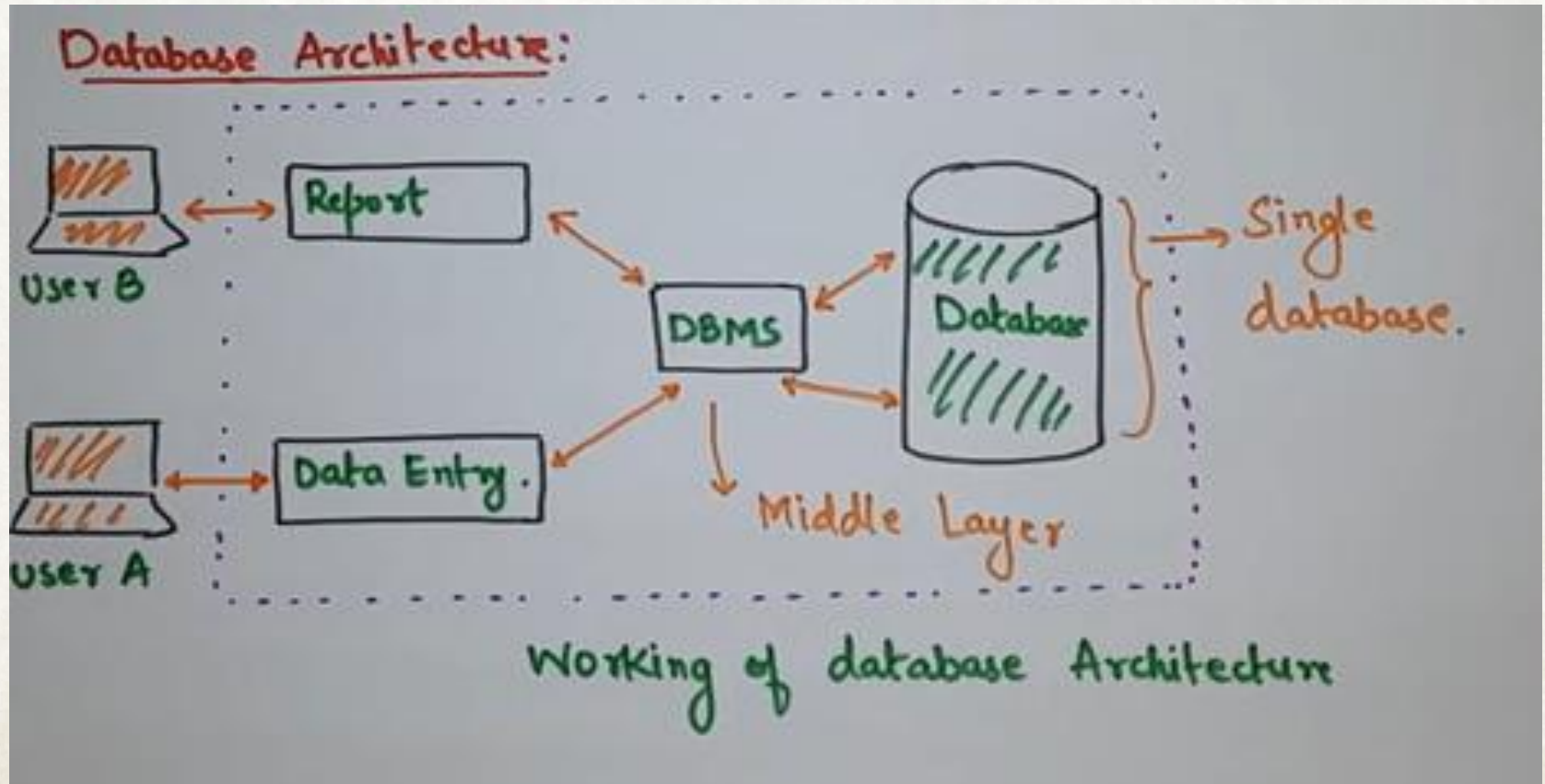


# DATA MODELS

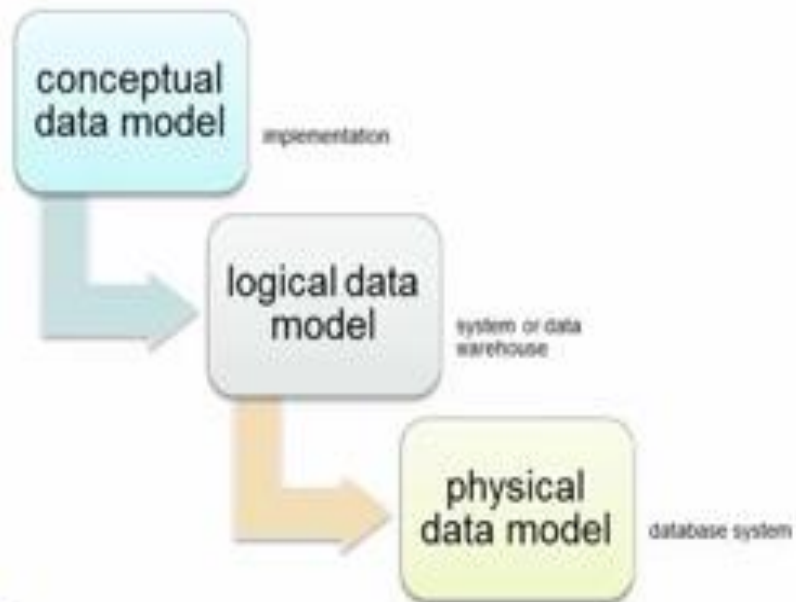


# Database Architecture

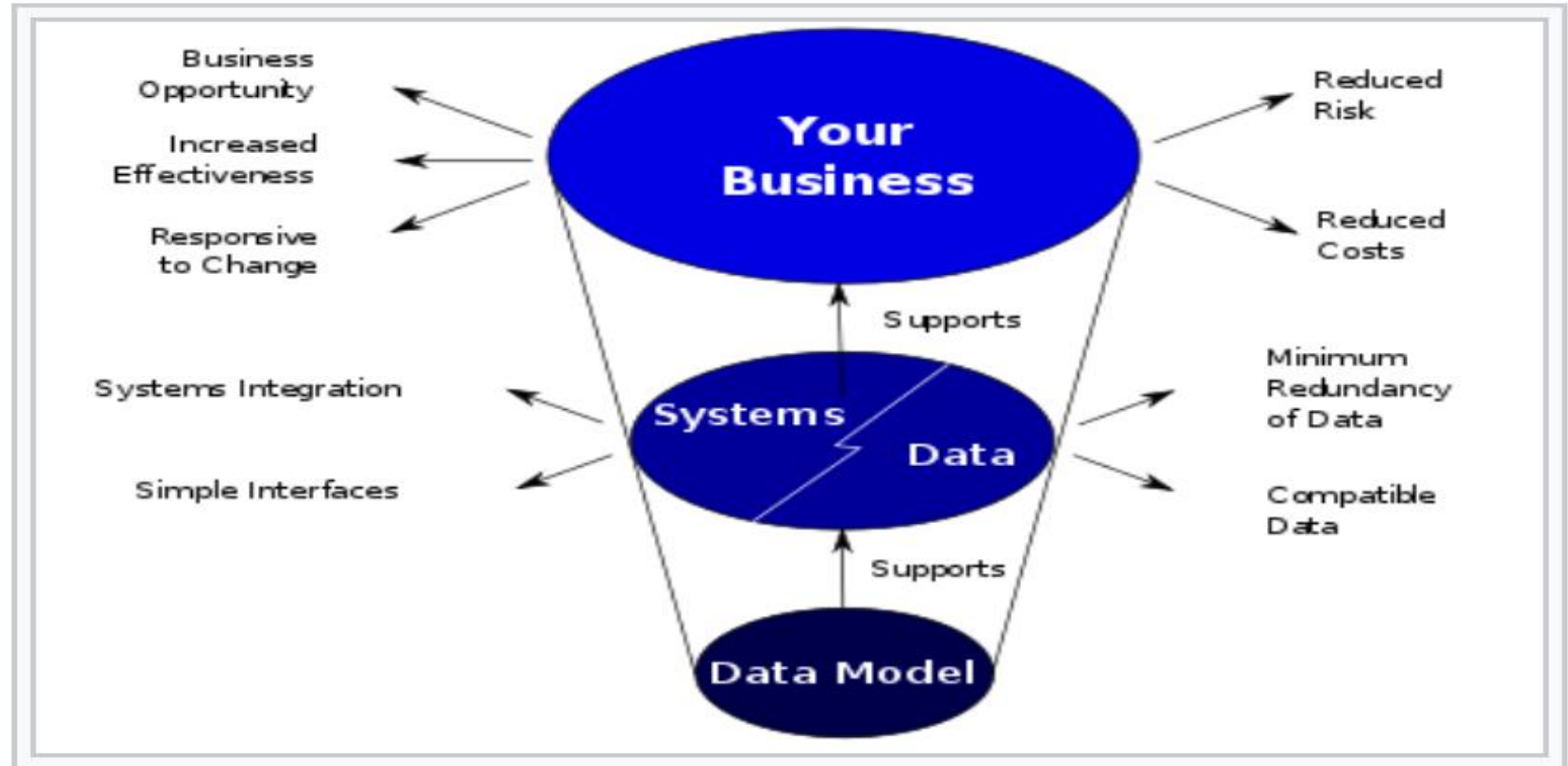


## What is Data Model?

A collection of conceptual tools for describing data, data relationships, data semantics and constraints.



# Role of Data Model



# Evolution of Data Model

TABLE 2.1 Evolution of Major Data Models

GENERATION	TIME	MODEL	EXAMPLES	COMMENTS
First	1960s–1970s	File system	VMS/VSAM	Used mainly on IBM mainframe systems Managed records, not relationships
Second	1970s	Hierarchical and network	IMS ADABAS IDS-II	Early database systems Navigational access
Third	Mid-1970s to present	Relational	DB2 Oracle MS SQL-Server	Conceptual simplicity Entity relationship (ER) modeling support for relational data modeling
Fourth	Mid-1980s to present	Object-oriented  Extended Relational	Versant VFS/FastObjects Objectivity/DB DB/2 UDB Oracle 10g	Support complex data Extended relational products support objects and data warehousing Web databases become common
Next Generation	Present to future	XML	dbXML Tamino DB2 UDB Oracle 10g MS SQL Server	Organization and management of unstructured data Relational and object models add support for XML documents

# Types of Model

1) Relational Data model: In this Data is organized into tables (rows and columns).

Table → Relation

Rows = Tuples.

Columns = Attributes.

↳ represents relationship among a set of values.

Example:-

class ] → Relation, columns/ Attributes

No.	Head
1	A
2	B
3	C
4	D

→ Rows/  
→ tuples

Student

sid	Sname	class
1	Aman	1
2	Melon	2

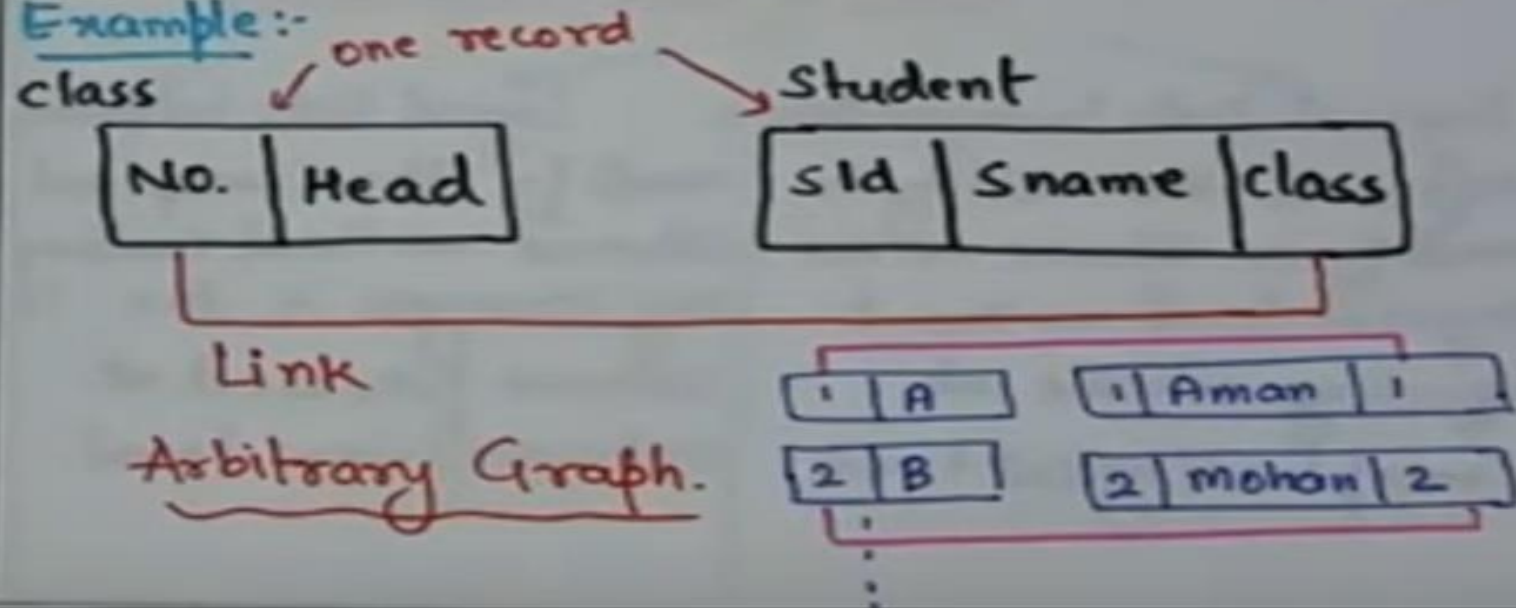
✓ Relationship  
blw two tables,  
By Pri-Foreign Key Concept.

(2) Network Data Model: In this data is represented by collection of records and relationships among data are represented by links.

Record → It is a coll<sup>n</sup> of attributes, each of which contains only one data value.

