Database Management System

UNIT-I
INTRODUCTION
TO
DATABASE CONCEPTS

Mr. Naresh A. Kamble

COURSE OUTCOME (CO)

CO1 - Explain the fundamental database concepts.

POINTS TO BE COVERED

- **INTRODUCTION**
- Purpose of Database Systems
- View of Data
- Data Models
- Database Architecture
- Roles in Database Environment
- The Entity-Relationship Model
- Entity-Relationship Diagrams
- Reduction to Relational Schemas
- Introduction to Relational Model
- Relational Query Languages- The Relational Algebra

INTRODUCTION

- **WHAT IS DATABASE?**
- A database is an organized collection of ***structured information***, or ***data***, typically stored ***electronically*** in a computer system.
- A database is usually controlled by a ***database management system (DBMS).***
- Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

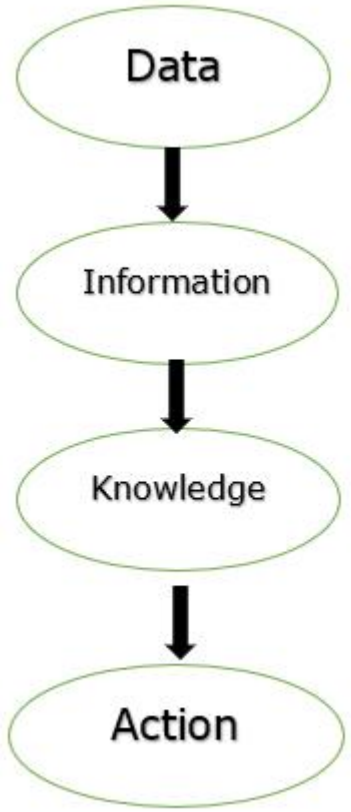
INTRODUCTION

- **WHAT IS DBMS?**
- A ***Database Management System (DBMS)*** is a software system that is designed to ***manage*** and ***organize data*** in a ***structured manner***.

- It allows users to ***create***, ***modify***, and ***query*** a ***database***, as well as manage the ***security*** and ***access controls*** for that database.

INTRODUCTION

- **PURPOSE OF DBMS?**
- The purpose of DBMS is to transform the following
- Data into information.
- Information into knowledge.
- Knowledge to the action.



INTRODUCTION

- **USE OF DBMS**
- The main uses of DBMS are as follows
- Data independence and efficient access of data.
- Application Development time reduces.
- Security and data integrity.
- Uniform data administration.
- Concurrent access and recovery from crashes.

INTRODUCTION

- **APPLICATIONS OF DBMS**
- The different applications of DBMS are as follows

- **Railway Reservation System**
- It is used to keep record of booking of tickets, departure of the train and the status of arrival and give updates to the passengers with the help of a database.

INTRODUCTION

- **APPLICATIONS OF DBMS**
- **Library Management System**
- There will be so many numbers of books in the library and it is very hard to keep a record of all the books in a register or a copy. So, DBMS is necessary to keep track of all the book records, issue dates, name of the books, author and maintain the records.

INTRODUCTION

- **APPLICATIONS OF DBMS**
- **Banking** − We are doing a lot of transactions daily without directly going to the banks. The only reason is the usage of databases and it manages all the data of the customers over the database.
- **Educational Institutions** − All the examinations and the data related to the students maintained over the internet with the help of a database management system. It contains registration details of the student, results, grades and courses available. All these works can be done online without visiting an institution.

INTRODUCTION

- **APPLICATIONS OF DBMS**
- **Social Media Websites**
- By filling the required details we are able to access social media platforms. Many users daily sign up for social websites such as Facebook, Pinterest and Instagram. All the information related to the users are stored and maintained with the help of DBMS.

INTRODUCTION

- **KEY FEATURES OF DBMS**
- **Data modeling:** A DBMS provides tools for creating and modifying data models, which define the structure and relationships of the data in a database.
- **Data storage and retrieval:** A DBMS is responsible for storing and retrieving data from the database, and can provide various methods for searching and querying the data.

INTRODUCTION

- **KEY FEATURES OF DBMS**
- **Concurrency control:** A DBMS provides mechanisms for controlling concurrent access to the database, to ensure that multiple users can access the data without conflicting with each other.

- **Data integrity and security:** A DBMS provides tools for enforcing data integrity and security constraints, such as constraints on the values of data and access controls that restrict who can access the data.

INTRODUCTION

- **KEY FEATURES OF DBMS**
- **Backup and recovery:** A DBMS provides mechanisms for backing up and recovering the data in the event of a system failure.

