

# Chapter 3

Data models

# Contents

- Design Process
- Modeling
- Entity
- Relationship
- Constraints
- Attribute
- E-R Diagram
- Extended E-R Features
- Reduction to Relation Schemas
- Relational Model
- Keys
- Integrity Constraints

# Design Phases

- The initial phase of database design is to characterize fully the data needs of the prospective database users.
- Next, the designer chooses a data model and, by applying the concepts of the chosen data model, translates these requirements into a conceptual schema of the database.
- A fully developed conceptual schema also indicates the functional requirements of the enterprise. In a “specification of functional requirements”, users describe the kinds of operations (or transactions) that will be performed on the data.

# Design Phases (Cont.)

The process of moving from an abstract data model to the implementation of the database proceeds in two final design phases.

- Logical Design – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

# Design Approaches

- Entity Relationship Model

Models an enterprise as a collection of *entities* and *relationships*

- Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
  - Described by a set of *attributes*
- Relationship: an association among several entities
- Represented diagrammatically by an *entity-relationship diagram*

# ER model -- Database Modeling

- The ER data model was developed to facilitate database design by allowing specification of an **enterprise schema** that represents the overall logical structure of a database.
- The ER model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema. Because of this usefulness, many database-design tools draw on concepts from the ER model.
- The ER data model employs three basic concepts:
  - entity sets,
  - relationship sets,
  - attributes.
- The ER model also has an associated diagrammatic representation, the ER diagram, which can express the overall logical structure of a database graphically.



Entity Name

### Entity

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

**Example:** Car, Student



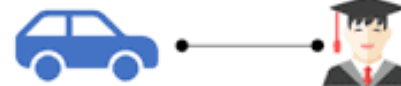
Jack

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 Sets

- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays
- An entity is represented by a set of attributes; i.e., descriptive properties possessed by all members of an entity set.
  - Example:  
*instructor = (ID, name, street, city, salary )*  
*course = (course\_id, title, credits)*
- A subset of the attributes form a **primary key** of the entity set; i.e., uniquely identifying each member of the set.



# Entity Sets -- *instructor* and *student*

instructor\_ID instructor\_name

76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

*instructor*

student-ID student\_name

98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

*student*

# Relationship Sets

- A **relationship** is an association among several entities

Example:

44553 (Peltier)   advisor   22222 (Einstein)  
*student* entity-relationship set *instructor* entity

- A **relationship set** is a mathematical relation among  $n \geq 2$  entities, each taken from entity sets

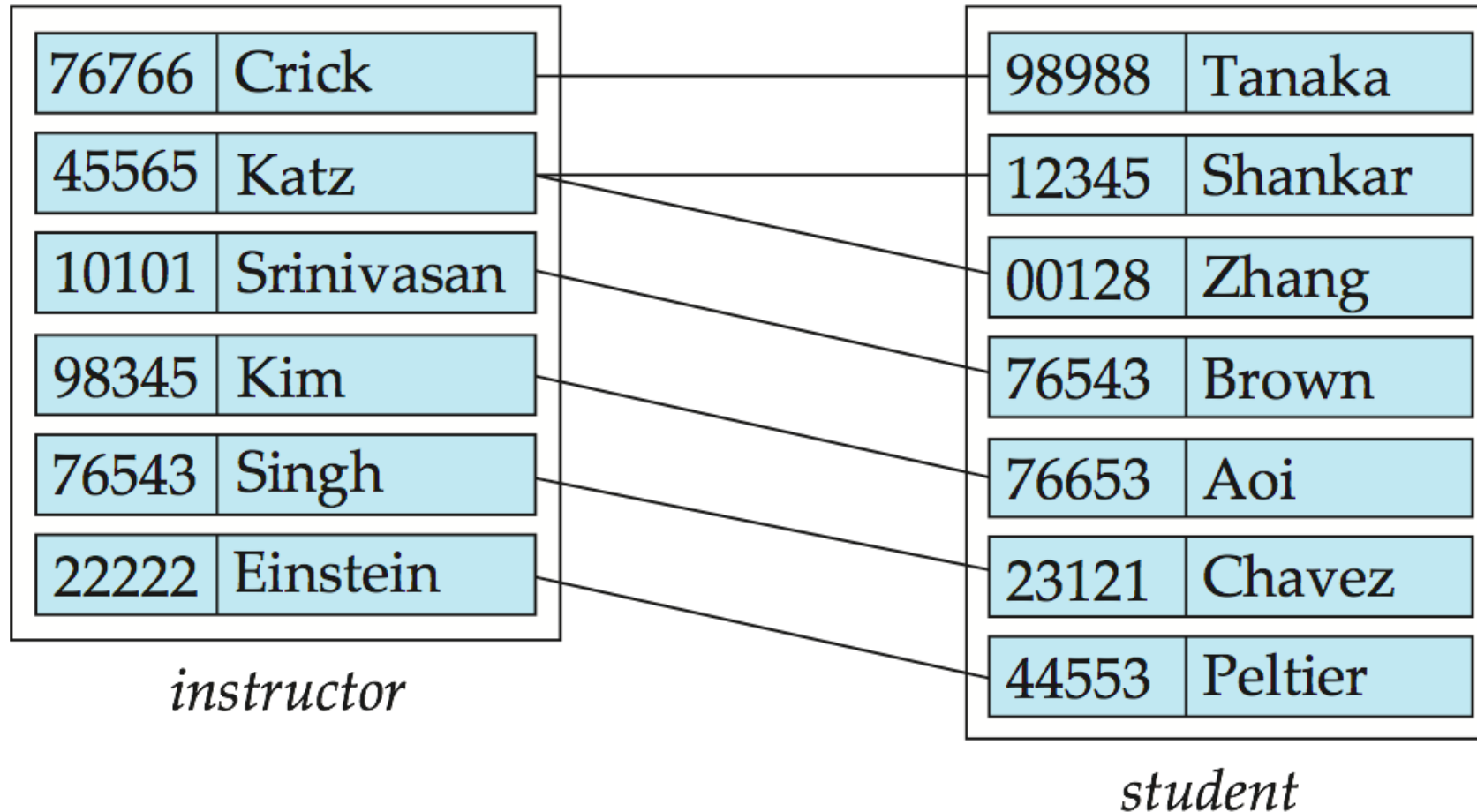
$$\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$$

where  $(e_1, e_2, \dots, e_n)$  is a relationship

- Example:

$(44553, 22222) \in \text{advisor}$

# Relationship Set *advisor*

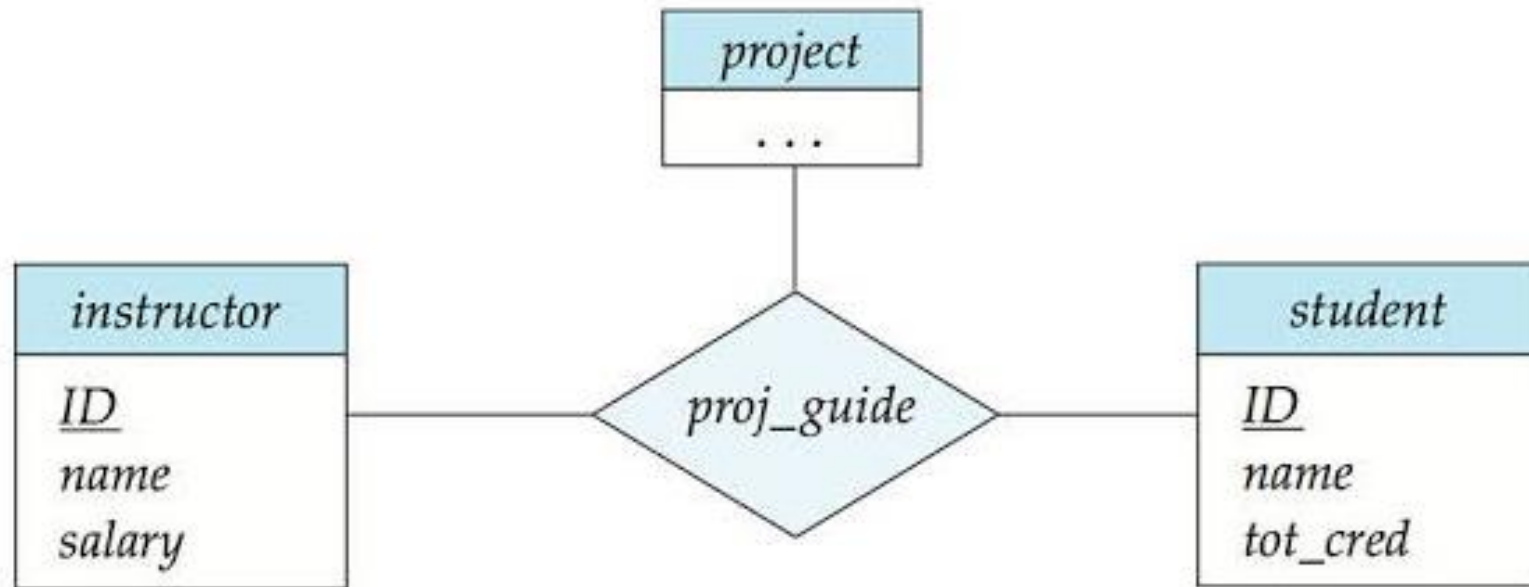


# Degree of a Relationship Set

- Binary relationship
  - involve two entity sets (or degree two).
  - most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare. Most relationships are binary.

Example: *students* work on research *projects* under the guidance of an *instructor*.

- ▶ relationship *proj\_guide* is a ternary relationship between *instructor*, *student*, and *project*



Ternary Relationship

# Constraints

## A) Mapping Cardinality/Cardinality Ratio

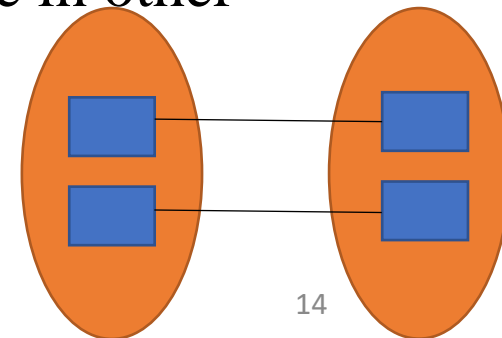
It express the number of entities to which another entity can be associated via a relationship set.

Express the number of entities to which another entity can be associated via a relationship set.