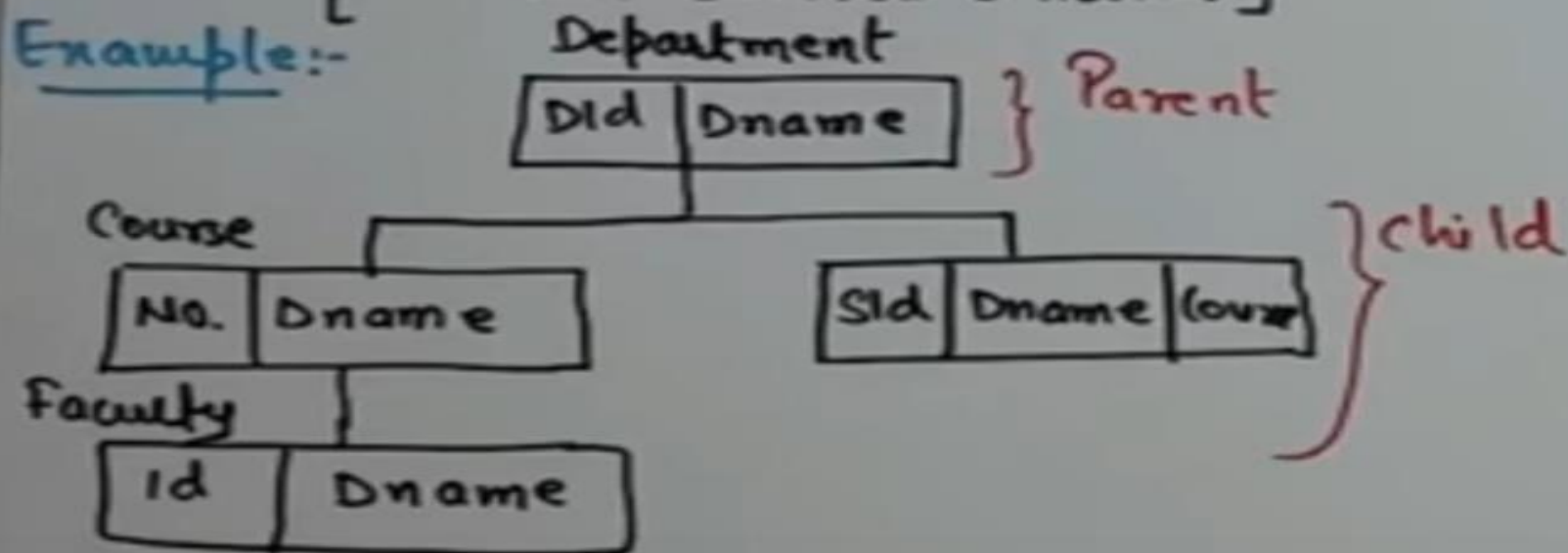
Link → It is an association b/w two records.

Example:-



(3) Hierarchical Data Model:- Similar to that of Network Data Model. The Only difference is that in hierarchical model, records are organized as trees rather than arbitrary graphs. [ Each entity has only one parent but can have several children ]

Example:-







# E-R model

- First introduced ER model in 1976, which was graphical representation of entities and their relationship in a database structure.
- Represented by entity - relationship diagram.
- E-R model is based on real world perception that comprises a collection of objects or entities and the relationships among these objects.

# Components of E-R Model

## ENTITY

- Is a real-world object distinguishable or unique from other objects.
- An entity can be a **concrete** or **physical** object like employee, student, faculty, customer etc. Or it could also be **conceptual** or **abstract** like transaction, order, course, subjects etc.
- It can be thought of as a **noun** like student, employee etc.
- It is normally represented by a rectangle shape.

**Strong Entity** –An entity that has a primary key is called as Strong entity. Rectangle represents strong entity.



Entity

**Weak Entity** –Weak entity doesn't have sufficient attributes to form a primary key of its own. Double rectangle represents weak entity.



Weak Entity

Represented by  
**Rectangle**



**Example**

STUDENT





## ENTITY SET

- It is a collection of similar type of entities. An entity set may contain entities with attribute sharing similar values.

**Example:** “Staff” set may contain all the staffs of a company from all employees.

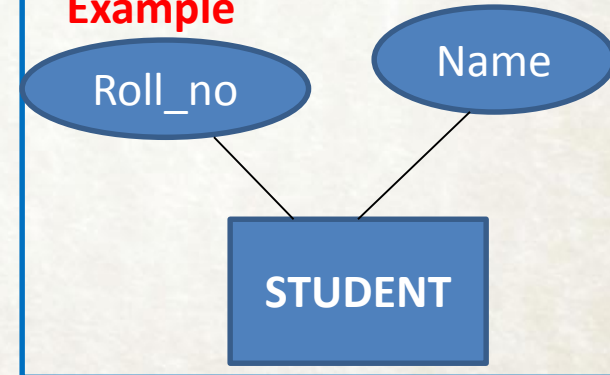
# ATTRIBUTE

- It is the **fact** about, or **properties** , and entity which are descriptive and is possessed by each member of an entity set.
- Attribute is an element of a data type like string, integer, date.
- These values are printable representation.

Represented by  
**Ellipse**

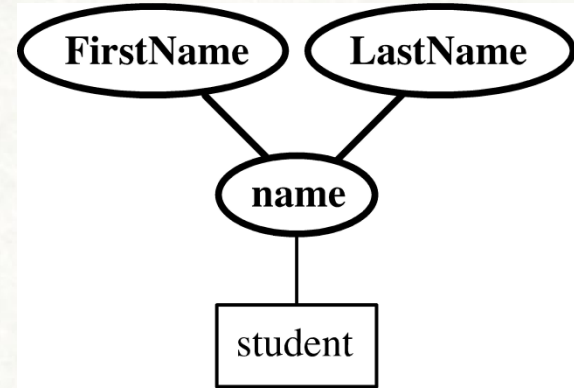


**Example**

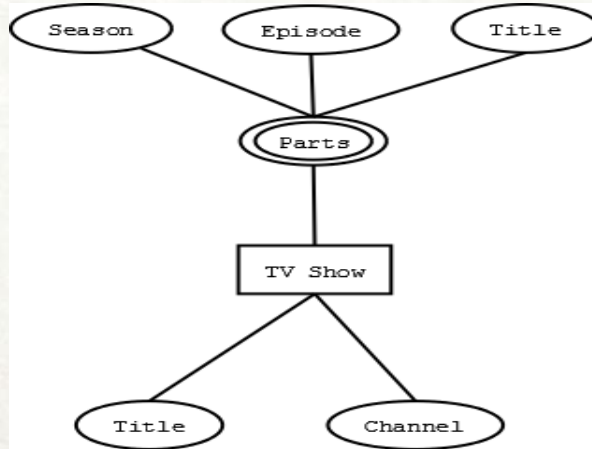


# Characteristics of Attribute

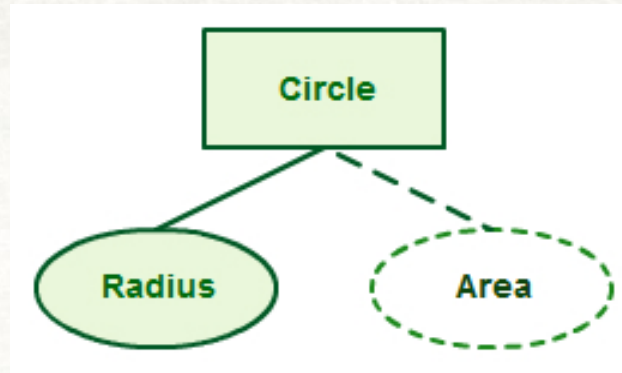
1. Simple and composite attribute



1. Single valued and multivalued attribute

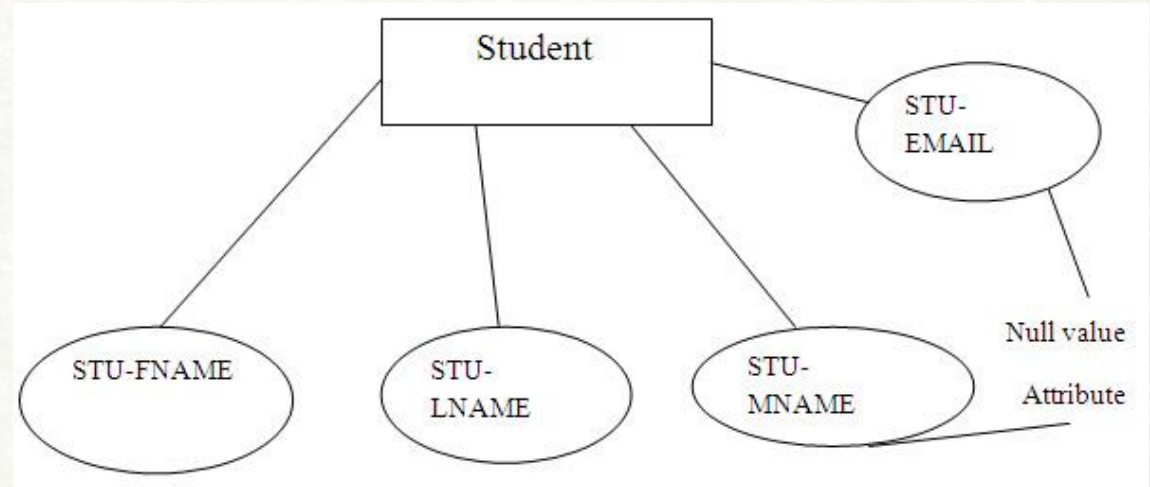


3. Derived attribute

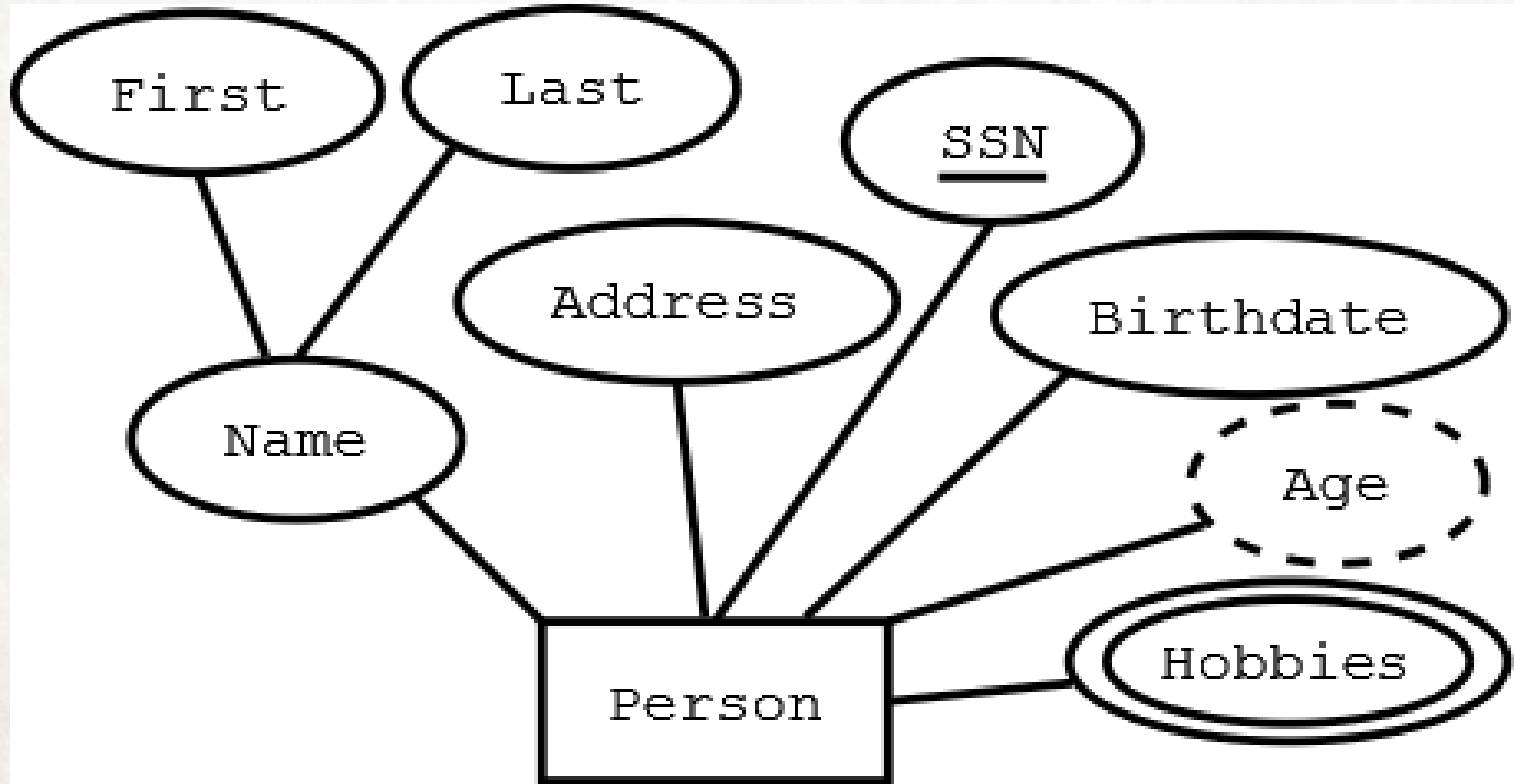


4. Null attribute

5. Key attribute

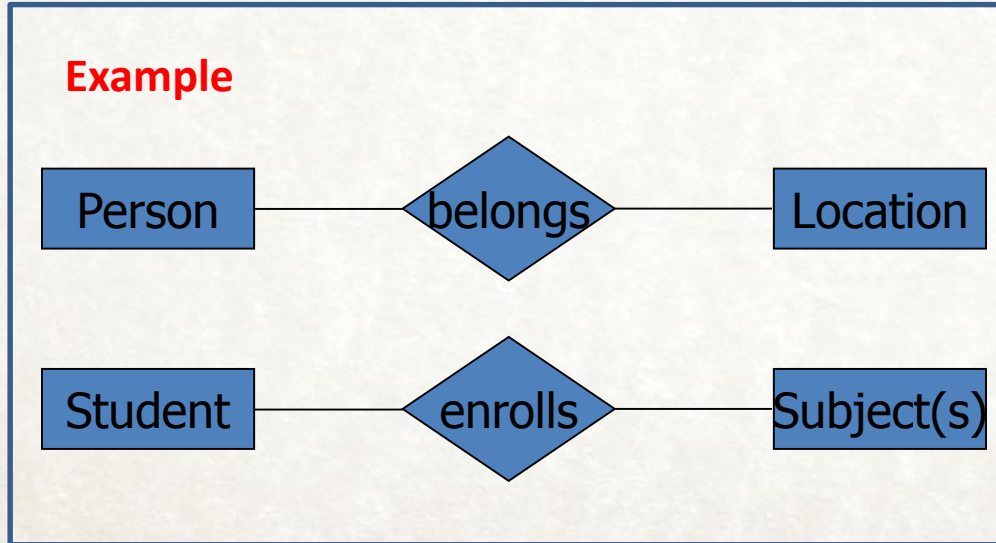


# Example

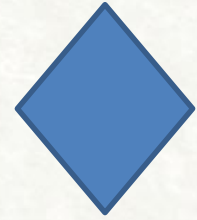


# RELATIONSHIP

- It depicts the link between entities.



