOW TO DEAL WITH UPDATE ANOMALY

<u>EmpID</u>	EmpName	Address	DeptID	DeptName	DeptMngr
E1	Raj	Rajkot	D1	Computer	Patel
E2	Samir	Rajkot	D2	Civil	Shah
E3	Meet	Baroda	D1	Computer	Patel
E4	Deepak	Surat	D1	Computer	Patel
E5	Suresh	Surat	D3	Electrical	Joshi

Changing the name of department D1 from “Computer” to “IT” may update one or more records, but not all.

<u>EmpID</u>	EmpName	Address	DeptID
E1	Raj	Rajkot	D1
E2	Samir	Rajkot	D2
E3	Meet	Baroda	D1
E4	Deepak	Surat	D1
E5	Suresh	Surat	D3

<u>DeptID</u>	DeptName	DeptMngr
D1	Computer	Patel
D2	Civil	Shah
D3	Electrical	Joshi

HOW MANY NORMAL FORMS ARE THERE?

- ❖ Data redundancy can be removed with the help of normal forms.
- ❖ Normal forms provide method where we can remove data redundancy by using Normal forms.
- ❖ Normal forms are :
 1. 1NF (First normal form)
 2. 2NF (Second normal form)
 3. 3NF (Third normal form)

As we **move from 1NF to 3NF** number of tables and **complexity increases** but **redundancy decreases**.



1NF (FIRST NORMAL FORM)

❖ Conditions for 1NF

Each cells of a table should contain a single value.

- ❖ A relation R is in first normal form (1NF) if and only if it **does not contain** any **composite attribute** or **multi-valued attributes** or their combinations.

OR

- ❖ A relation R is in first normal form (1NF) if and only if all underlying **domains contain atomic values / single values** only.



1NF (FIRST NORMAL FORM) [COMPOSITE ATTRIBUTE]

Customer

<u>Customer ID</u>	Name	Address
C01	Raj	Jamnagar Road, Rajkot
C02	Meet	Nehru Road, Jamnagar

- ❖ In above relation **address** is **composite attribute** which is further divided into sub-attributes as “Road” and “City”.
- ❖ So above relation is not in 1NF.



1NF (FIRST NORMAL FORM) [COMPOSITE ATTRIBUTE]

Customer

<u>CustomerID</u>	Name	Address
C01	Raj	Jamnagar Road, Rajkot
C02	Meet	Nehru Road, Jamnagar

❖ Problem:

- ❖ It is **difficult to retrieve the list of customers living in 'Jamnagar'** from above table.
- ❖ The reason is that **address attribute is composite attribute** which **contains road name as well as city name** in single cell.
- ❖ It is **possible that city name word is also there in road name**.
- ❖ In our example, **'Jamnagar'** word **occurs in both records**, in **first record it is a part of road name** and in **second one it is the name of city**.



1NF (FIRST NORMAL FORM) [COMPOSITE ATTRIBUTE]

Customer

<u>CustomerID</u>	Name	Address
C01	Raj	Jamnagar Road, Rajkot
C02	Meet	Nehru Road, Jamnagar

- ❖ **Solution:** Divide composite attributes into number of sub-attributes and insert value in proper sub-attribute.

Customer

<u>Customer ID</u>	<u>Name</u>	<u>Road</u>	<u>City</u>
C01	Raj	Jamnagar Road	Rajkot
C02	Meet	Nehru Road	Jamnagar



1NF (FIRST NORMAL FORM) [MULTIVALUED ATTRIBUTE]

Student

<u>RollNo</u>	Name	FailedinSubjects
101	Raj	DS, DBMS
102	Meet	DBMS, DS
103	Jeet	DS, DBMS, DE
104	Harsh	DBMS
105	Nayan	DE, DS

Student

Roll No		Subject
101	Raj	DS
101	Raj	DBMS
102	Meet	DBMS
102	Meet	DS
103	Jeet	DS
103	Jeet	DBMS
104	Harsh	DBMS
105	Nayan	DE
105	Nayan	DS

1NF

2NF (SECOND NORMAL FORM)

- ❖ A relation R is in second normal form (2NF)
 - ❖ If and only if it is in **1NF** and
 - ❖ **Every non-primary key attribute is fully dependent on the primary key**



2NF (SECOND NORMAL FORM)

TEACHER_ID	SUBJECT	TEACHER_AGE
111	JAVA	38
111	C++	38
222	PHP	36
333	DBMS	40
333	PHP	40

- ❖ This table is in 1NF
- ❖ It is not in 2NF because non prime attribute Teacher_Age is dependent on Teacher_id
- ❖ This violates the 2NF rule : **No non-prime attribute is dependent on the proper subset of any candidate key of table.**

2NF (SECOND NORMAL FORM)

Teacher details

TEACHER_ID	TEACHER_AGE
111	38
222	36
333	40

Teacher Subject

TEACHER_ID	SUBJECT
111	JAVA
111	C++
222	PHP
333	DBMS
333	PHP



3NF (THIRD NORMAL FORM)

- ❖ A relation R is in third normal form (3NF)
 - ❖ If And Only If It Is In **2NF** And
 - ❖ **Every Non-key Attribute Is Non-transitively Dependent On The Primary Key**



3NF (SECOND NORMAL FORM)

student

<u>RollNo</u>	State	City
101	punjab	Maholi
102	gujarat	baroda
103	gujarat	Surat
104	punjab	Maholi
105	Nayan	Baroda
106	Bihar	Maholi

Primary key: rollno

Determines → state

→ determine → city

State is non prime so its called
transitive dependency



3RD NORMAL FORM

HOUSING (SID, Building, Fee)

Key: SID

Functional

dependencies: Building \rightarrow Fee

SID \rightarrow Building \rightarrow Fee

SID	Building	Fee
100	Randolph	1200
150	Ingersoll	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200

(a)

STU-HOUSING (SID, Building)

Key: SID

SID	Building
100	Randolph
150	Ingersoll
200	Randolph
250	Pitkin
300	Randolph

(b)

BLDG-FEE (Building, Fee)

Key: Building

Building	Fee
Randolph	1200
Ingersoll	1100
Pitkin	1100