Types of mapping cardinalities:

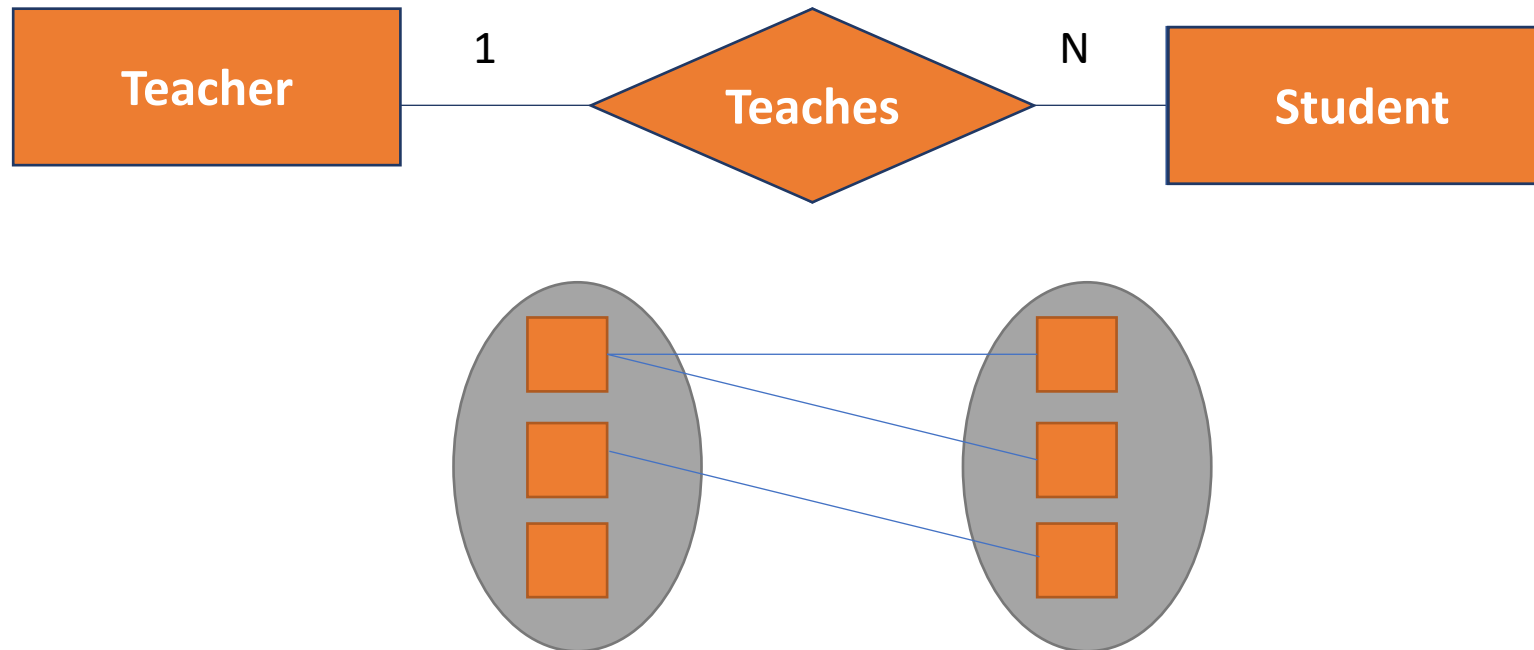
### 1. One is to One

In this type of constraint one tuple in entity is related with only one tuple in other entity. i.e. One row in table is related with only one row in other table.



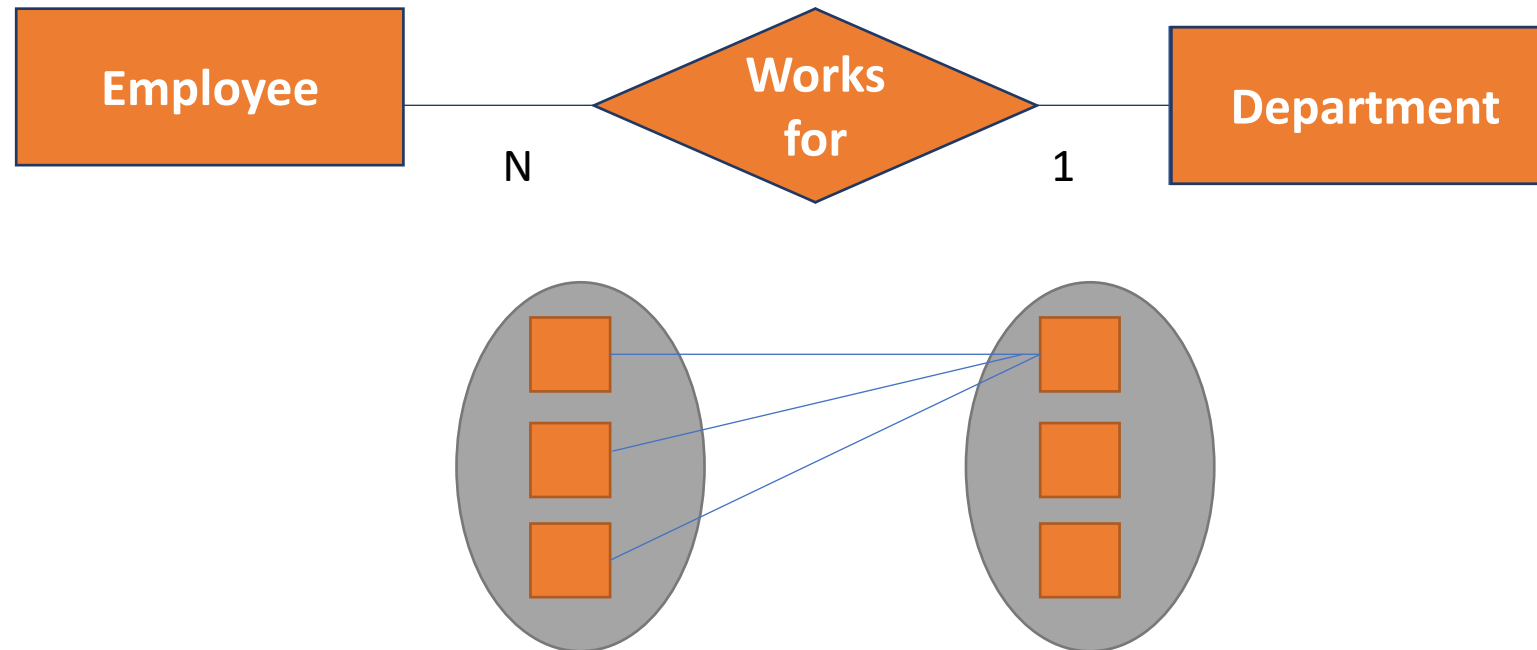
## 2. One to Many

In this type of constraint one tuple in entity can be related with many tuples in other entity.



### 3. Many to One

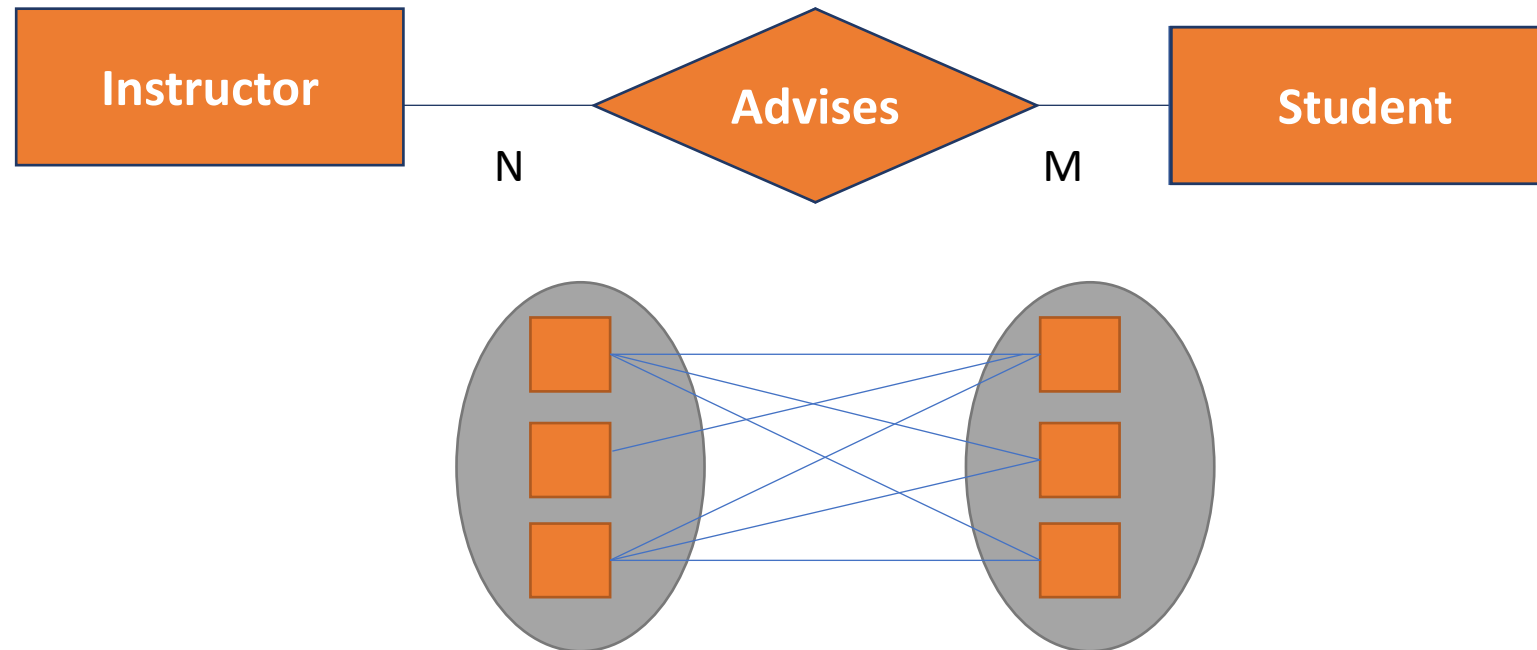
- In this type of constraint many tuple in entity can be related with only one tuple in other entity.