Represented by  
**Diamond**





# Types of Relationship



Zero or one



Many



One



One (and only one)



Zero or many



One or many

**One-to-One**

Student

1

has

1

ID

**One-to-Many**

Division

1

has

M

Program

**Many-to-Many**

Student

M

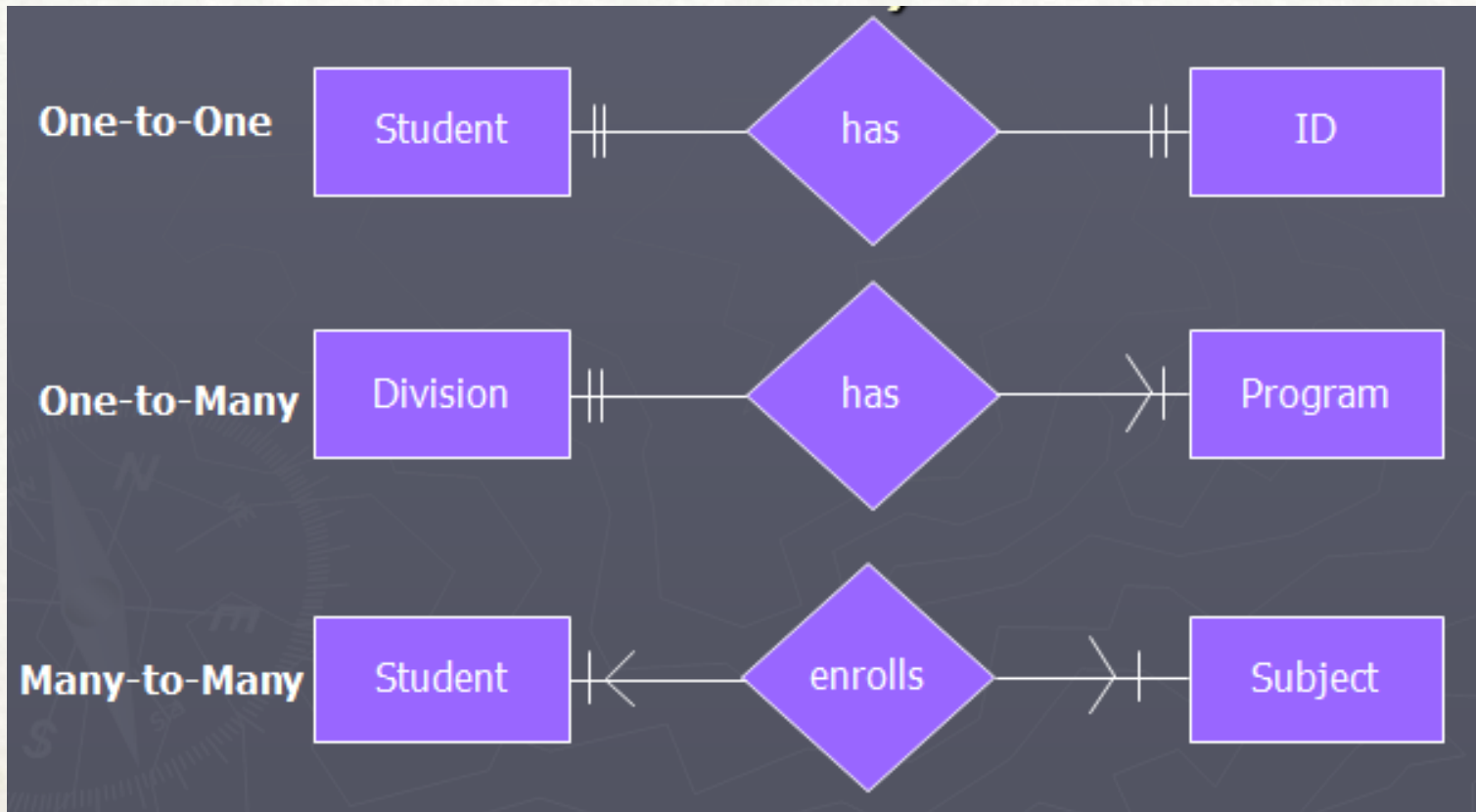
enrolls

M

Subject



# Relationship (crow notation)



- one-to-one

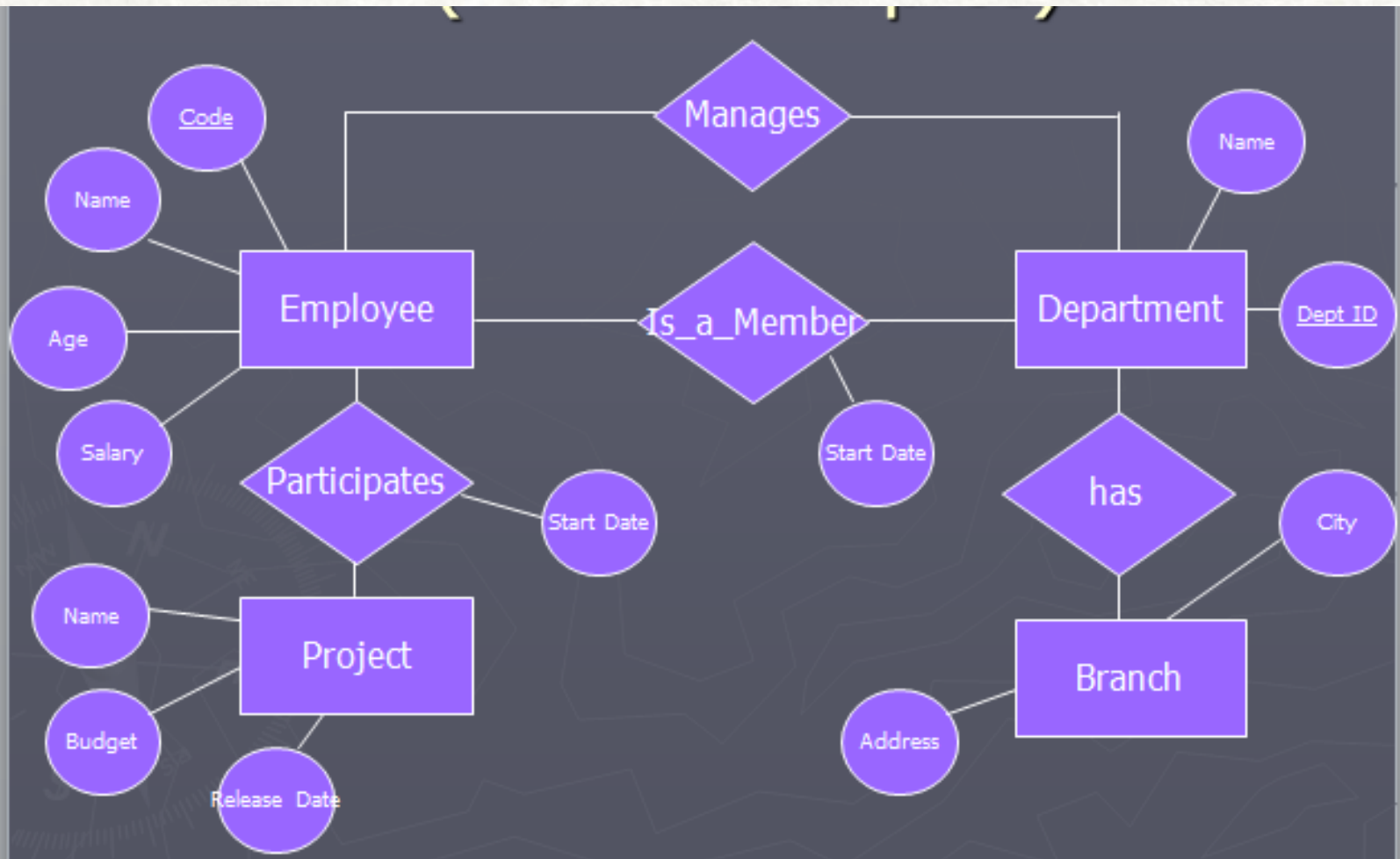


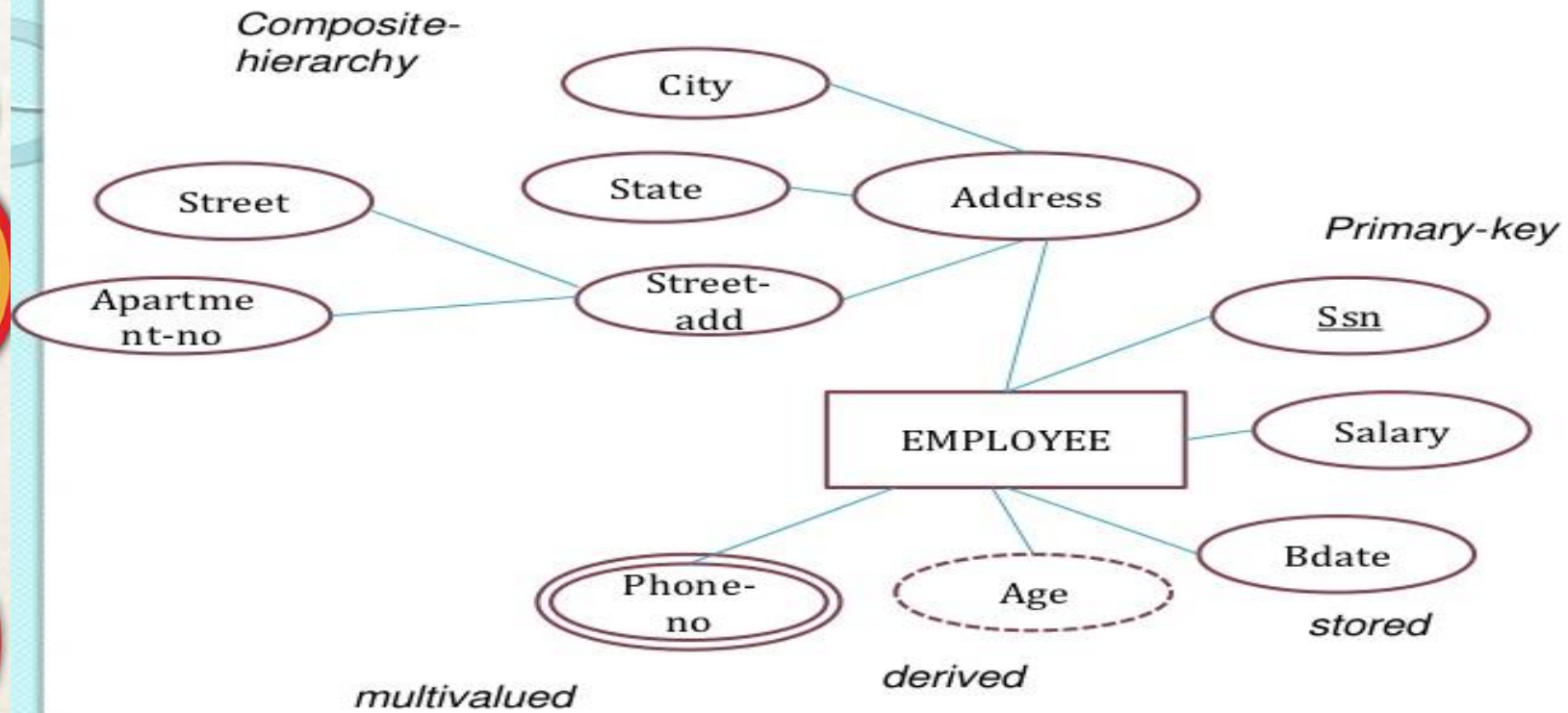
- one to many



- many-to-many









# Example

A university consists of a number of departments. Each department offers several courses. A number of modules make up each course. Students enrol in a particular course and take modules towards the completion of that course. Each module is taught by a lecturer from the appropriate department, and each lecturer tutors a group of students



# Example - Entities

A university consists of a number of **departments**. Each department offers several **courses**. A number of **modules** make up each course. **Students** enrol in a particular course and take modules towards the completion of that course. Each module is taught by a **lecturer** from the appropriate department, and each lecturer tutors a group of students



# Example - Relationships

- A university consists of a number of departments. Each department **offers** several courses. A number of modules **make up** each course. Students **enrol in** a particular course and **take** modules towards the completion of that course. Each module is **taught by** a lecturer **from the** appropriate department, and each lecturer **tutors** a group of students