- **DBMS can be classified into two types:** Relational Database Management System (RDBMS) and Non-Relational Database Management System (NoSQL or Non-SQL)

INTRODUCTION

- **KEY FEATURES OF DBMS**
- **RDBMS:** Data is organized in the form of tables and each table has a set of rows and columns. The data is related to each other through primary and foreign keys.
- **NoSQL:** Data is organized in the form of key-value pairs, document, graph, or column-based. These are designed to handle large-scale, high-performance scenarios.

INTRODUCTION

- **TYPES OF DATA LANGUAGES**
- Data Definition Language (DDL)
- Data Manipulation Language(DML)
- Data Control Language(DCL)
- Transactional Control Language(TCL)

INTRODUCTION

- **TYPES OF DATA LANGUAGES**
- **Data Definition Language (DDL)**
- It deals with database schemas and descriptions, of how the data should reside in the database.
- **CREATE:** to create a database and its objects like (table, index, views, store procedure, function, and triggers)
- **ALTER:** alters the structure of the existing database
- **DROP:** delete objects from the database
- **TRUNCATE:** remove all records from a table, including all spaces allocated for the records are removed
- **COMMENT:** add comments to the data dictionary
- **RENAME:** rename an object

INTRODUCTION

- **TYPES OF DATA LANGUAGES**
- **Data Manipulation Language (DML)**
- DML is the short name for Data Manipulation Language which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.
- **SELECT:** retrieve data from a database
- **INSERT:** insert data into a table
- **UPDATE:** updates existing data within a table
- **DELETE:** Delete all records from a database table
- **MERGE:** UPSERT operation (insert or update)
- **CALL:** call a PL/SQL or Java subprogram
- **EXPLAIN PLAN:** interpretation of the data access path
- **LOCK TABLE:** concurrency Control

INTRODUCTION

- **TYPES OF DATA LANGUAGES**
- **Data Control Language(DCL)**
- DCL is short for Data Control Language which acts as an access specifier to the database.(basically to grant and revoke permissions to users in the database

- **GRANT:** grant permissions to the user for running DML(SELECT, INSERT, DELETE,…) commands on the table
- **REVOKE:** revoke permissions to the user for running DML(SELECT, INSERT, DELETE,…) command on the specified table

INTRODUCTION

- **TYPES OF DATA LANGUAGES**
- **Transactional Control Language(TCL)**
- TCL is short for Transactional Control Language which acts as an manager for all types of transactional data and all transactions.

- Some of the command of TCL are
- **Roll Back:** Used to cancel or Undo changes made in the database
- **Commit:** It is used to apply or save changes in the database
- **Save Point:** It is used to save the data on the temporary basis in the database

VIEW OF DATA

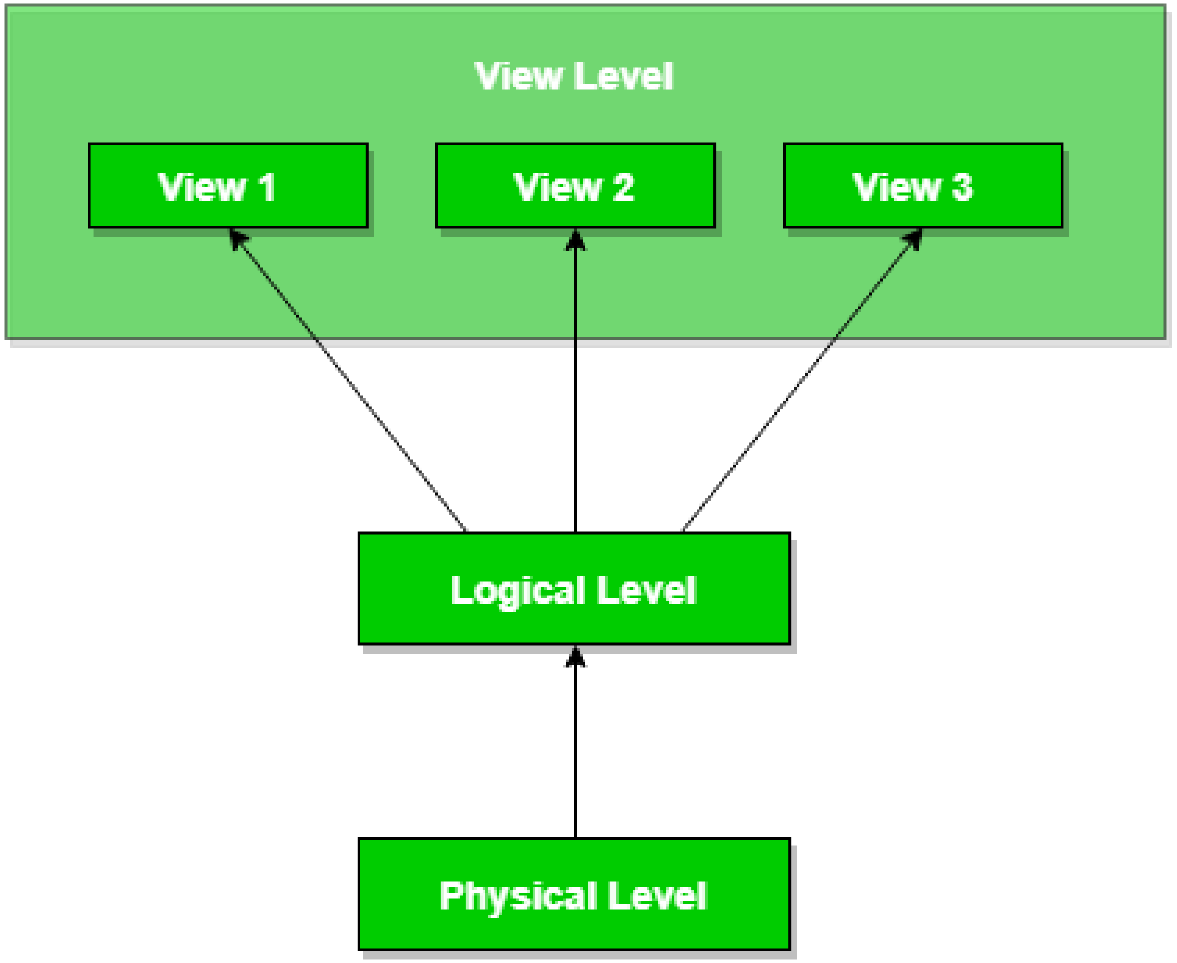
- Main purpose is an **ABSTRACT VIEW** of data which is called as **DATA ABSTRACTION**.