## 4. Many to Many

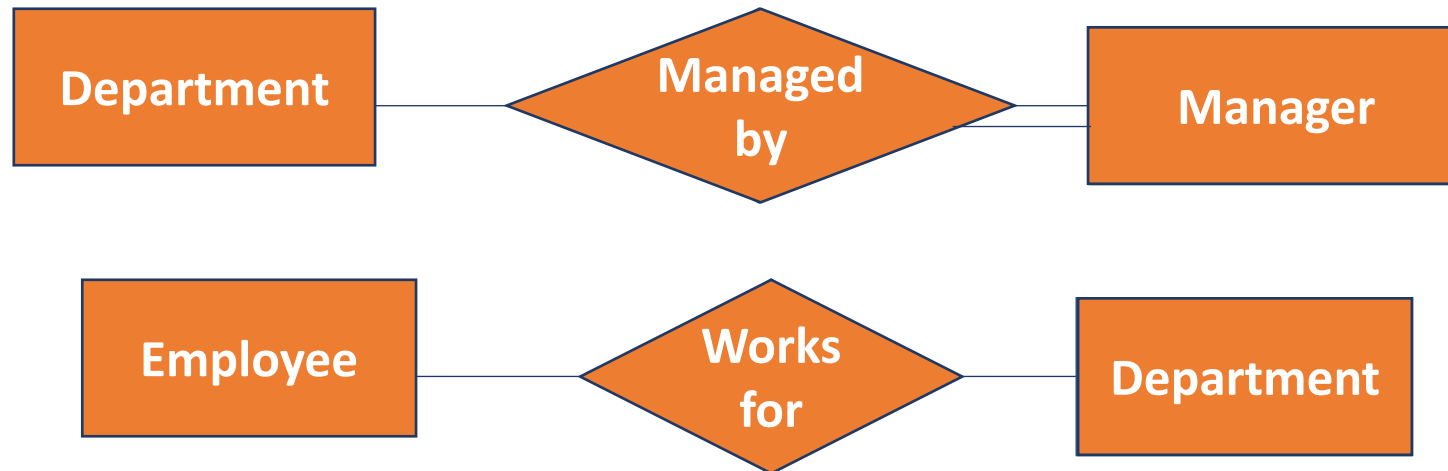
- In this type of constraint many tuples in constraint can be related with multiple tuples in other entity.



## B) Participation constraint

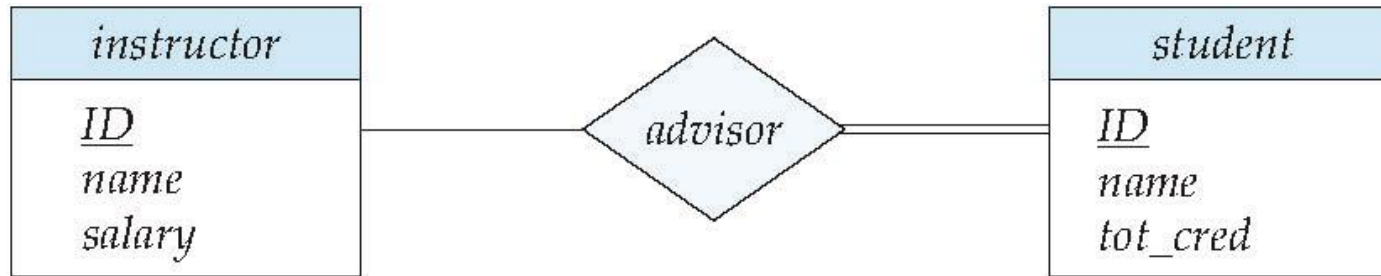
A participation of an entity set E in a relationship set R is said to be **“TOTAL”** if every entity in E participates in at least one relationship in R

If only some entities in E participates in relationships in R, the participation of entity set E in relationship R is said to be **“PARTIAL”**.



# Total and Partial Participation

- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set



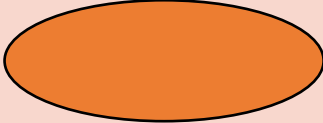
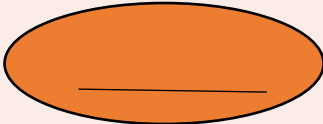
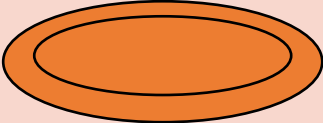
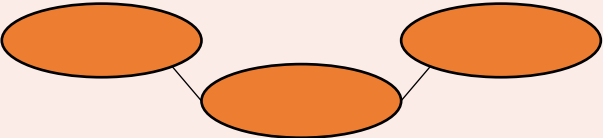

- participation of *student* in *advisor* relation is total
  - ▶ every *student* must have an associated instructor
- Partial participation: some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial

# ATTRIBUTE TYPE

- Attribute types:

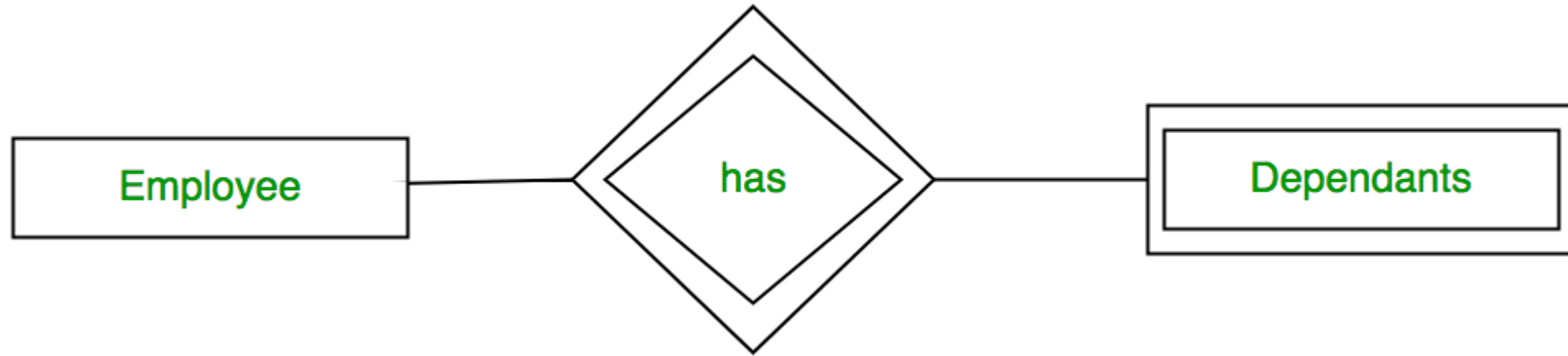
1. **Simple Attributes** – Which can not be divided to subparts
2. **Composite Attributes** – Which can be further divided into subparts
3. **Single Valued** – The attribute having only one value for a particular entity
4. **Multivalued** – The attribute having many values for a particular entity
5. **Stored** – The simple attribute stored in database
6. **Derived** – The value of this attribute can be derived from the value of related stored attribute

# ER Notation for various types of attributes

Type	Notation
Attribute (Simple/Single valued/ Stored)	
Key Attribute	
Multi Valued	
Composite	
Derived	

# WEAK ENTITY

- A weak entity set is one whose existence is dependent on another entity, called its **identifying entity**; instead of associating a primary key with a weak entity, we use the identifying entity, along with extra attributes called **discriminator** to uniquely identify a weak entity. An entity set that is not a weak entity set is termed a **strong entity set**.
- Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set. The identifying entity set is said to **own** the weak entity set that it identifies. The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.


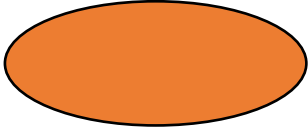
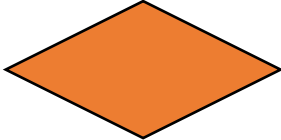
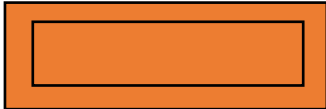


**WEAK ENTITY**

# E-R DIAGRAM



# NOTATION USED

- Rectangle: Entity 
- Ellipses: Attributes 
- Diamonds: Relationship Sets 
- Lines: Link attributes to entity sets
- Double lines: Total Participation of an entity in a relationship set
- Double Rectangle: Weak Entity sets 

Draw ER Diagram for car insurance company that has a set of customer each of whose having one or more car. Each car associated with any number of accidents