# Example - E/R Diagram

Entities: Department, Course, Module, Lecturer, Student

Department

Course

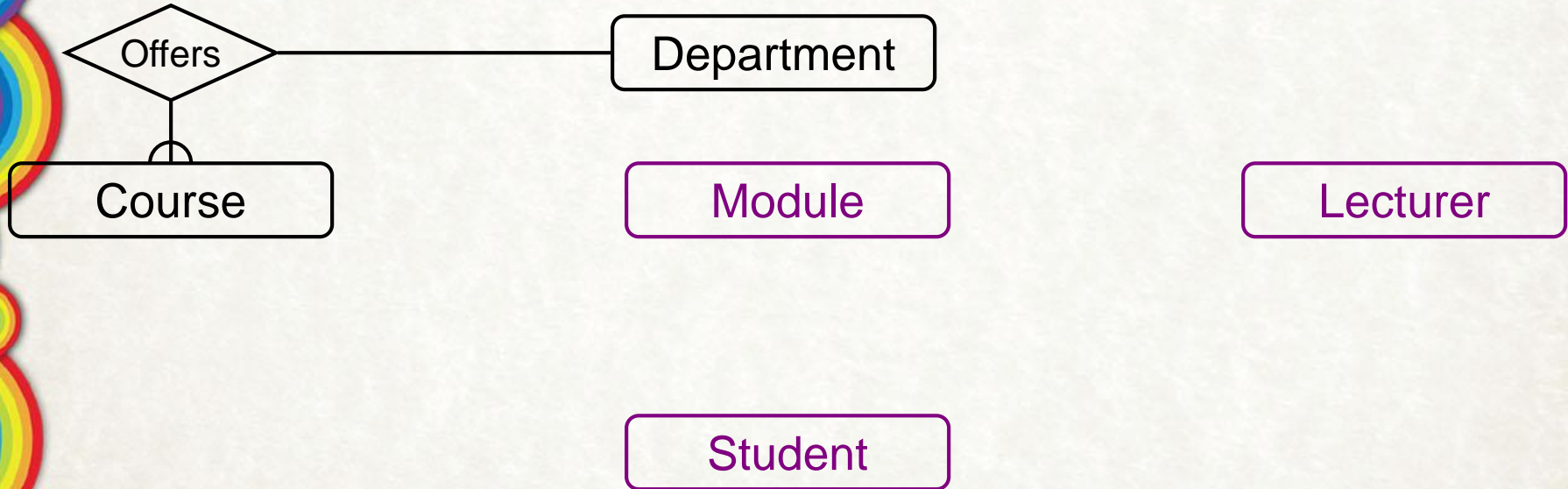
Lecturer

Module

Student

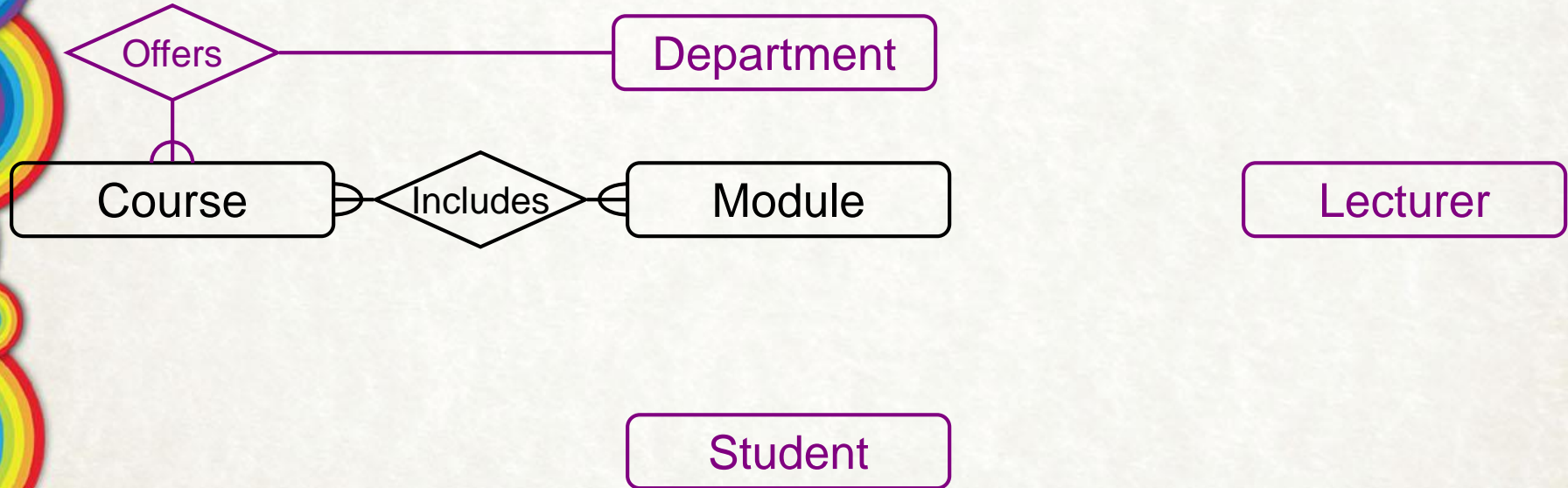
# Example - E/R Diagram

Each department offers several courses



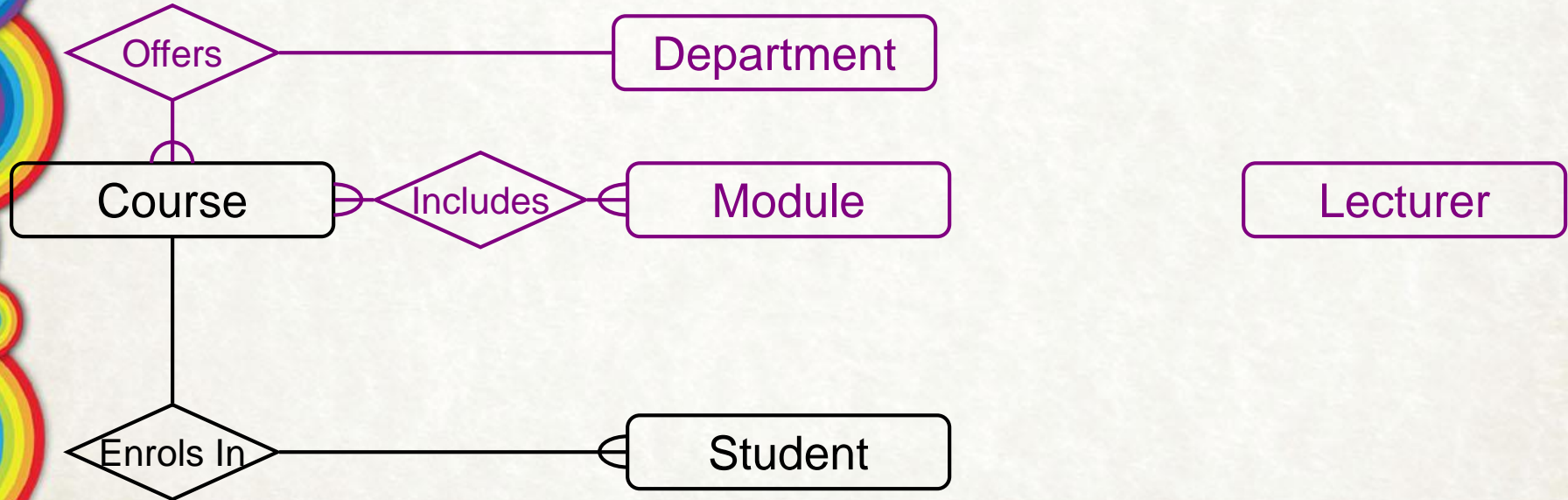
# Example - E/R Diagram

A number of modules **make up** each courses

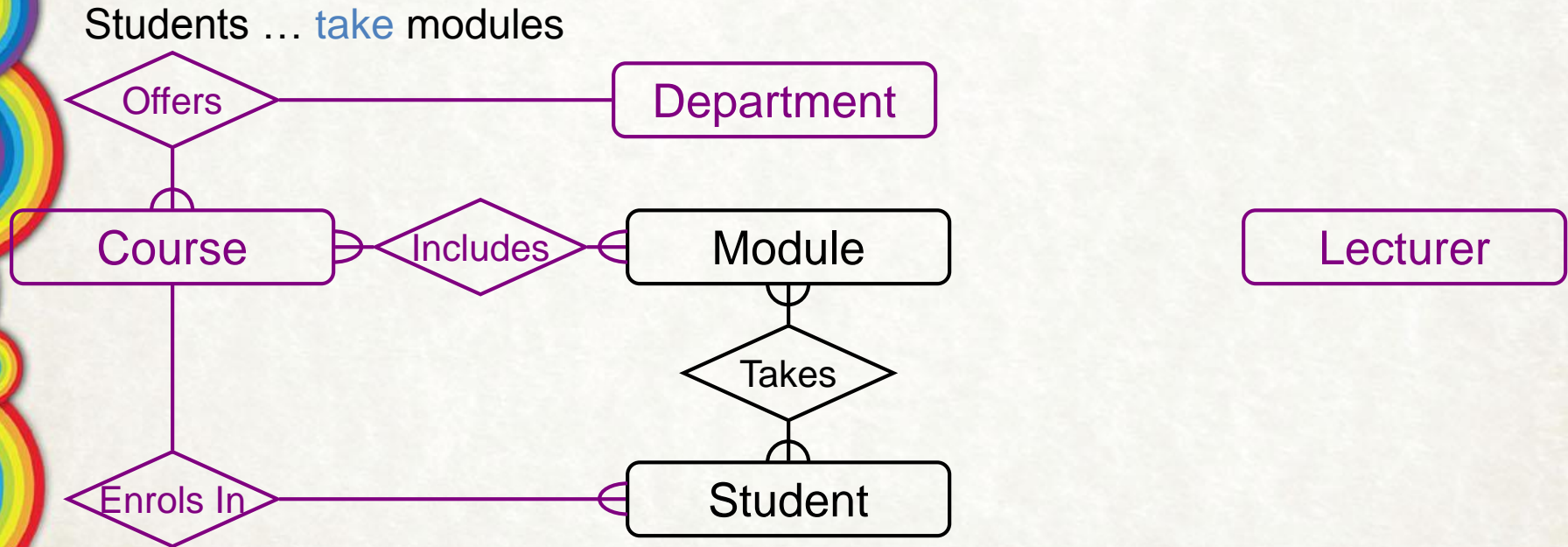


# Example - E/R Diagram

Students enrol in a particular course

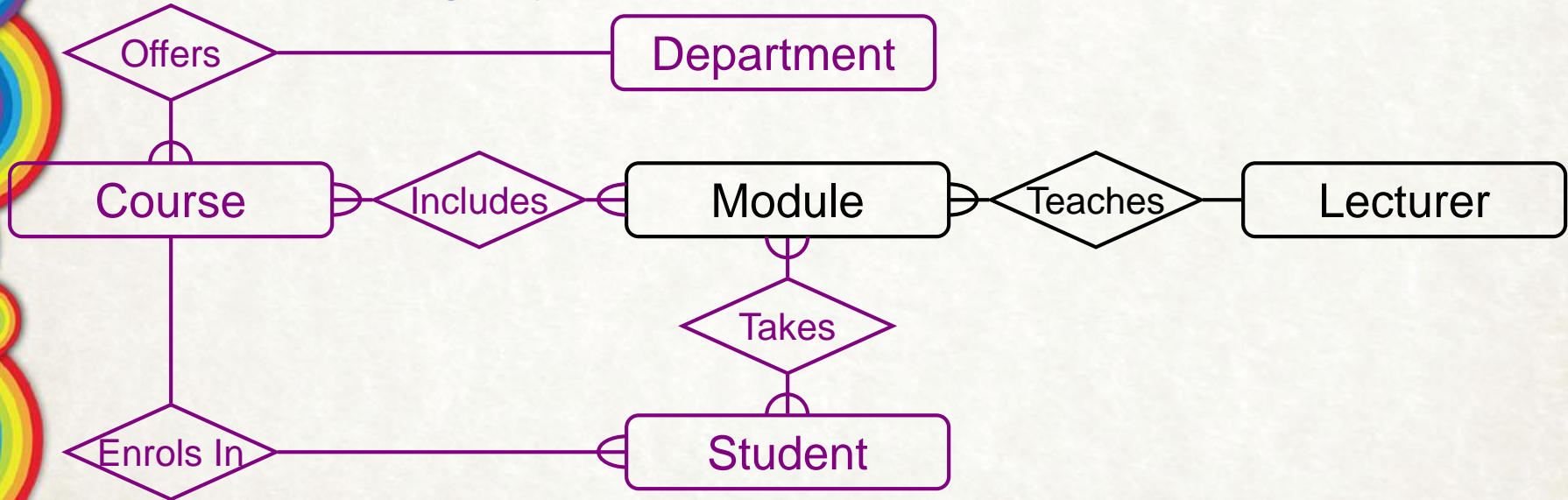


# Example - E/R Diagram

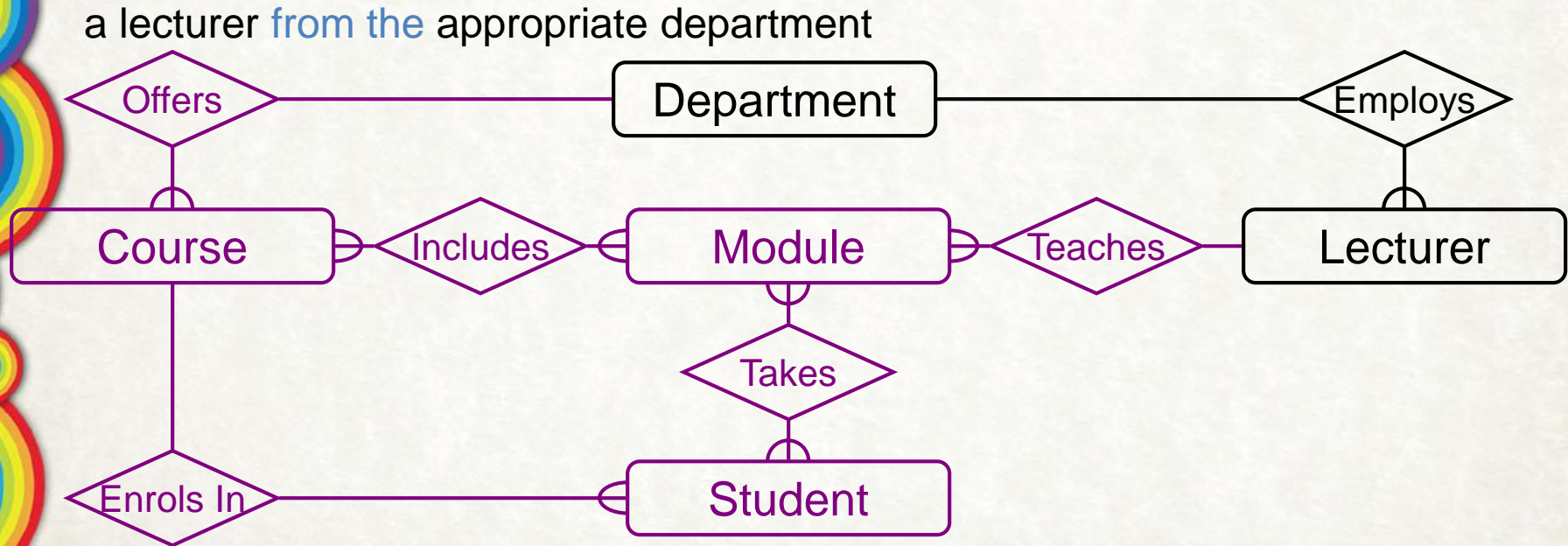


# Example - E/R Diagram

Each module is taught by a lecturer



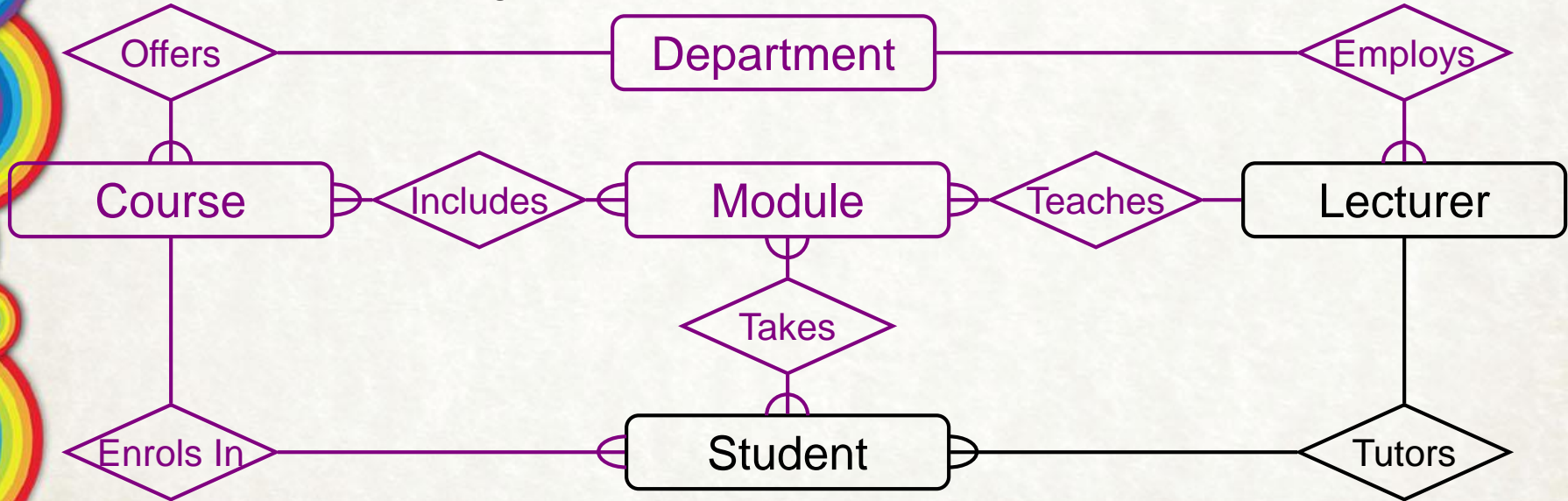
# Example - E/R Diagram



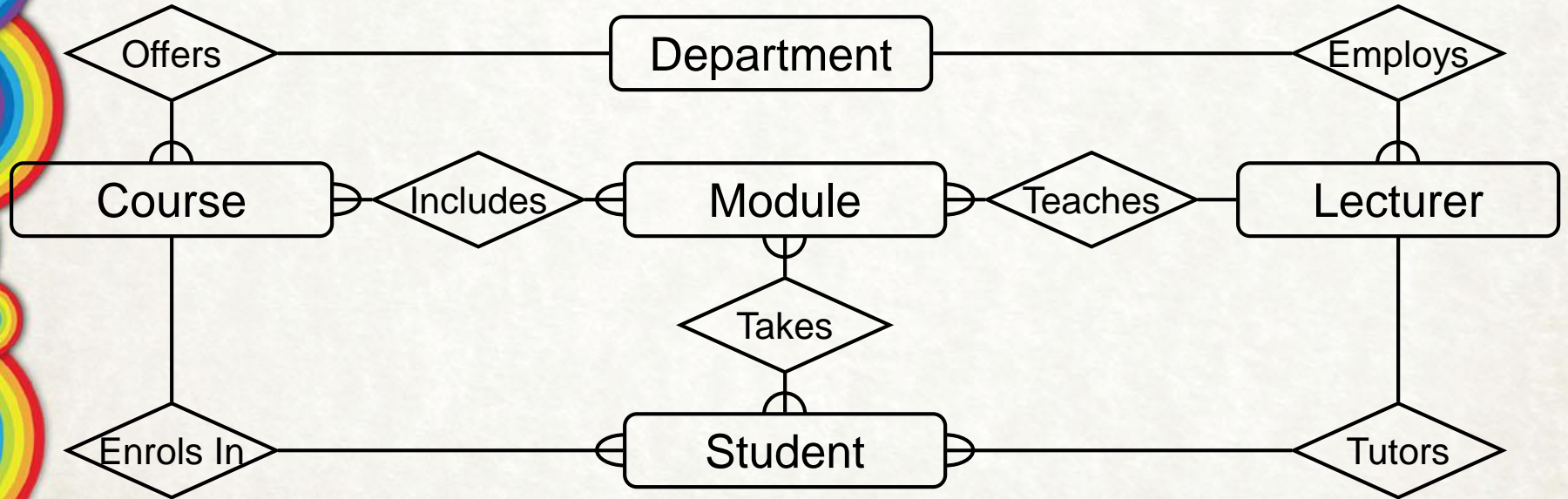


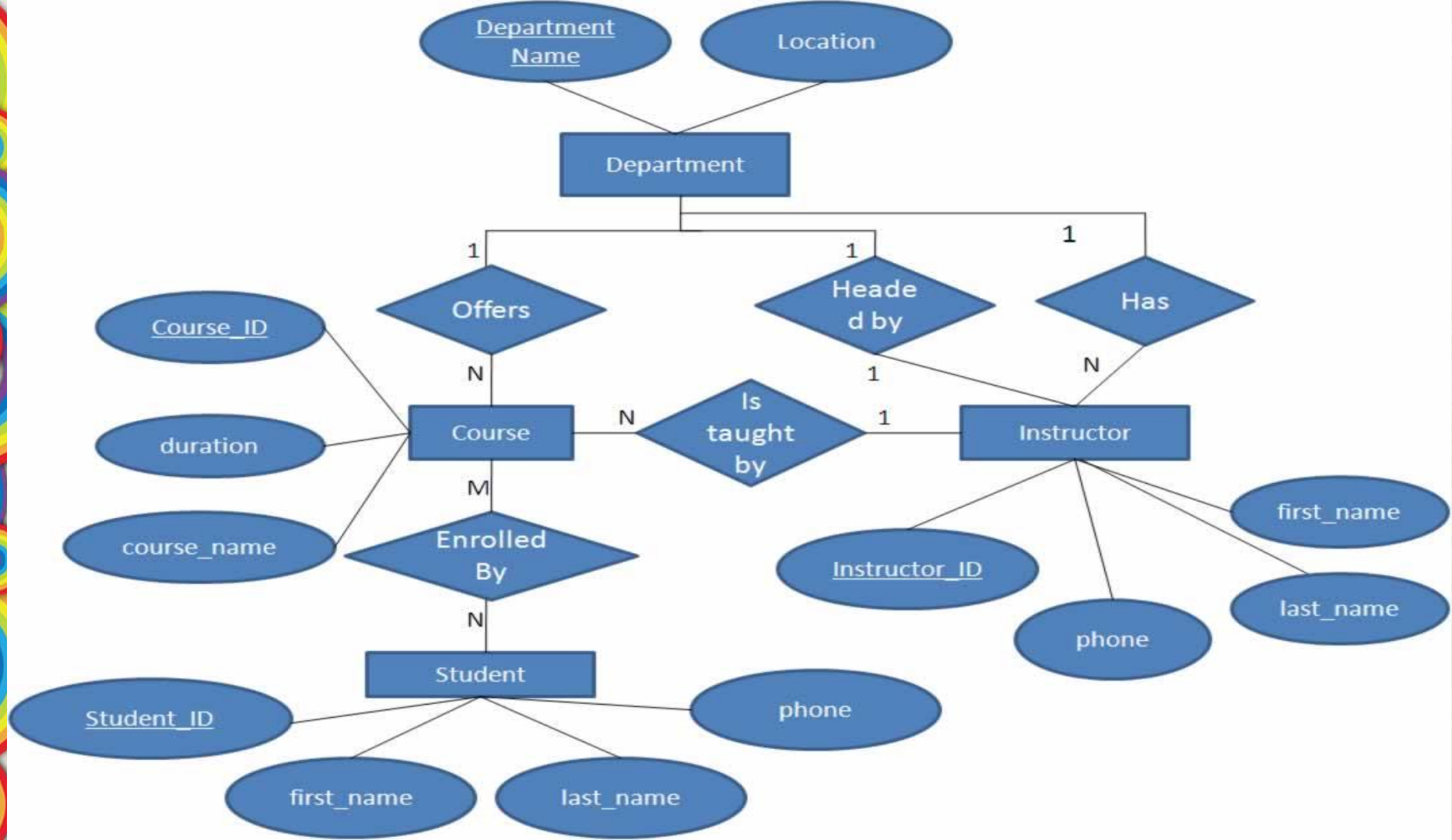
# Example - E/R Diagram


each lecturer **tutors** a group of students



# Example - E/R Diagram



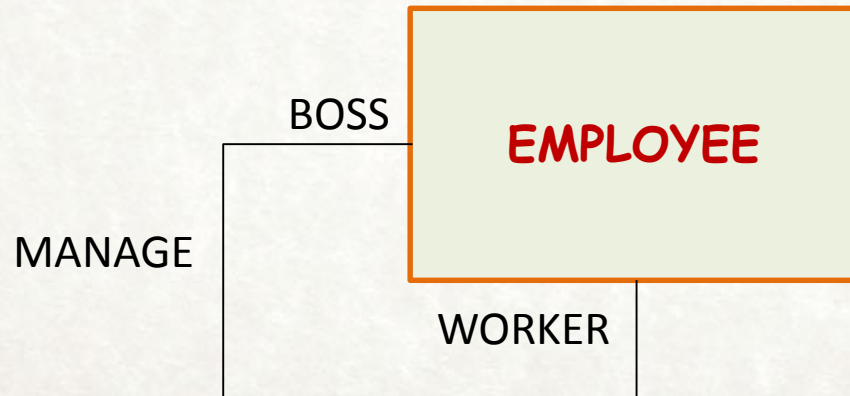


- 
- [https://www.slideshare.net/Tech\\_MX/er-diagram-14155323](https://www.slideshare.net/Tech_MX/er-diagram-14155323)

# Types of Relationship

1. Unary Relationship – a unary relationship exists when an association is maintained with in single entity.

For e.g. Boss and worker distinguish the two employees participating in the manage association as shown

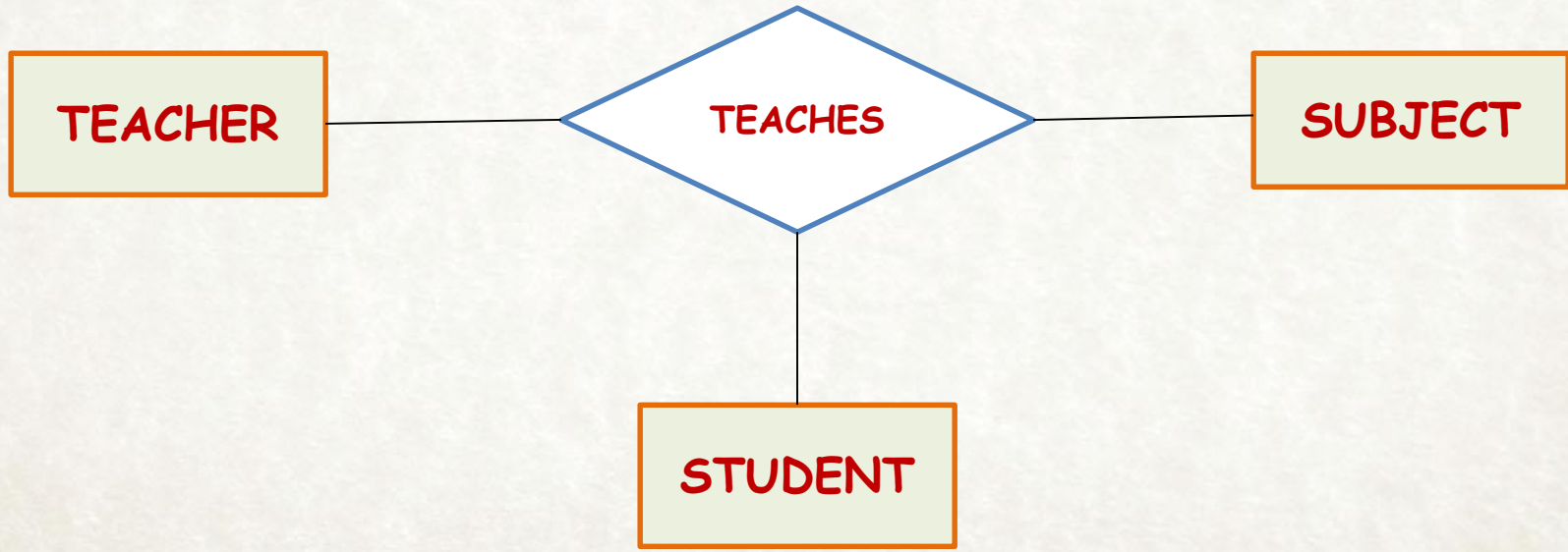


2. Binary Relationship – exists when two entities are associated .  
For e.g. the book – publisher relationship shown



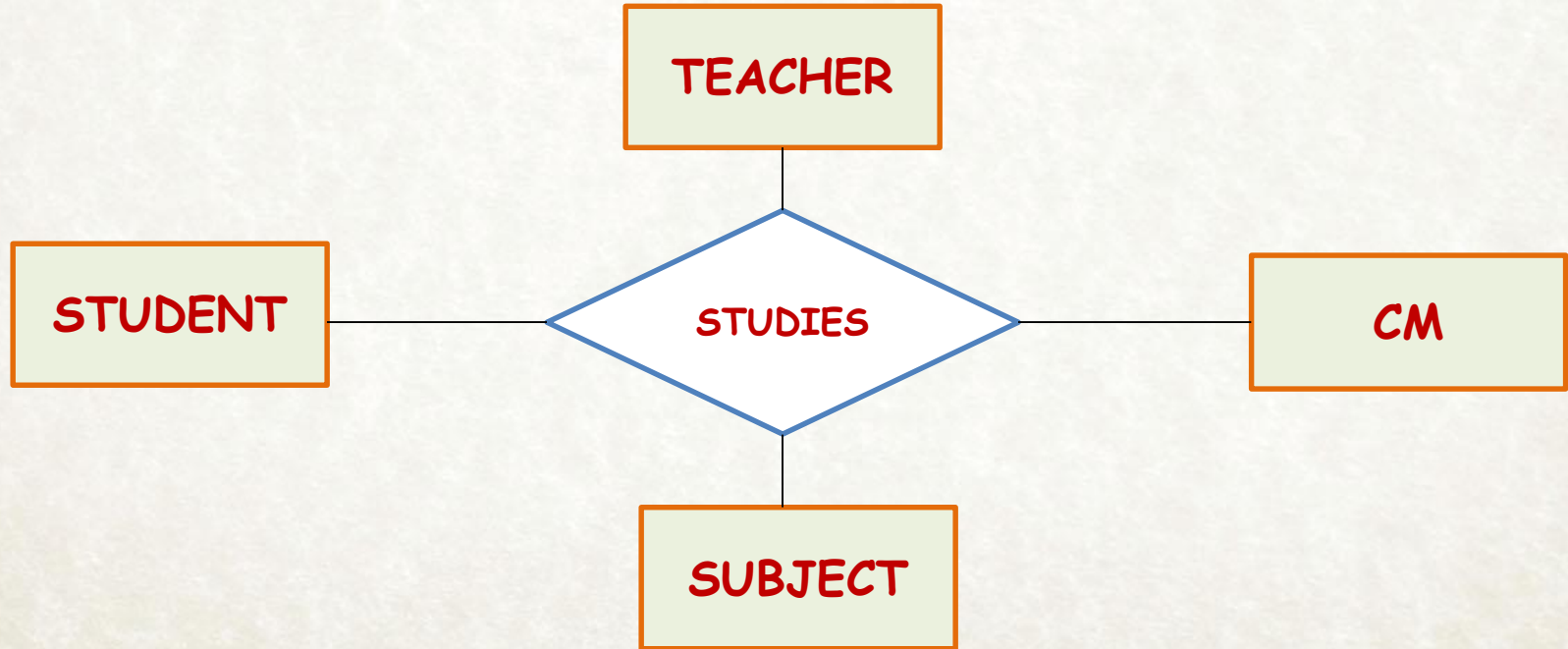
3. Ternary Relationship – exists when there are three entities associated.

For e.g. entities teacher , subject and student are related using a ternary relationship called 'Teaches' as shown in



4. Quaternary Relationship – exists when there are four entities associated.

for e.g. relationship is 'studies' where four entities are involved Student, Teacher, Subject and Course Material







# keys

- Key allows us to identify a set of attributes and thus distinguishes entities from each other.
- Keys also help uniquely identify relationship, and thus distinguish relationship from each other.

Different keys are :

1. Candidate Key
2. Super Key
3. Primary Key
4. Foreign Key

## ❑ Primary Key:

- An attribute or set of attributes that uniquely identifies a row or record in a relation/table is known as primary key.
- A relation can have only one Primary Key.
- Each value in Primary Key attribute must be unique.
- Primary Key cannot contain null values.

## ❑ Candidate Key:

- The attributes or set of attributes that can be used as primary key is called Candidate Key.

Reg No	Name	Class
10	Asad	C.A
20	Mohsin	P.H.D
30	Zeeshan	M.B.B.S

The attribute Reg No is Primary Key.

### ☐ **Alternate Key:**

- The candidate keys that are not selected as primary key are known as Alternate Keys.

### ☐ **Composite Key:**

- A primary key that consists of two or more attributes is known as Composite Key.