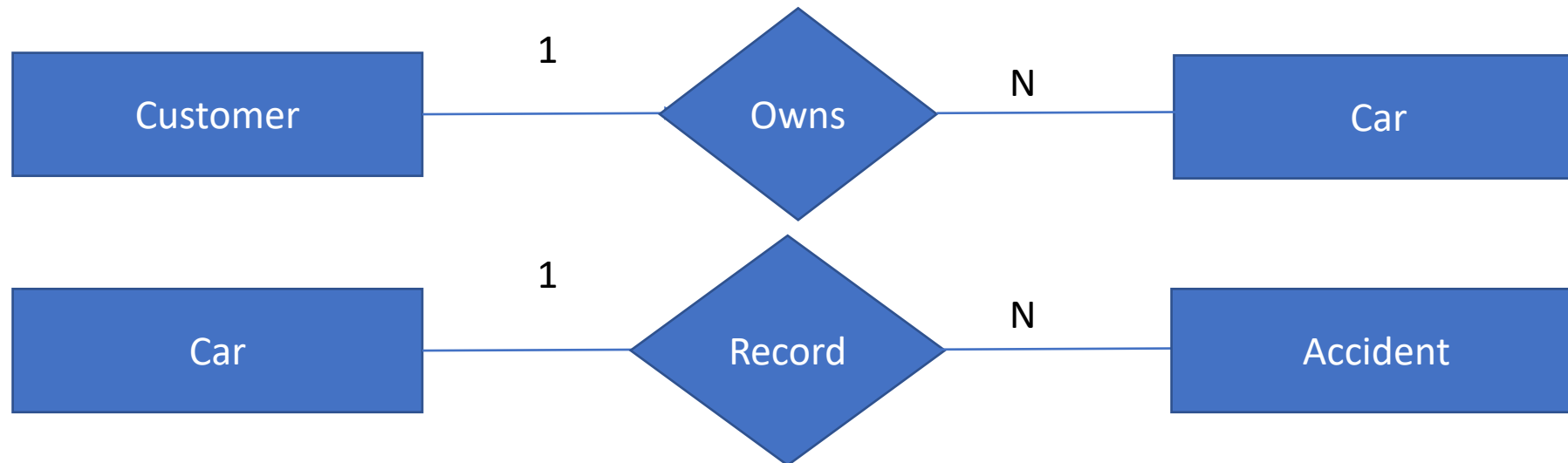
## **Entities:**

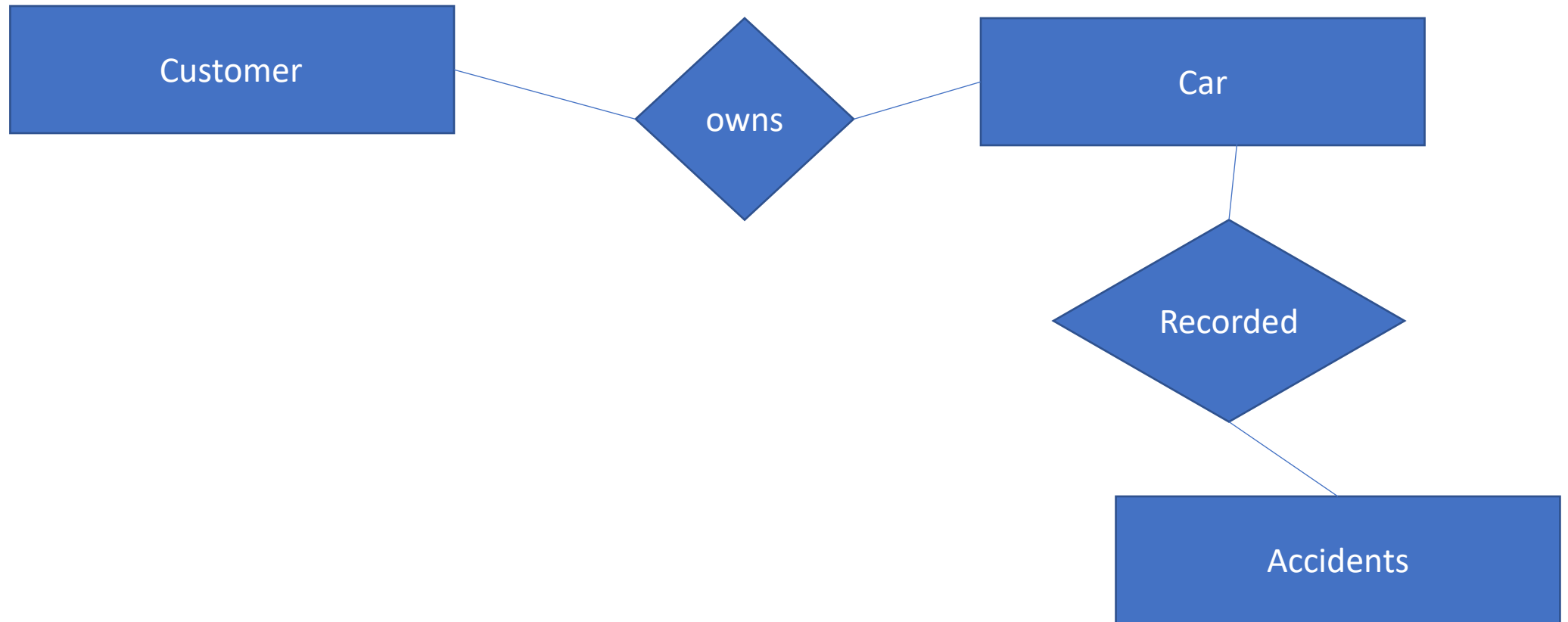
- Customer
- Car
- Accidents

## **Attributes:**

- Customer: Customer ID, Customer\_name
- Car : Car number, Car\_model, Owner\_name
- Accidents: Car, Acc ID

- **Relationship**

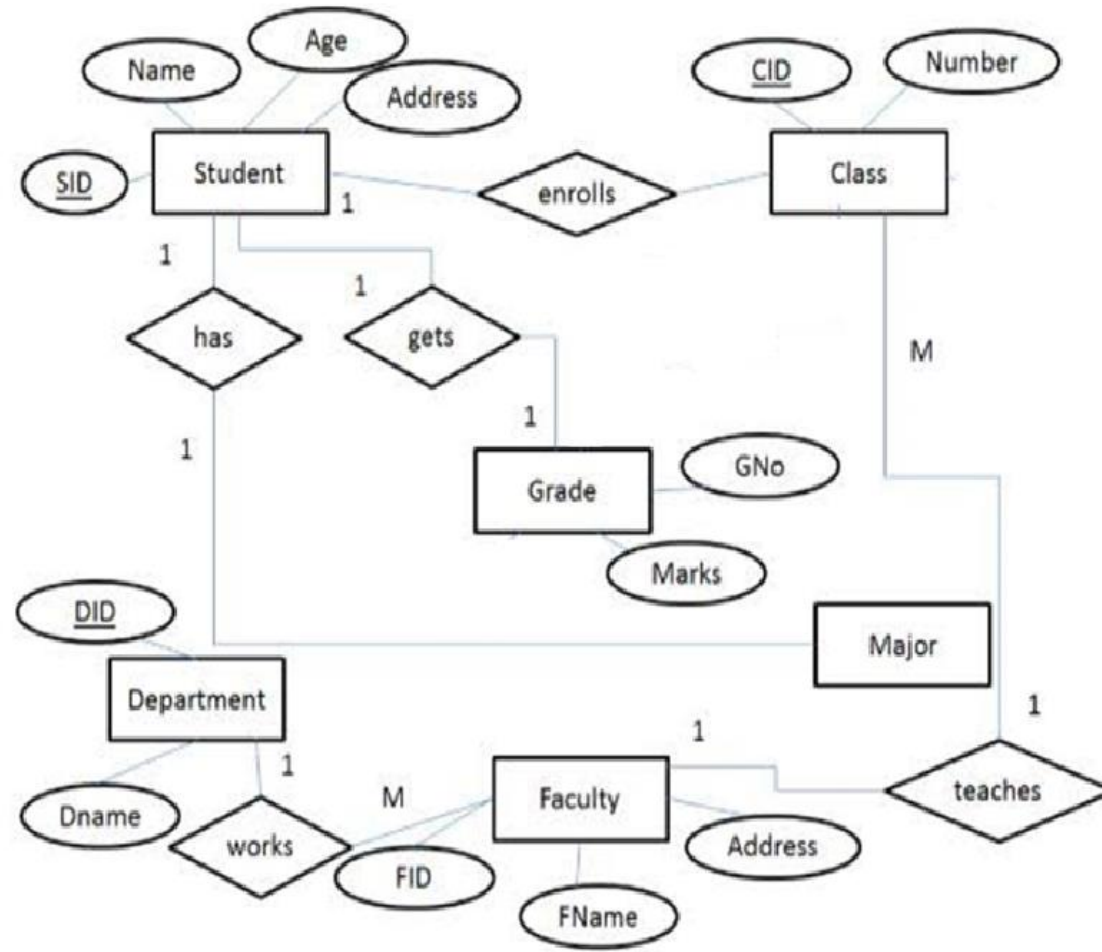




**ER Model for Insurance Company**

Draw ER Diagram for university database

- Student has a unique ID
- Student can enroll for multiple classes and has at most one major
- Faculty must belong to department
- Faculty can teach multiple classes
- Every student will get grade for the class he/she has enrolled.



# Extended E-R Features

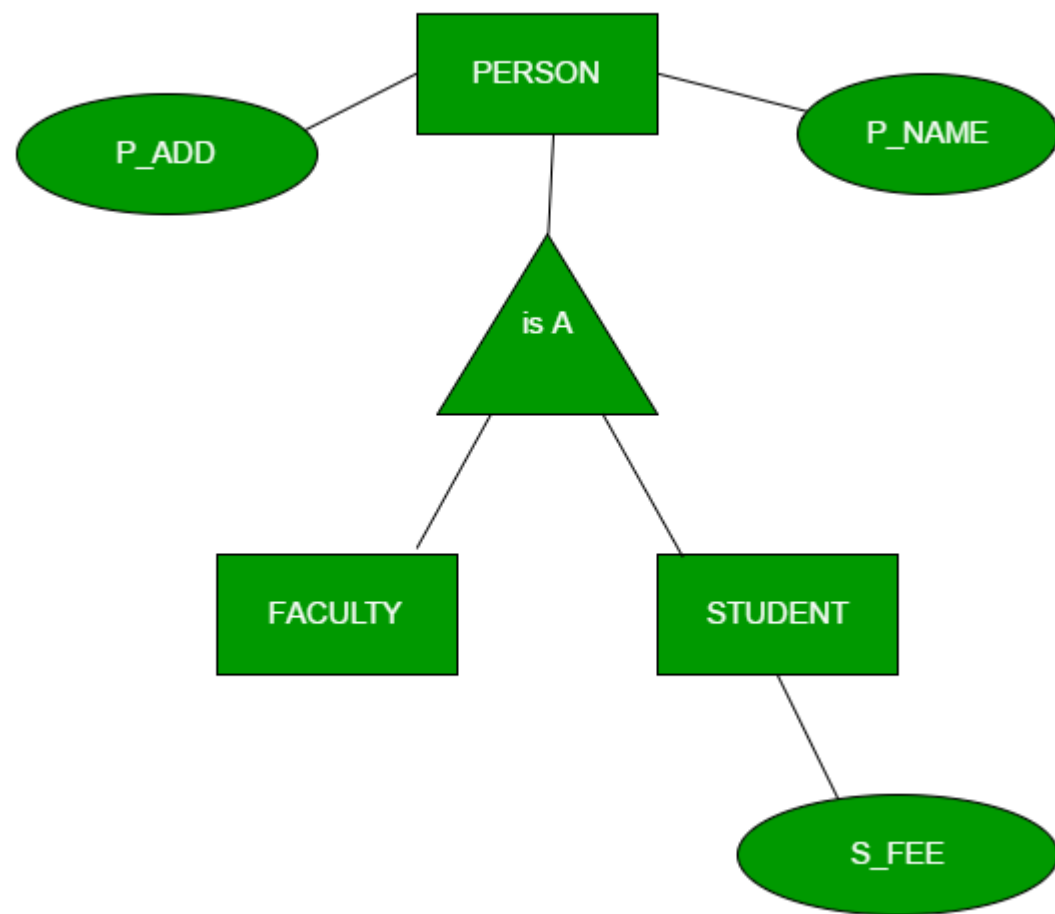


# Specialization

- Top-down design process; we designate sub-groupings within an entity set that are distinctive from other entities in the set.
- These sub-groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* component labeled ISA (e.g., *instructor* “is a” *person*).
- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

# Generalization

- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.