P<sub>i</sub>K      A<sub>i</sub>K  
↓            ↓

Reg No	Roll No	Name	Class
10	1	Asad	C.A
20	2	Mohsin	P.H.D
30	3	Zeeshan	M.B.B.S

C.P.K  
└──┬──┘

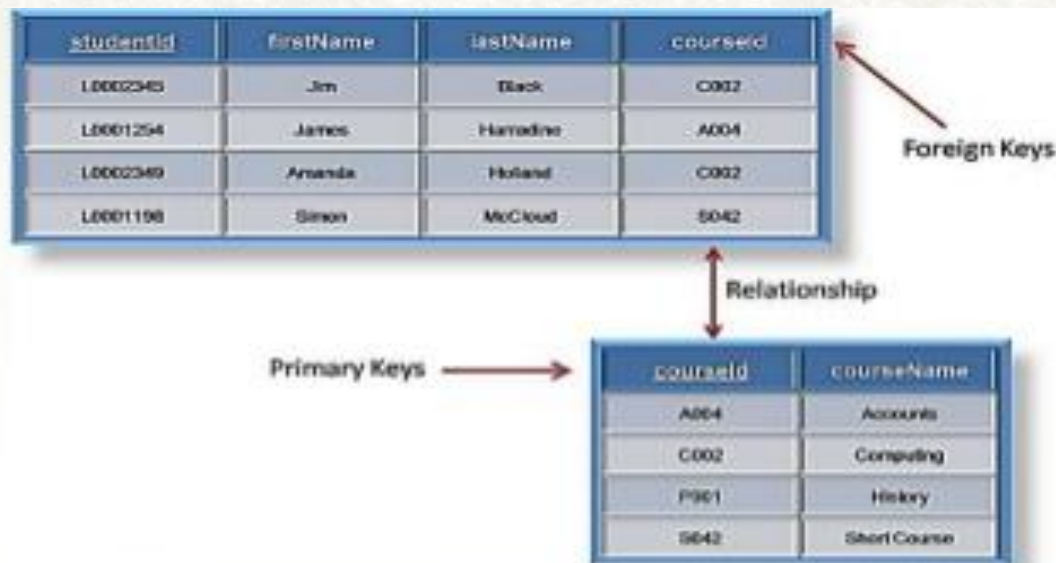
Roll No	Subject	Marks
1	English	75
1	Math	70
1	Computer	88
2	English	76
2	Math	71
2	Computer	89
3	English	76
3	Math	72
3	Computer	90

## Foreign Key:

- A Foreign Key is an attribute or set of attributes in a relation/table whose values match a primary key in another relation.

## Secondary Key:


- An attribute or set of attributes that is bases for retrieval is called S.K
- One Secondary Key value may refer to many records.
- Secondary Key can be a non-unique attribute.



Primary  
Key

Secondary Key

Roll_No	Name	Branch	City
01	Deepak	Computers	Bhiwani
02	Mukesh	Electronics	Rohtak
03	Teena	Computers	Bhiwani
04	Deepak	Electronics	Rohtak
05	Monika	Computers	Delhi



Customer ID	Forename	Surname
1	Simon	Jones
2	Emma	Price
3	Laura	Jones
4	Jonathan	Hale
5	Emma	Smith

Simple primary key

Suit	Value	No. times played
Hearts	Ace	5
Diamonds	Three	2
Hearts	Jack	3
Clubs	Three	5
Spades	Five	1

One is not enough to identify, but both combined make a unique value

### S\_DEPT

Primary key  
v

Foreign key  
v

id	name	region_id
10	Finance	1
31	Sales	1
32	Sales	2
33	Sales	3
34	Sales	4
35	Sales	5

### S\_REGION

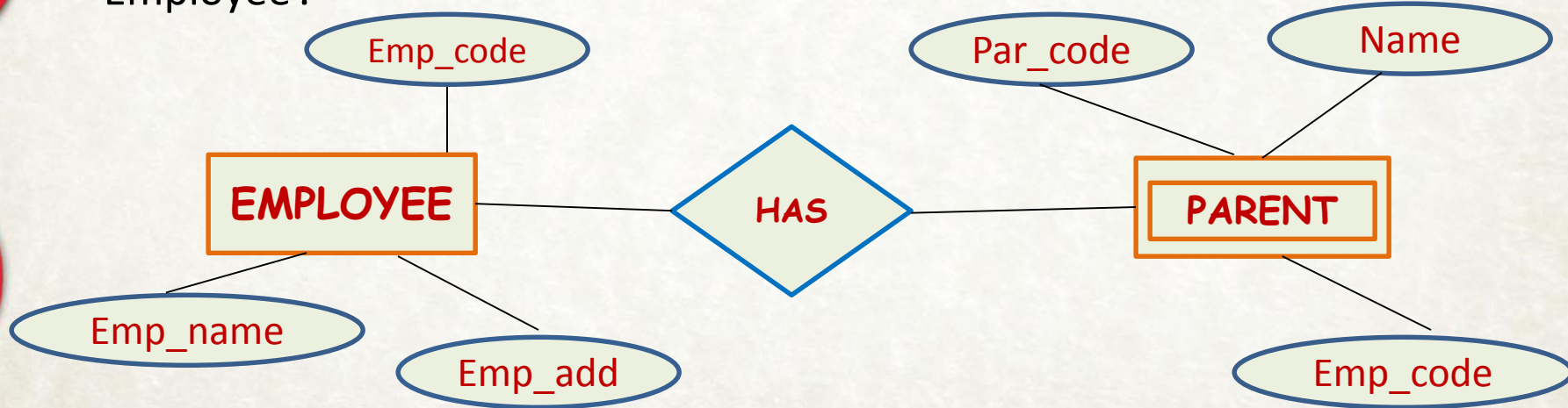
Primary key  
v

Region_id	name
1	North
2	South
3	Africa
4	Asia
5	Europe

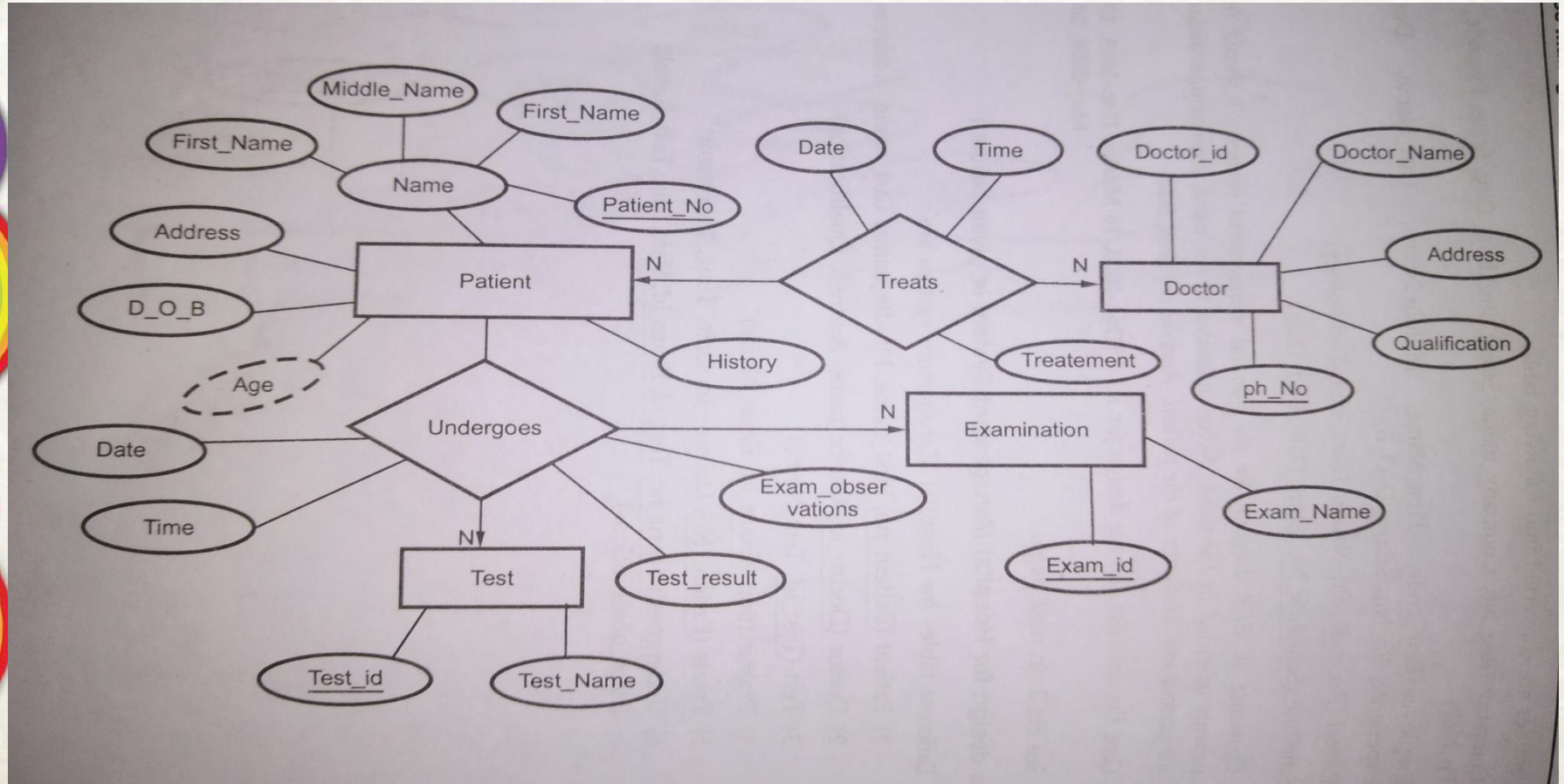


## Strong and weak entity sets

- The entity that is existence dependent on some other entity is called a weak entity type.
- The entity set on which weak entity set depends, is called strong entity set.
- For e.g. weak entity set 'Parent' which depends on strong entity set 'Employee'.



# Hospital Management ER Diagram



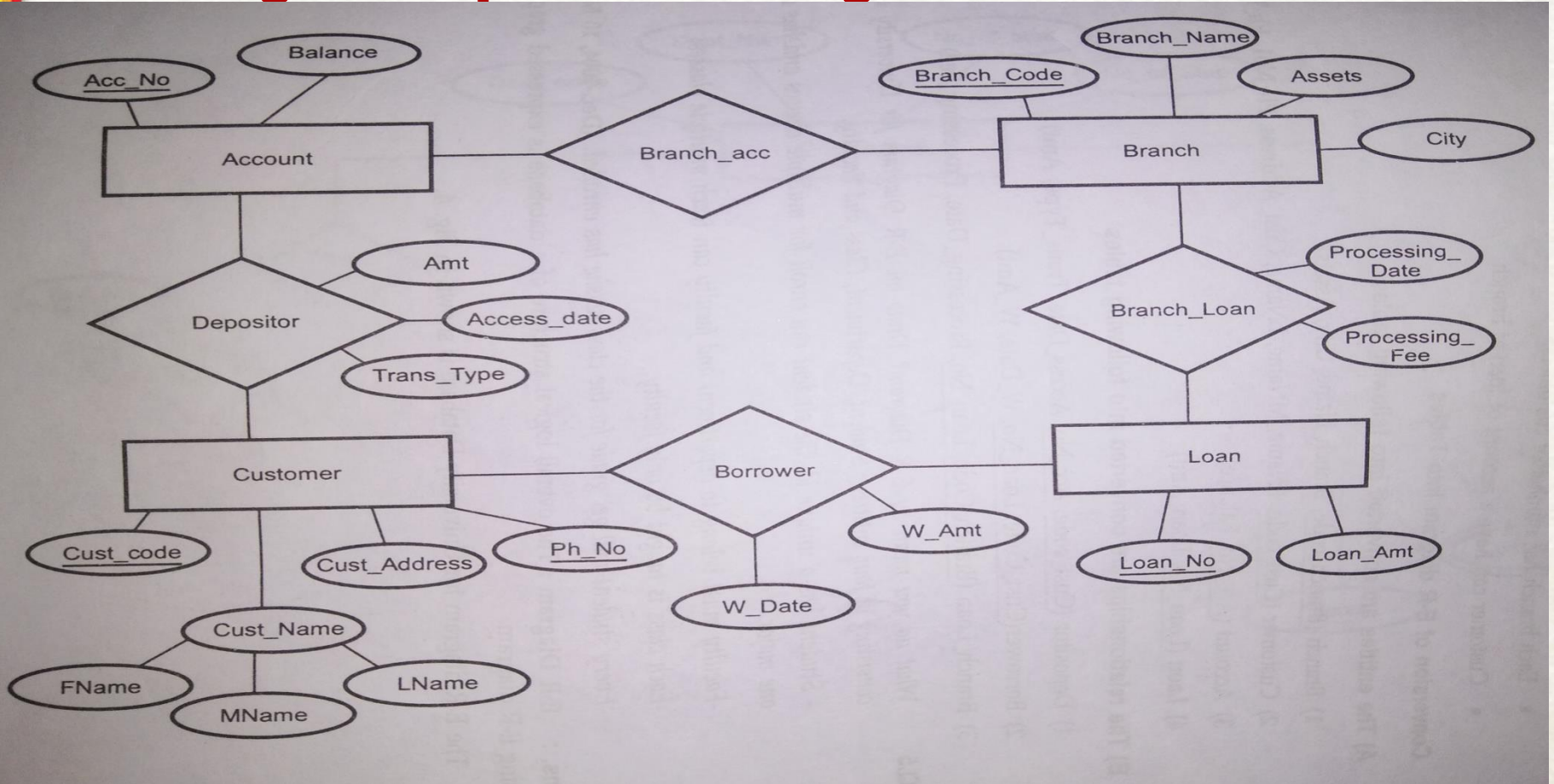


Database design for Hospital Management System is given below :

Different tables for Hospital Management system are :

- 1) Patient (Patient\_no, First\_name, Middle\_name, Last\_name, Address, D\_O\_B)
- 2) Doctor (Doctor\_id, Doctor\_name, Address, Qualification).
- 3) Test (Test\_id, Test\_Name)
- 4) Examination (Exam\_id, Exam\_Name)
- 5) Treats (Patient\_No, Docotor\_id, Date, Time, Treatment).
- 6) Undergoes(Patient\_no, Test\_id, Exam\_id, Date,Time,Test\_Result, Exam\_observations).

# Banking Enterprise ER Diagram



### Assumption Made :

- Here it is assumed that the bank has number of branches at different cities.
- Each branch has number of customers.
- Customer can have a account or loan at branch.

### Conversion of E-R diagram into Tables

#### A) The entities are converted into following tables

- 1) Branch (Branch\_code, Branch\_Name, City, Assets)
- 2) Customer (Cust\_code, FName, MName, LName, Cust\_Address, Ph\_No1, Ph\_No2)
- 3) Account (Acc\_No, Balance)
- 4) Loan (Loan\_No, Loan\_Amt)

#### B) The relationships are converted into following tables

- 1) Depositor (Cust\_code, Acc\_No, Access\_Date, Trans\_Type, Amt)
- 2) Borrower (Cust\_Code, Loan\_No, W\_Date, W\_Amt)
- 3) Branch\_Loan (Branch\_Code, Loan\_No, Processing\_Date, Processing\_Fee)

# Question

What do you mean by E-R Diagram? Draw an E-R Diagram for University database consisting of four entities : Student, Department, Class and Faculty.

- Student has a unique id, the student can enroll for multiple classes and has at most one major.

-Faculty must belong to department and faculty can teach multiple classes.

-Each class is taught by only faculty.

-Every student will get grade for the class he/she has enrolled. (Dec.-2004, 10 Marks)

...hically

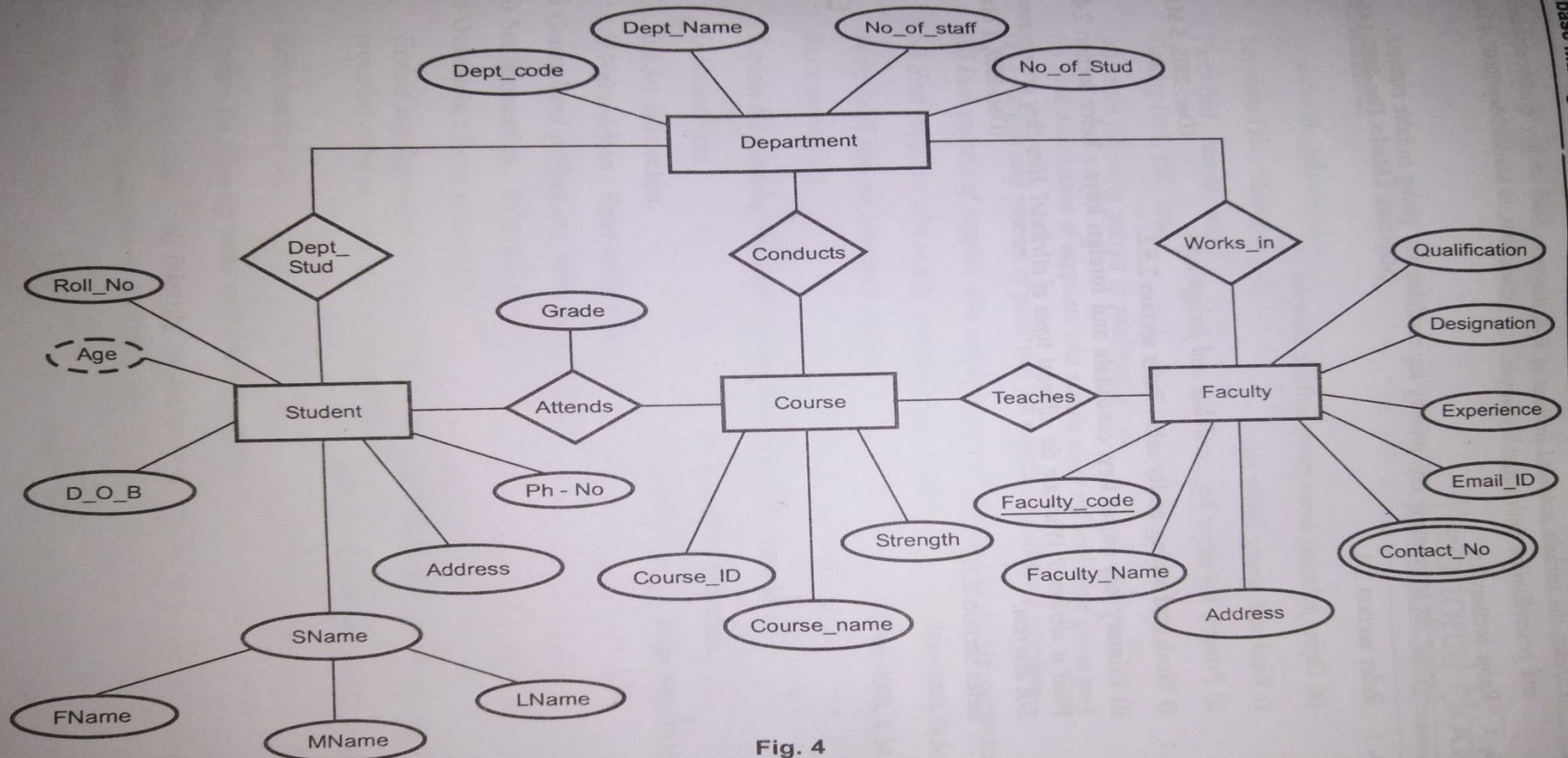


Fig. 4

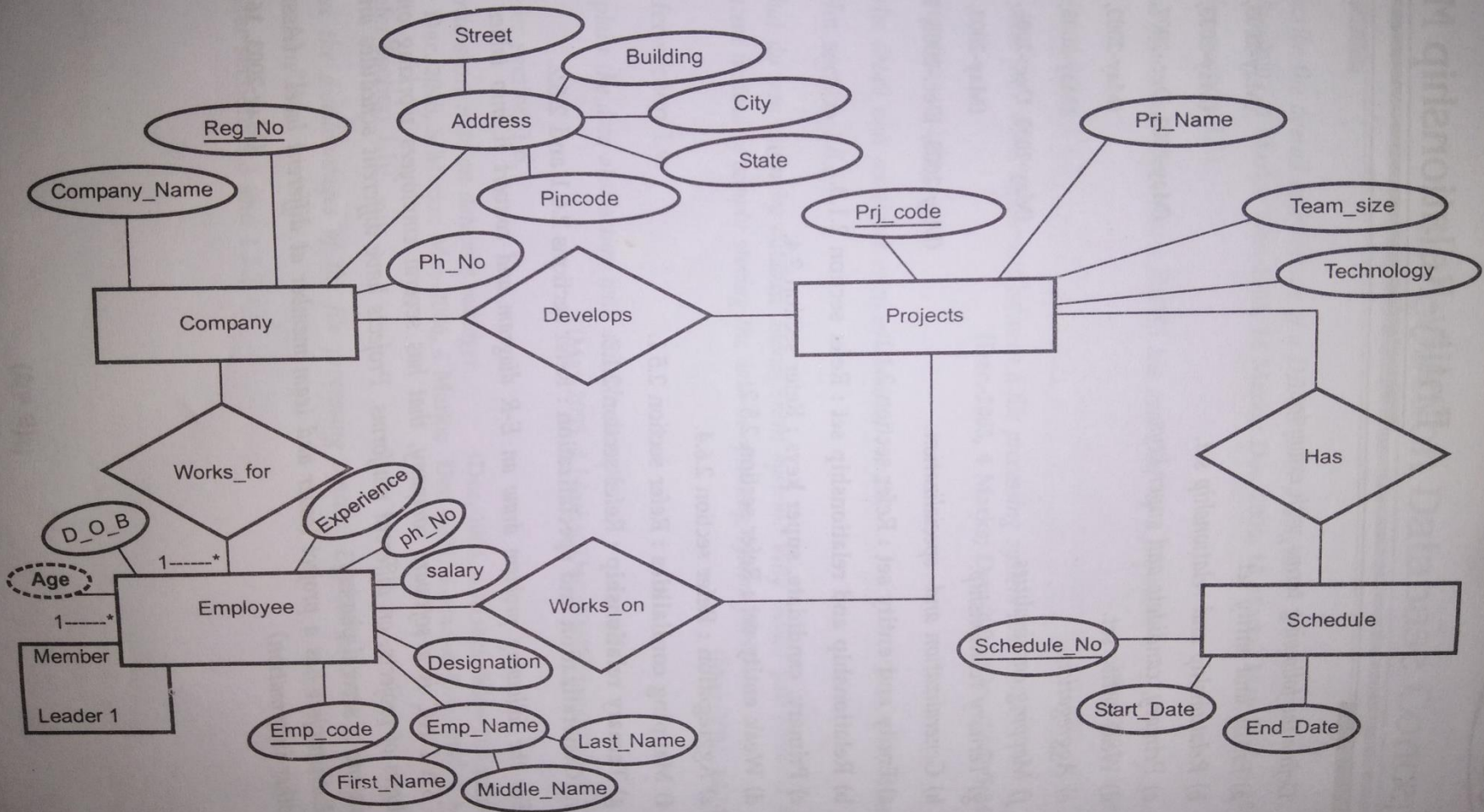
# Question

For the following problem draw an E-R diagram and convert it into tables using the rules.

"SW-INC is a software company, that has several employees working on different types of projects, on different platforms. Projects have different schedules and may be in one of several phases.

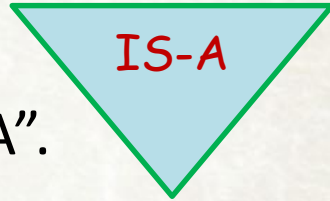
Each project has a project leader and team member at different level". (Assume any other information)

(May-2003, 16 Marks)



# What is Generalization ?

- A generalization hierarchy is a form of abstraction that specifies that two or more entities that share common attributes can be generalized into higher level entity type called as super type.
- It is a bottom up approach.
- It is denoted with triangular component labeled “IS-A”.
- E.g.. Account is a higher entity set where saving account and current account is the lower entity set.





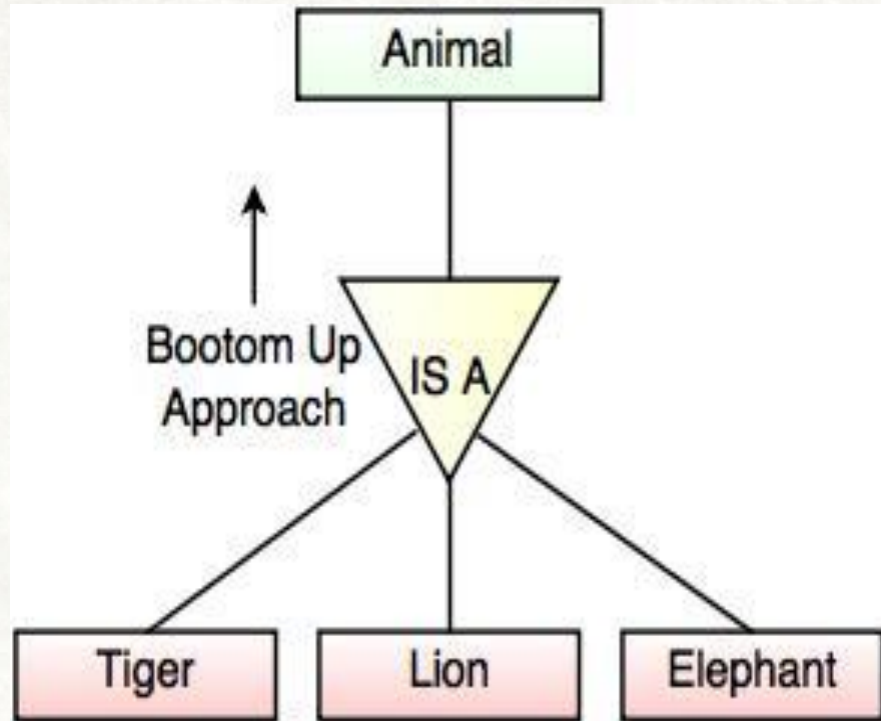
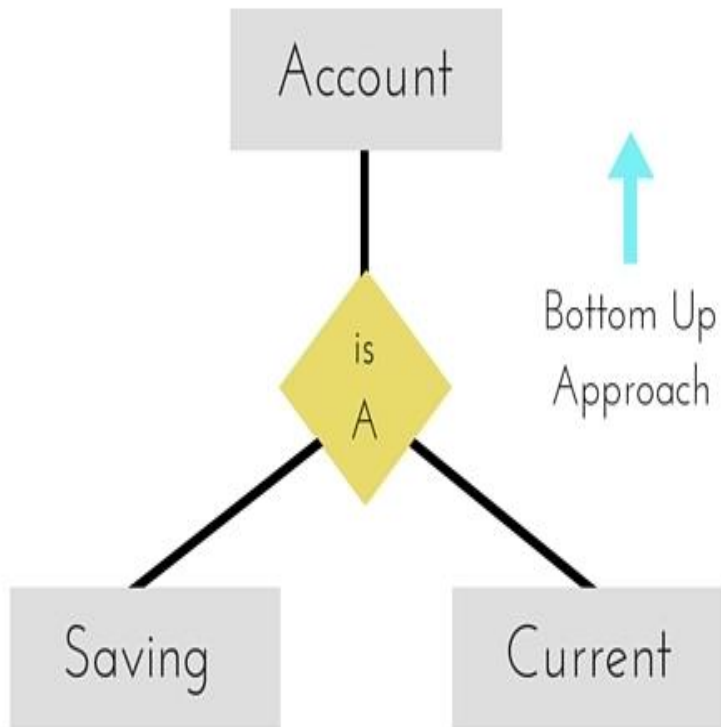
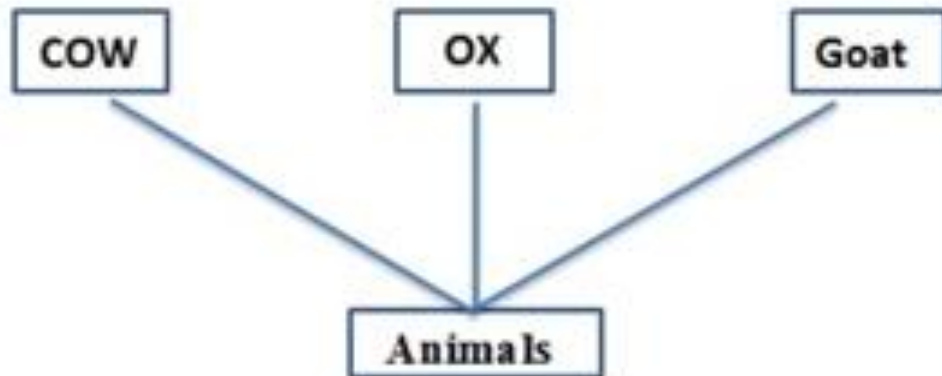


Fig. Generalization

**Generalization:** - In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics. For example, Cow, Ox, and Goat can all be generalized as Animals.



**Fig: - Generalization**

# What is Specialization ?

- Specialization is the **opposite** of generalization.
- In specialization, a group of entities is divided into sub-groups based on their characteristics.
- Taking subset of the higher level entity set to form lower level entity sets.
- It's a Top Down Approach.
- Sub classes of entity type is called super class of specialization.
  
- Take a group 'Person' for **EXAMPLE**.

A person has name, date of birth, gender, etc. These properties are common in all persons, human beings. But in a company, persons can be identified as employee, employer, customer, or vendor, based on what role they play in the company..