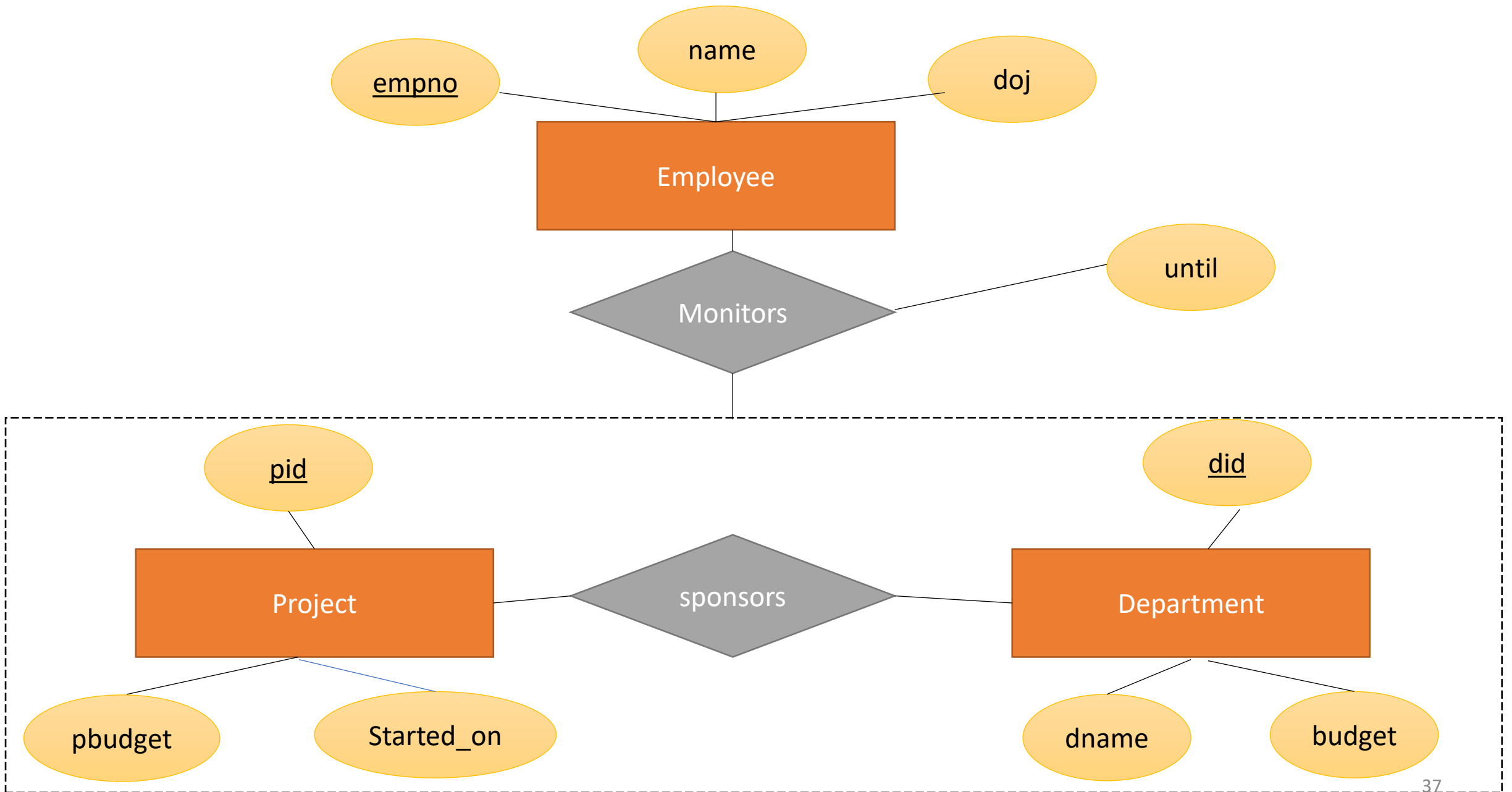


## AGGREGATION

- Aggregation is meant to represent a relationship between a whole object and its component parts.
- It is used when we have to model a relationship involving entity sets and a relationship set.
- It allows us to treat a relationship set as an entity set for the purpose of participation in (other) relationships.

### Example:

- A project is sponsored by a department. This is a simple relation.
- An employee monitors this Sponsorship ( and not project or department)  
This is called as aggregation.



# Reduction to Relation Schemas

- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of schemas.
- For each entity set there is a schema that is assigned the name of the corresponding entity set.
- Each schema has a number of columns (generally corresponding to attributes), which have unique names.

# Representing Entity Sets

- A strong entity set reduces to a schema with the same attributes

*student(ID, name, tot\_cred)*

- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set

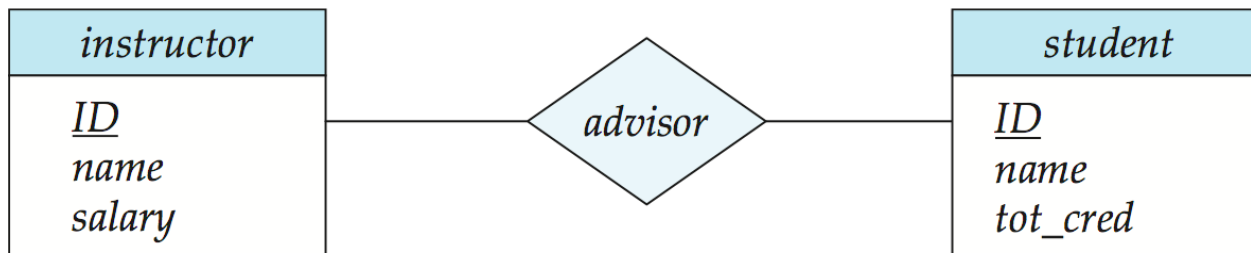
*section ( course\_id, sec\_id, sem, year )*



# Representing Relationship Sets

- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

*advisor* = (*s\_id*, *i\_id*)





- Composite attributes are flattened out by creating a separate attribute for each component attribute

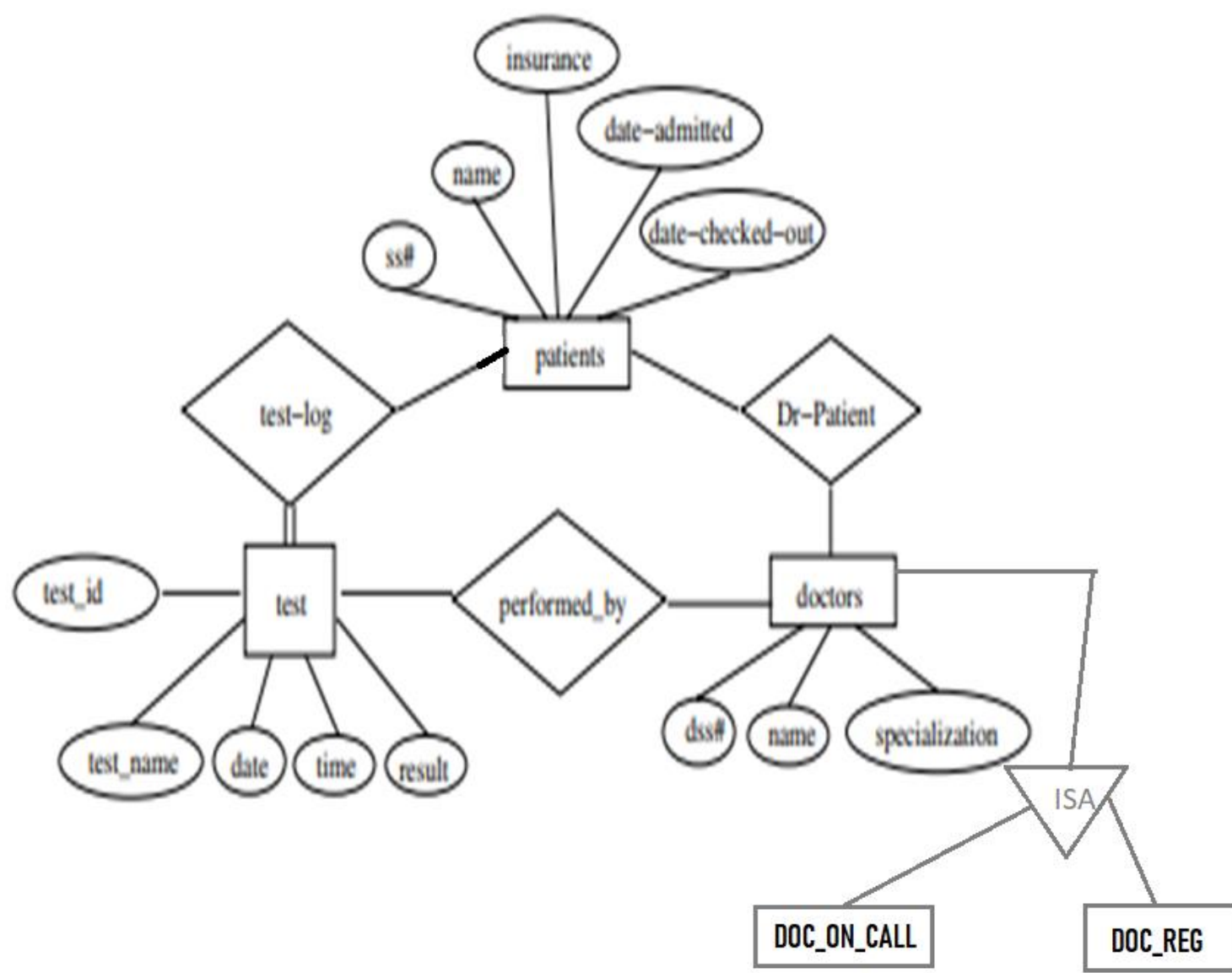
Example: given entity set *instructor* with composite attribute *name* with component attributes *first\_name* and *last\_name* the schema corresponding to the entity set has two attributes *name\_first\_name* and *name\_last\_name*

*instructor*(*ID*,  
    *first\_name*, *middle\_initial*, *last\_name*,  
    *street\_number*, *street\_name*,  
        *apt\_number*, *city*, *state*, *zip\_code*,  
    *date\_of\_birth*)

- A **multivalued attribute**  $M$  of an entity  $E$  is represented by a separate schema  $EM$
- Schema  $EM$  has attributes corresponding to the primary key of  $E$  and an attribute corresponding to multivalued attribute  $M$
- Example: Multivalued attribute *phone\_number* of *instructor* is represented by a schema:  
 $inst\_phone = ( \underline{ID}, \underline{phone\_number} )$
- Each value of the multivalued attribute maps to a separate tuple of the relation on schema  $EM$ 
  - For example, an *instructor* entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples:  
(22222, 456-7890) and (22222, 123-4567)

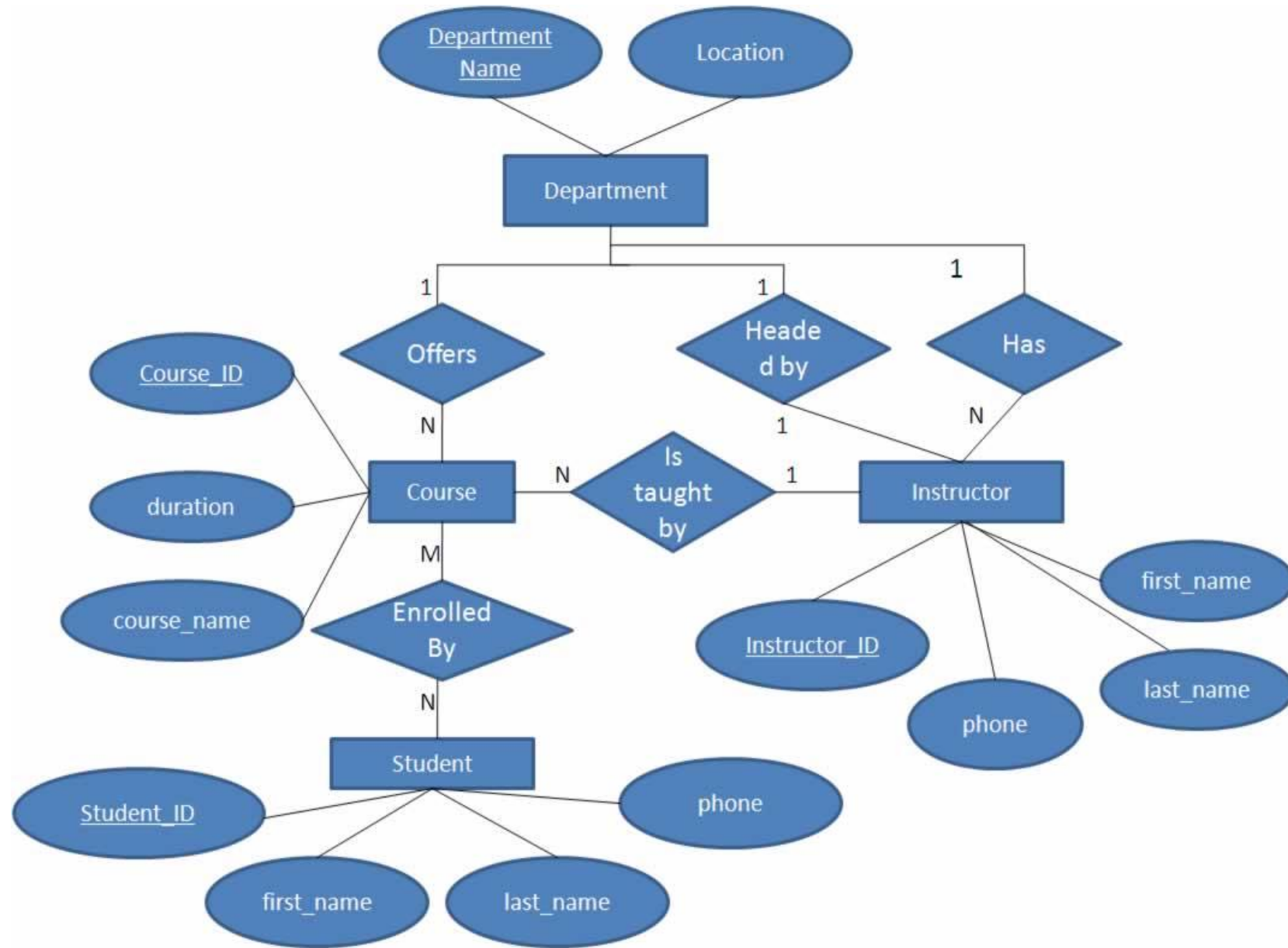
# PRACTICE QUESTIONS

Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. There are two types of doctor regular doctor and on all doctor. Associate with each patient a log of the various tests conducted.



# COLLEGE DATABASE

- A college contains many departments
- Each department can offer any number of courses
- Many instructors can work in a department
- An instructor can work only in one department
- For each department there is a Head
- An instructor can be head of only one department
- Each instructor can take any number of courses
- A course can be taken by only one instructor
- A student can enroll for any number of courses
- Each course can have any number of students

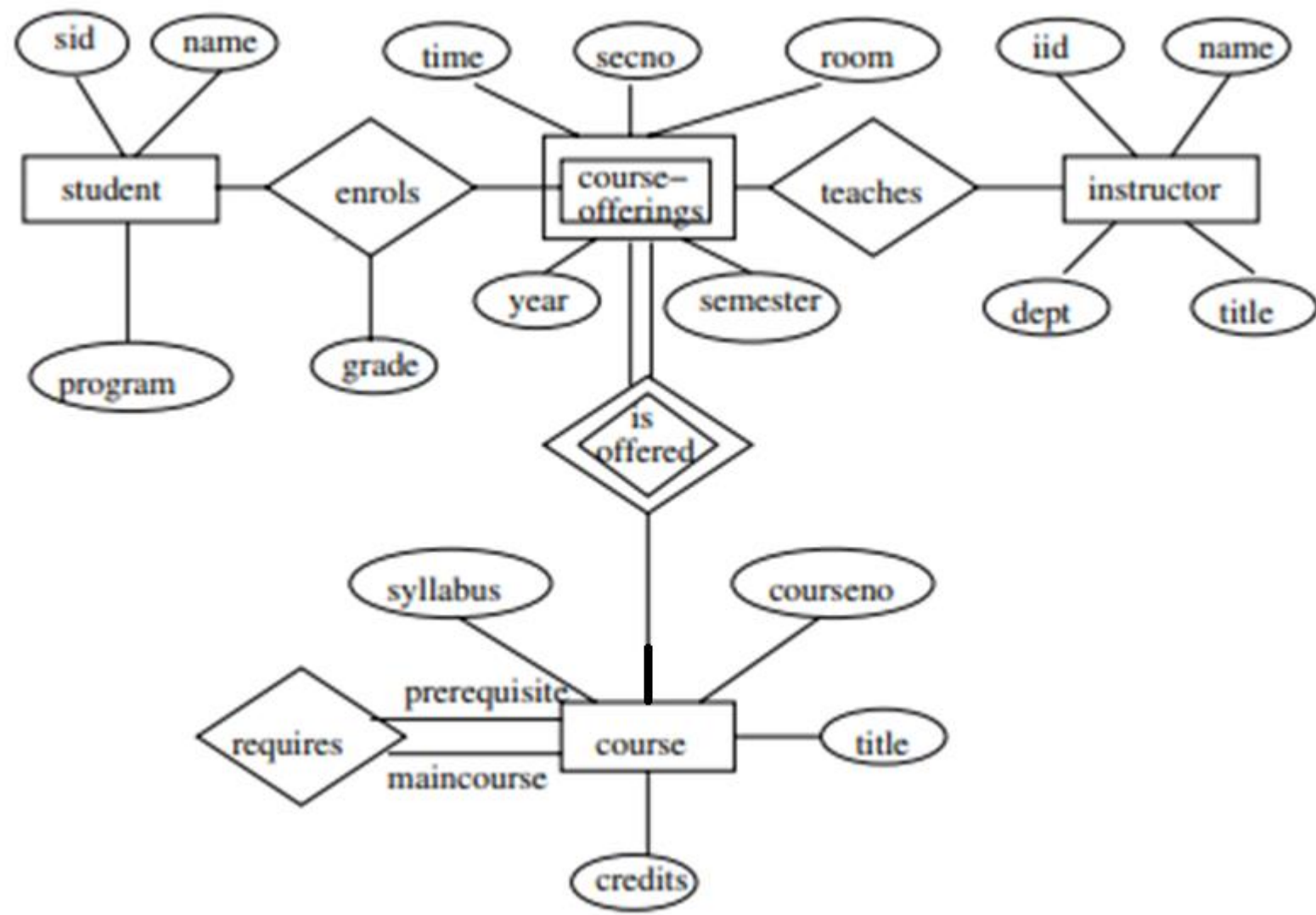


A university registrar's office maintains data about the following entities:

- (a) courses, including number, title, credits, syllabus, and prerequisites;
- (b) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom
- (c) students, including student-id, name, and program
- (d) instructors, including identification number, name, department, and title.
- (e) The enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled.

Construct an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints.





# The Flight Database

The flight database stores details about an airline's fleet, flights, and seat bookings.

Consider the following requirements list:

- The airline has one or more airplanes.
- An airplane has a model number, a unique registration number, and the capacity to take one or more passengers.
- An airplane flight has a unique flight number, a departure airport, a destination airport, a departure date and time, and an arrival date and time.
- One flight can have multiple bookings.
- Each booking details will store the ticket Number, fare
- Each flight is carried out by a single airplane.
- A passenger has given names, a surname, and a unique email address.
- A passenger can have multiple bookings.

# BOOK CLUB DATABASE

- The book club maintain various types of books.
- Club has members to whom books are sold.
- Books are made available in different places in the city called book club chapter for convenience of the members.
- Books are identified by ISBN, author and publisher.
- An author can write more than one book and a book can have more than one author.
- Members have information such as members\_id, name, phone\_number, status (Permanent or Temporary).
- Member can place more than one order
- For members club keeps track of their membership\_date, status, along with books he have issued.
- For each book it is necessary to keep track of the issue\_date and return\_date.

Draw an ER Diagram for the above given scenario.

# Relational Database Model

- Basic Concept of RDBMS (Relation, Attribute, Tuple , Domain)
- Codd's Rule
- Concept of keys in RDBMS
- Database Integrity Constraints (Domain Integrity, Entity & Referential Integrity)
- Database Schema Diagram

# Concept of Relation

- Relational Model was first proposed by E.F. Codd
- Relational database was an attempt to simplify database structure by making use of tables and columns.
- A relational database is a collection of 2 dimensional tables which consists of rows and columns.
- Tables are known as “**RELATION**”, Columns are known as “**ATTRIBUTES**” and rows are known as “**TUPLES**”.
- A relational database consists of a collection of tables having a unique name.
- A row in a table represents a relationship among a set of values. Thus a table represents a collection of relationships.