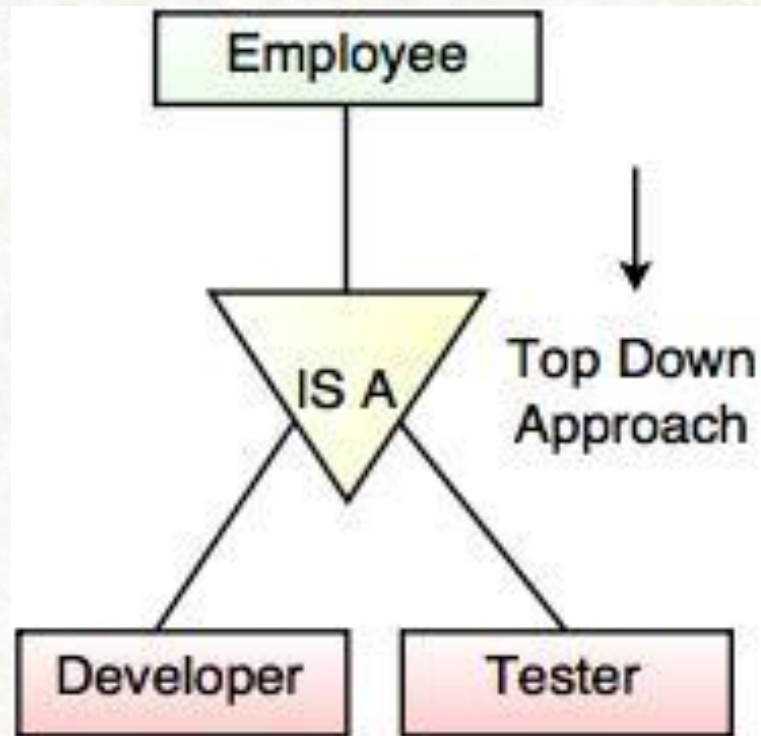
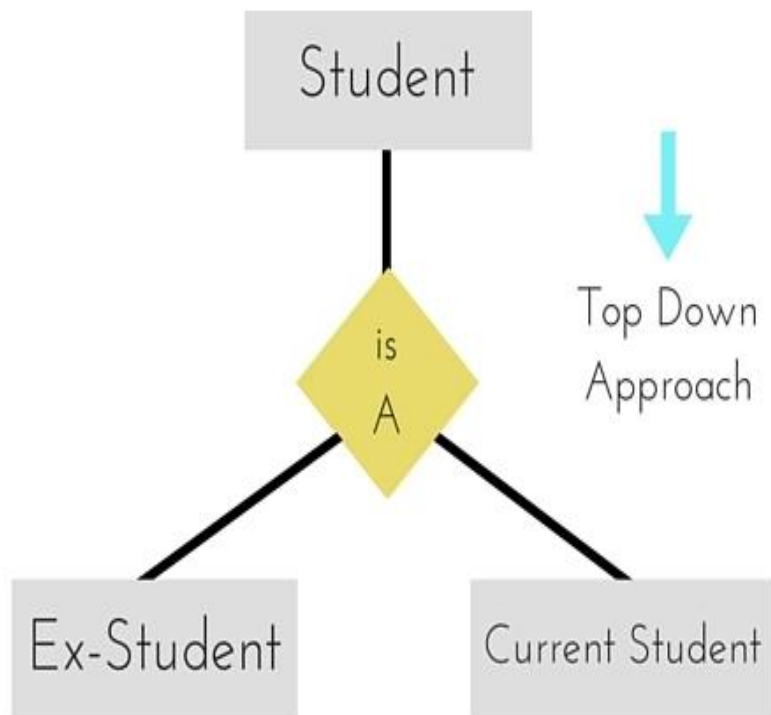
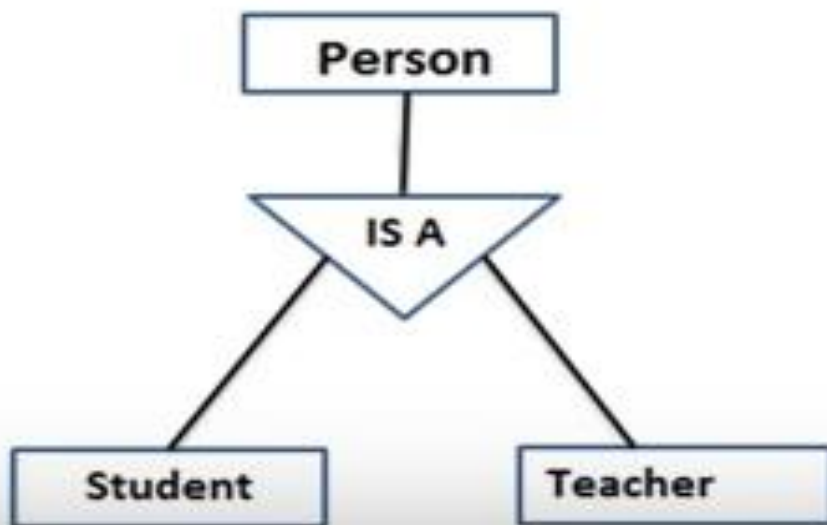



Fig. Specialization

**Specialization:** - A group of entities is divided into sub-groups based on their characteristics. Specialization is the just opposite of generalization. For example a person can be identified as student and teacher.

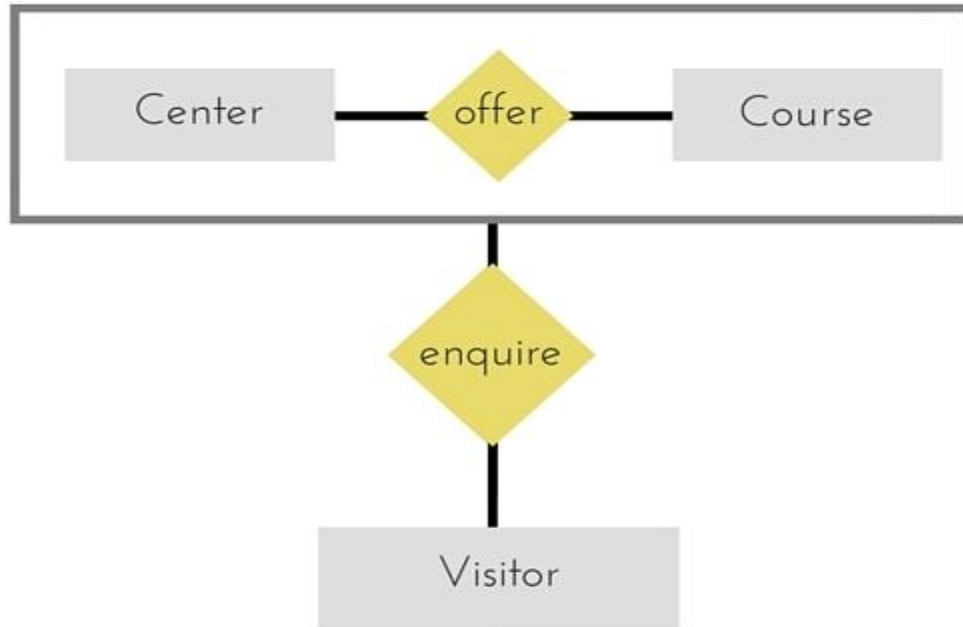




Basis of Distinction	Generalization in DBMS	Specialization in DBMS
Definition	The process in which all the sets of objects get viewed as the same class focused on the characteristics they have in common and ignores the differences between them.	The process in which all the sets of objects get viewed as individual sets based on their characteristics that make them different from others and ignores the similarities between them.
Approach	Bottom-up	Top-down
Working	Takes all the information that have universal nature within the entities and then forms a new entity.	Creates new objects based on the difference between the existing ones and have some features of the parents.

# What is Aggregation ?

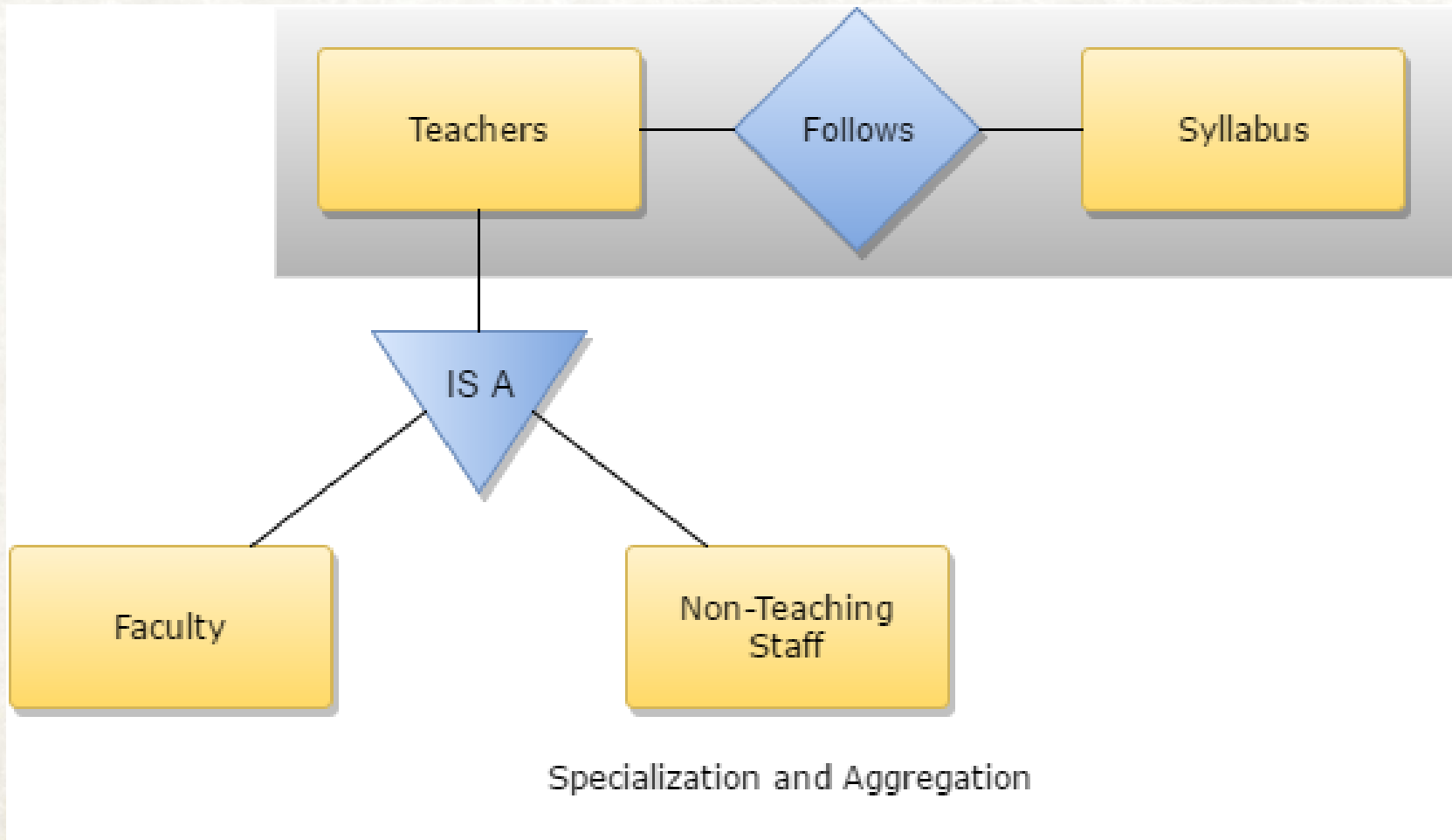
- Aggregation is a process when relation between two entities is treated as a **single entity**.



In the diagram

The relationship between **Center** and **Course** together, is acting as an **Entity**, which is in relationship with another entity **Visitor**.

Now in real world, if a **Visitor** or a **Student** visits a Coaching Center, he/she will never enquire about the center only or just about the course, rather he/she will ask enquire about both.





# Codd's rules



- **Dr. Edgar Frank Codd** (August 19, 1923 – April 18, 2003) was a computer scientist while **working for IBM** he invented the relational model for database management ( theoretical basis for relational databases).
- Codd proposed **thirteen rules** (numbered zero to twelve) and said that if a Database Management System meets these rules, it can be called as a **Relational Database Management System**.
- These rules are called as Codd's 12 rules.



## Rule Zero

- The system must qualify as relational, as a database, and as a management system.

## Rule 1 - The information rule :

- All information in the database (Data or Meta data) is to be represented in one and only one way, namely by values in column positions within rows of tables.

## Rule 2 - The guaranteed access rule:

- All data must be accessible.
- Each and every datum (atomic value) in a relational data base is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name.

# Primary Key



ID	NAME	CLASS	MARK	SEX
1	John Deo	Four	75	female
2	Max Ruin	Three	85	male
3	Arnold	Three	55	male
4	Krish Star	Four	60	female
5	John Mike	Four	60	female
6	Alex John	Four	55	male
7	My John Rob	Fifth	78	male
8	Asruid	Five	85	male
9	Tes Qry	Six	78	male
10	Big John	Four	55	female



### Rule 3 - Systematic treatment of null values:

- Given a systematic and uniform treatment.
- NULL is data is missing, data is not known and data is not applicable.
- The DBMS must allow each field to remain null (or empty).

### Rule 4 - Active online catalog based on the relational model:

- The system must support an online, inline, relational catalog that is accessible to authorized users by means of their regular query language.
- That is, users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data.



## Rule 5 - The comprehensive data sub language rule:

The system must support at least one relational language that

- Has a linear syntax
- Supports data definition operations (including view definitions), data manipulation operations (update as well as retrieval), security and integrity constraints, and transaction management operations (begin, commit, and rollback).
- Database allow access to data without any help of this language, then it is considered violation.

## Rule 6 - The view updating rule:

- All views those can be updated theoretically, must be updated by the system.



## Rule 7 - High-level insert, update, and delete:

- The system must support set-at-a-time insert, update, and delete operators.

## Rule 8 - Physical data independence:

- Changes to the physical level (how the data is stored, whether in arrays or linked lists etc.) must not require a change to an application based on the structure.

## Logical data independence:

- Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind that theoretically permit un impairment are made to the data base tables.

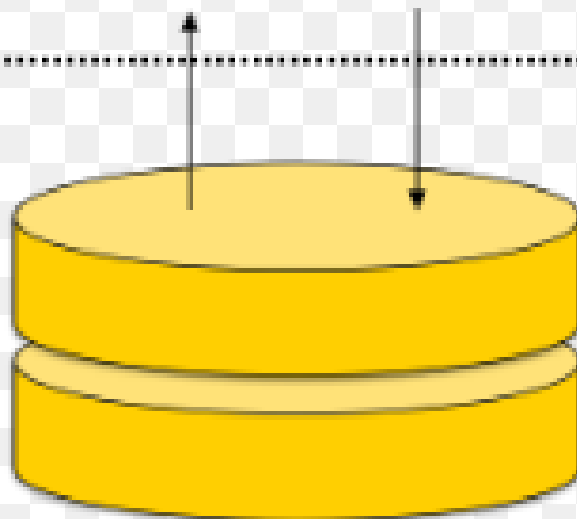
Student

Logical Schema

Stu\_ID

Stu\_Name

Proj\_ID



Physical Schema



## Rule 10: Integrity independence:

- Integrity constraints specific to a particular relational data base must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.

## Rule 11: Distribution independence:

- The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only.

## Rule 12: The non-subversion rule:

- If a relational system has a low-level (single-record-at-a-time) language, that low level cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language (multiple-records-at-a-time).