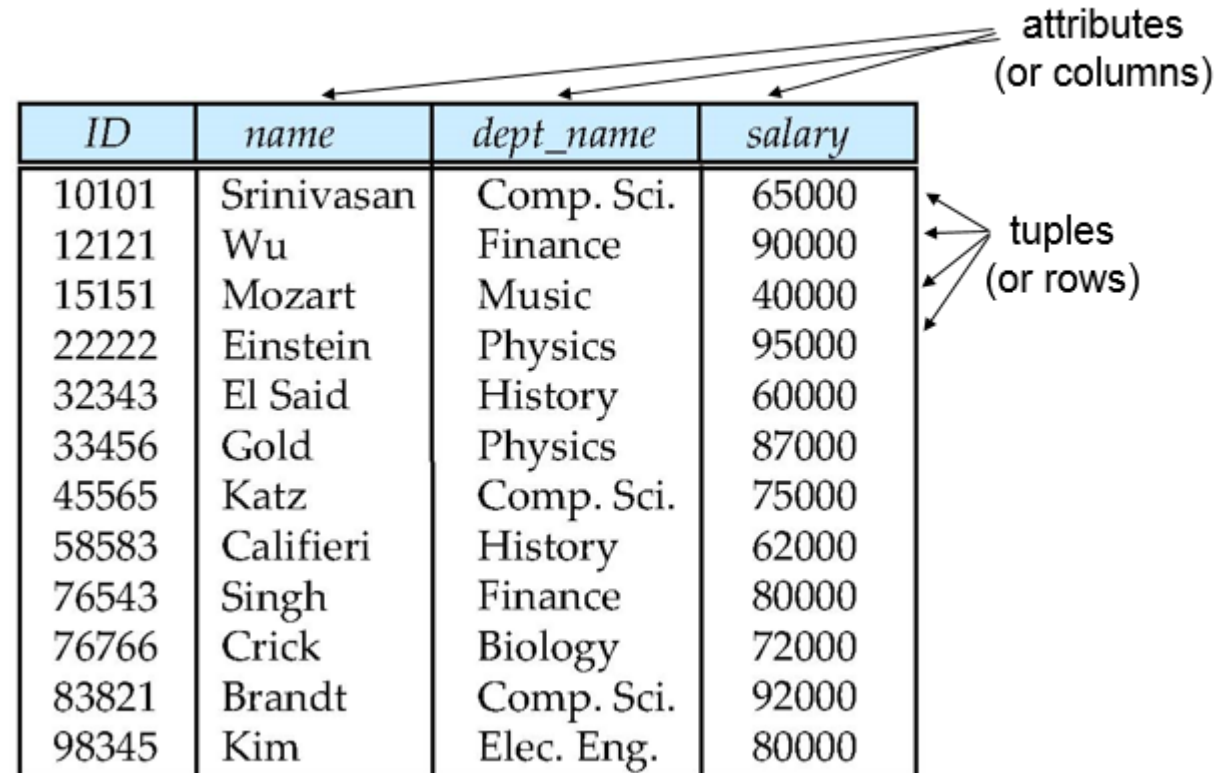
# Characteristics of a RELATION

- A table comprises of rows and columns.
- Each table row represents a single entity occurrence within the entity set.
- Each table column represents an attribute and each column has distinct name.
- All the values in a same column must conform to the same format of data.
- Each table must have a single attribute or set of attributes that uniquely identifies each row.
- A **domain** is defined as the set of all unique values permitted for an attribute.

# ATTRIBUTES

- Every relation must have at least one column in it.
- It is not possible to have two columns with same name in one relation while it is possible to have two columns with same column name in two different relations.
- The SQL standard does not specify any maximum number of columns in a relation.

# EXAMPLE OF A RELATION



The diagram illustrates a relation as a table. The table has four columns: *ID*, *name*, *dept\_name*, and *salary*. The first row of data is (10101, Srinivasan, Comp. Sci., 65000). There are 12 rows in total. Annotations with arrows point to the columns from the text 'attributes (or columns)' and to the rows from the text 'tuples (or rows)'.

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000



# CONCEPT OF KEYS

# KEYS

- Definition

- The column value that uniquely identifies a single tuple in a relational table is called as “KEY”.
- An attribute or a set of attributes whose values can uniquely identify single entity in an entity set is called a key for that entity set.
- They allow you to find the relation between two tables

- Types

- Any key consisting of single attribute is called a simple key.
- Any key with a set of attributes is called a composite key.

# Why we need a Key?

Here, are reasons for using Keys in the DBMS system.

- Keys help you to identify any row of data in a table. In a real-world application, a table could contain thousands of records. Moreover, the records could be duplicated. Keys ensure that you can uniquely identify a table record despite these challenges.
- Allows you to establish a relationship between and identify the relation between tables
- Help you to enforce identity and integrity in the relationship.

# KEY TYPE

- SUPER KEY

- Set of any attributes which uniquely identifies a single record in a table. Such keys are called super keys.

STUDENT TABLE

ID	NAME	CLASS	BRANCH	AGE	ADDRESS	ROLL_NO
----	------	-------	--------	-----	---------	---------

Example:

ID + ROLL\_NO attribute also a super key.

- **CANDIDATE KEY**

- Minimum attributes of super key by omitting unnecessary attributes of table which are sufficient for identifying entity (row/record) uniquely are called as candidate keys.
- Also known as minimal super key.

ID	NAME	CLASS	BRANCH	AGE	ADDRESS	ROLL_NO
----	------	-------	--------	-----	---------	---------

ID and Roll\_No individually are candidate keys

- PRIMARY KEY

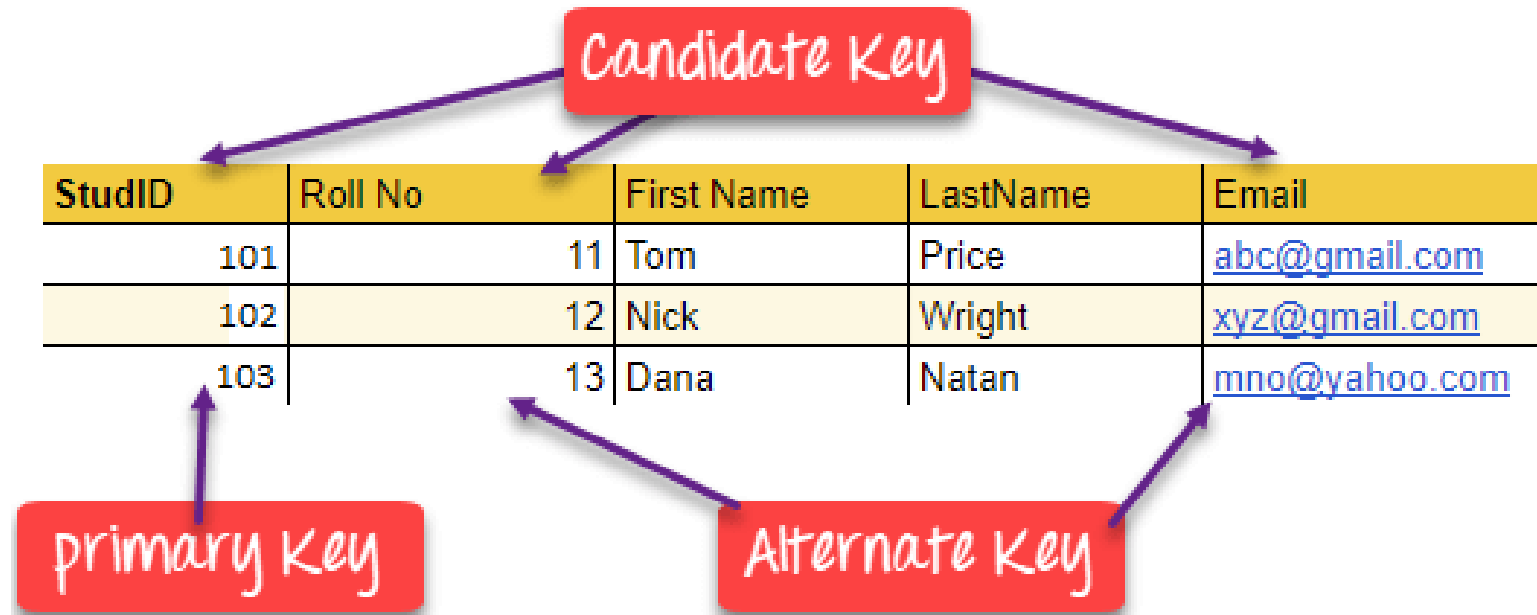
- Primary key of a table is a column or combination of some columns whose values uniquely identify a single row or tuple in that table.
- It is a unique identifier for the table (A column or a column combination with the property that at any given time no two rows of the table contains the same value in that column or column combination).

ID	NAME	CLASS	BRANCH	AGE	ADDRESS	ROLL_NO
----	------	-------	--------	-----	---------	---------

Example: Since only “ID” and “Roll\_No” are the Candidate keys , Any one becomes the primary key of that table depending on the database user.

- **Alternate Key**

- All the keys which are not primary key are called an alternate key. It is a candidate key which is currently not the primary key. However, A table may have single or multiple choices for the primary key.





- **FOREIGN KEY**

- A column or a collection of columns in one table whose value must matches the primary key in some other table is called foreign key.
- Likewise primary key foreign key can be combination of multiple columns.
- Foreign key can either have a matching primary key value in other table or it can be NULL.

ID	NAME	CLASS	BRANCH	AGE	ADDRESS
----	------	-------	--------	-----	---------

CLASS	DESCRIPTION	LOACTION
-------	-------------	----------

- Foreign key of one table should be PRIMARY KEY of other table.

- The maximum number of Super Keys for the relation schema  $R(E,F,G,H)$  with E as the key is \_\_\_\_\_ .

Given the STUDENTS relation as shown below.

<i>StudentID</i>	<i>StudentName</i>	<i>StudentEmail</i>	<i>StudentAge</i>	<i>CPI</i>
2345	Shankar	shankar@math	X	9.4
1287	Swati	swati@ee	19	9.5
7853	Shankar	shankar@cse	19	9.4
9876	Swati	swati@mech	18	9.3
8765	Ganesh	ganesh@civil	19	8.7

For (StudentName, StudentAge) to be the key for this instance, the value X should not be equal to

# Database Integrity Constraints

# DATA INTEGRITY

- **DOMAIN INTEGRITY**

- ☐ Domain restricts the values of attributes in the relation and is a constraint of the relational model.
- ☐ However, there are real-world semantics for data that cannot be specified if used only with domain constraints.
- ☐ We need more specific ways to state what data values are or are not allowed and which format is suitable for an attribute.
- ☐ For example, the Employee ID (EID) must be unique or the employee Birthdate is in the range [Jan 1, 1950, Jan 1, 2000]. Such information is provided in logical statements called integrity constraints.

- **ENTITY INTEGRITY**

- ☐ Every entity is uniquely identified by a unique not-null attribute or the primary key.
- ☐ **Primary key** : Primary key attribute values needs to be unique as well as null values are not allowed in primary key attributes

- **REFERENTIAL INTEGRITY**

- ☐ To show the relation into two tables, we use foreign key constraint.
- ☐ Referential integrity in a relational database is consistency between two tables, enforced by the combination of a primary key and a foreign key.
- ☐ Foreign key : A column or collection of columns in one table whose value must match the primary key in some other table is called “foreign key”.

# Other Constraints

- NOT NULL: As per requirements there are some values which should not be having NULL values.
- UNIQUE: In this case no two tuples can have equal value for the same attribute.
- CHECK: We can define own integrity rule using CHECK constraint.
- DEFAULT: We can set default values for those attributes which is null.

# NOT NULL

Microsoft Office Home | Mail - Ishani Saha - Outlook | Svkm's NMIMS deemed to be Un | G

127.0.0.1:8080/apex/f?p=4500:1003:3123589991619769::NO::

Apps | BTI Mentor Mentee... | Log In | Svkm's NMIMS dee... | B(Tech) Computer-... | BTech Final Year Pr...

## ORACLE® Application Express

Home | Application Builder | SQL Workshop | Team Development | Administration

Home > SQL Workshop > SQL Commands

☒ Autocommit Rows: 10 Save Run

```
Create table STUDENT
(
  SNO NUMBER(5) PRIMARY KEY,
  NAME VARCHAR(10) NOT NULL,
  CITY VARCHAR(10)
);
DESC STUDENT|
```

Results Explain Describe Saved SQL History

Object Type TABLE Object STUDENT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STUDENT	SNO	NUMBER	-	5	0	1	-	-	-
	NAME	VARCHAR2	10	-	-	-	-	-	-
	CITY	VARCHAR2	10	-	-	-	✓	-	-

1 - 3

Microsoft Office Home | Mail - Ishani Saha - Outlook | Sv

127.0.0.1:8080/apex/f?p=4500:1003:3123589991619769::NO::

Apps | BTI Mentor Mentee... | Log In | Svkm's NMIMS dee... | B(Tech) |

## ORACLE® Application Express

Home | Application Builder | SQL Workshop | Team Developm

Home > SQL Workshop > SQL Commands

☒ Autocommit Rows: 10 Save Run

```
Create table STUDENT
(
  SNO NUMBER(5) PRIMARY KEY,
  NAME VARCHAR(10) NOT NULL,
  CITY VARCHAR(10)
);
DESC STUDENT
ALTER TABLE STUDENT
MODIFY CITY NOT NULL
```

Results Explain Describe Saved SQL History

Table altered.

0.86 seconds

# UNIQUE

Microsoft Office Home | Mail - Ishani Saha - Outlook | Svkm's NMIMS deemed to be Un

127.0.0.1:8080/apex/f?p=4500:1003:3123589991619769::NO::

Apps | BTI Mentor Mentee... | Log In | Svkm's NMIMS dee... | B(Tech) Computer-... | BTech Final Yea

## ORACLE® Application Express

Home | Application Builder | SQL Workshop | Team Development | Administration

Home > SQL Workshop > SQL Commands

☒ Autocommit Rows: 10 Save Run

```
Create table STUDENT
(
  SNO NUMBER(5) PRIMARY KEY,
  ROLL_NO NUMBER(10),
  NAME VARCHAR(10) UNIQUE,
  CITY VARCHAR(10)
);
```

DESC STUDENT

Results Explain Describe Saved SQL History

Object Type	TABLE	Object	STUDENT						
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STUDENT	SNO	NUMBER	-	5	0	1	-	-	-
	ROLL_NO	NUMBER	-	10	0	-	✓	-	-
	NAME	VARCHAR2	10	-	-	-	✓	-	-
	CITY	VARCHAR2	10	-	-	-	✓	-	-

1 - 4

Microsoft Office Home | Mail - Ishani Saha - Outlook | Svkm's NMIMS deemed to be Un

127.0.0.1:8080/apex/f?p=4500:1003:3123589991619769::NO::

Apps | BTI Mentor Mentee... | Log In | Svkm's NMIMS dee... | B(Te

## ORACLE® Application Express

Home | Application Builder | SQL Workshop | Team Development

Home > SQL Workshop > SQL Commands

☒ Autocommit Rows: 10 Save Run

```
Create table STUDENT
(
  SNO NUMBER(5) PRIMARY KEY,
  ROLL_NO NUMBER(10),
  NAME VARCHAR(10) UNIQUE,
  CITY VARCHAR(10)
);
```

DESC STUDENT

```
ALTER TABLE STUDENT
MODIFY ROLL_NO UNIQUE
```

Results Explain Describe Saved SQL History

Table altered.

0.02 seconds



# CHECK

Microsoft Office Home | Mail - Ishani Saha - Outlook | Svkm's NMIMS deemed to be Un...

127.0.0.1:8080/apex/f?p=4500:1003:3123589991619769::NO::

Apps | BTI Mentor Mentee... | Log In | Svkm's NMIMS dee... | B(Tech) Computer-... | BTech Final Year Pr...

## ORACLE® Application Express

Home | Application Builder | SQL Workshop | Team Development | Administration

Home > SQL Workshop > SQL Commands

☒ Autocommit Rows: 10 Save Run

```
Create table STUDENT
(
  SNO NUMBER(5) PRIMARY KEY,
  ROLL_NO NUMBER(10),
  NAME VARCHAR(10),
  CITY VARCHAR(10),
  MARKS NUMBER(5) CHECK (MARKS>50)
);

DESC STUDENT
```

Results Explain Describe Saved SQL History

Object Type TABLE Object STUDENT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STUDENT	SNO	NUMBER	-	5	0	1	-	-	-
	ROLL_NO	NUMBER	-	10	0	-	✓	-	-
	NAME	VARCHAR2	10	-	-	-	✓	-	-
	CITY	VARCHAR2	10	-	-	-	✓	-	-
	MARKS	NUMBER	-	5	0	-	✓	-	-

1 - 5

Microsoft Office Home | Mail - Ishani Saha - Outlook | Svkm's NMIMS d

127.0.0.1:8080/apex/f?p=4500:1003:3123589991619769::NO::

Apps | BTI Mentor Mentee... | Log In | Svkm's NMIMS dee... | B(Tech) Computer-...

## ORACLE® Application Express

Home | Application Builder | SQL Workshop | Team Development

Home > SQL Workshop > SQL Commands

☒ Autocommit Rows: 10 Save Run

```
Create table STUDENT
(
  SNO NUMBER(5) PRIMARY KEY,
  ROLL_NO NUMBER(10),
  NAME VARCHAR(10),
  CITY VARCHAR(10),
  MARKS NUMBER(5) CHECK (MARKS>50)
);

DESC STUDENT

ALTER TABLE STUDENT
MODIFY ROLL_NO CHECK (ROLL_NO BETWEEN 10 AND 50)
```

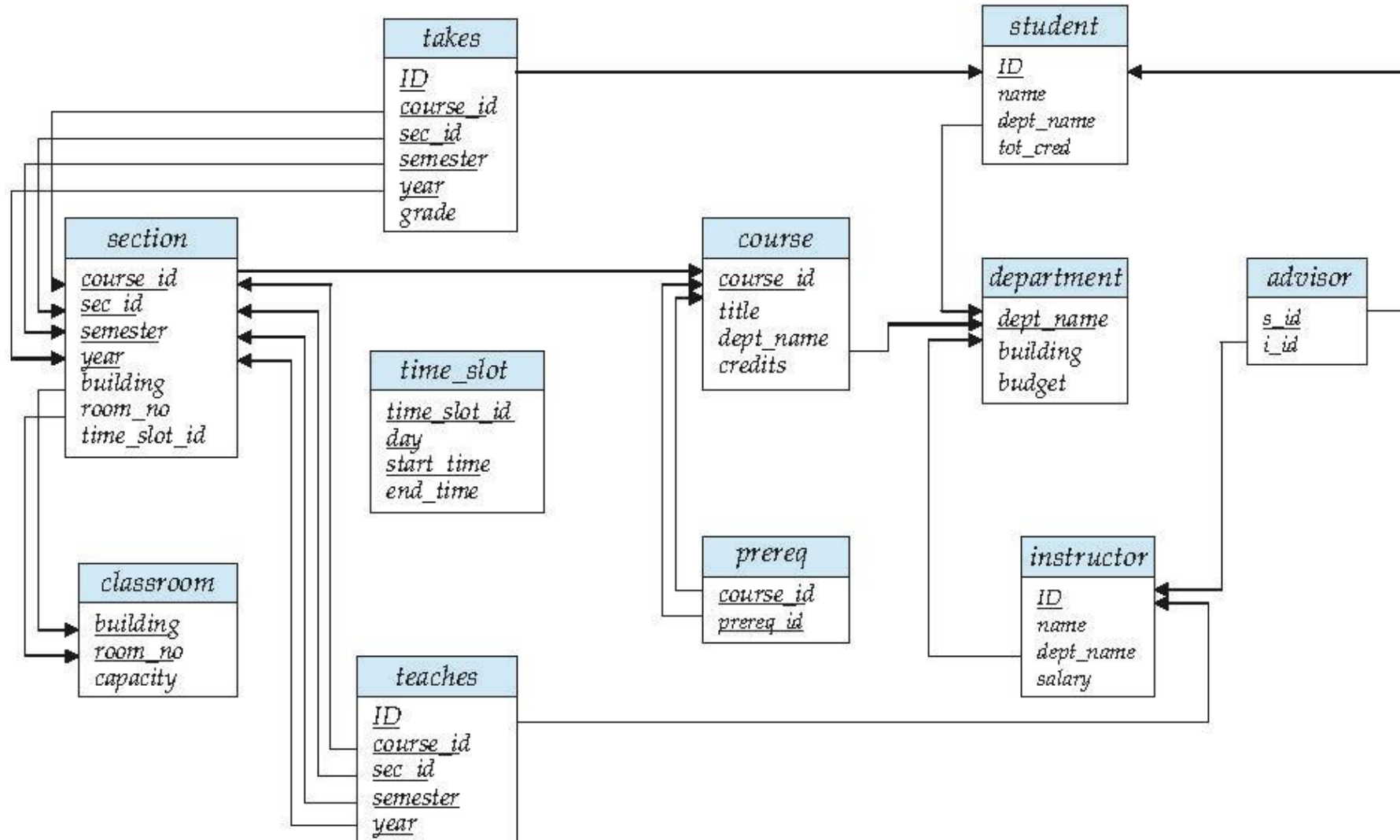
Results Explain Describe Saved SQL History

Table altered.

0.00 seconds



# Database Schema Diagram



- Let E1 and E2 be two entities in an E/R diagram with simple single-valued attributes. R1 and R2 are two relationships between E1 and E2, where R1 is one-to-many and R2 is many-to-many. R1 and R2 do not have any attributes of their own. What is the minimum number of tables required to represent this situation in the relational model?

- Which of the following is NOT a superkey in a relational schema with attributes V, W, X, Y, Z and primary key V Y ?

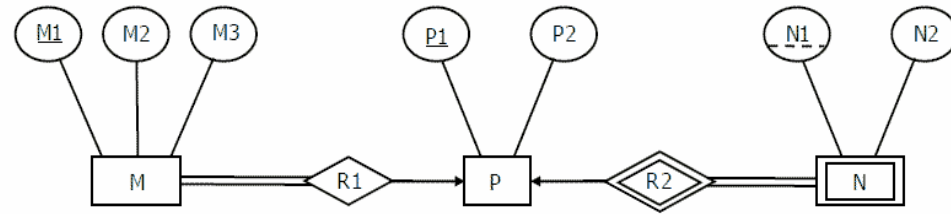
**(A)** V X Y Z

**(B)** V W X Z

**(C)** V W X Y

**(D)** V W X Y Z

Consider the following ER diagram. \_



The minimum number of tables needed