- ***Data Abstraction is a process of hiding unwanted or irrelevant details from the end user. It provides a different view and helps in achieving data independence which is used to enhance the security of data.***

- The database systems consist of complicated data structures and relations. For users to access the data easily, these complications are kept hidden, and only the relevant part of the database is made accessible to the users through data abstraction.

VIEW OF DATA

- There are mainly 3 levels of data abstraction:



VIEW OF DATA

- **L1 - PHYSICAL OR INTERNAL LEVEL**
- It is the lowest level of abstraction for DBMS which defines how the data is actually stored, it defines data-structures to store data and access methods used by the database. Actually, it is decided by developers or database application programmers how to store the data in the database.
- So, overall, the entire database is described in this level that is physical or internal level. It is a very complex level to understand.
- For example, customer's information is stored in tables and data is stored in the form of blocks of storage such as bytes, gigabytes etc.

VIEW OF DATA

- **L2 - LOGICAL OR CONCEPTUAL LEVEL**
- Logical level is the intermediate level or next higher level. It describes what data is stored in the database and what relationship exists among those data.
- It tries to describe the entire or whole data because it describes what tables to be created and what are the links among those tables that are created.
- It is less complex than the physical level. Logical level is used by developers or database administrators (DBA). So, overall, the logical level contains tables (fields and attributes) and relationships among table attributes.

VIEW OF DATA

- **L3 - VIEW OR EXTERNAL LEVEL**
- It is the highest level. In view level, there are different levels of views and every view only defines a part of the entire data. It also simplifies interaction with the user and it provides many views or multiple views of the same database.
- View level can be used by all users (all levels' users). This level is the least complex and easy to understand.
- For example, a user can interact with a system using GUI that is view level and can enter details at GUI or screen and the user does not know how data is stored and what data is stored, this detail is hidden from the user.

VIEW OF DATA

- The main purpose of data abstraction is to achieve ***data independence*** in order to save the ***time*** and ***cost*** required when the database is ***modified*** or ***altered***.

- ***Data Independence*** is mainly defined as a property of DBMS that helps you to change the ***database schema*** at one level of a system without requiring to change the ***schema*** at the next level.

- It helps to keep the data separated from all program that makes use of it.

VIEW OF DATA

- We have namely two levels of data independence arising from these levels of abstraction :
- **Physical level data independence:** It refers to the characteristic of being able to modify the ***physical schema*** without any ***alterations*** to the ***conceptual*** or ***logical schema***, done for optimization purposes.
- These alterations or modifications to the physical structure may include:
  – Utilizing new storage devices.
  – Modifying data structures used for storage.
  – Altering indexes or using alternative file organization techniques etc.

VIEW OF DATA

- **Logical level data independence:** It refers characteristic of being able to modify the logical schema without affecting the external schema or application program.

- The user view of the data would not be affected by any changes to the conceptual view of the data.

- These changes may include insertion or deletion of attributes, altering table structures entities or relationships to the logical schema, etc.

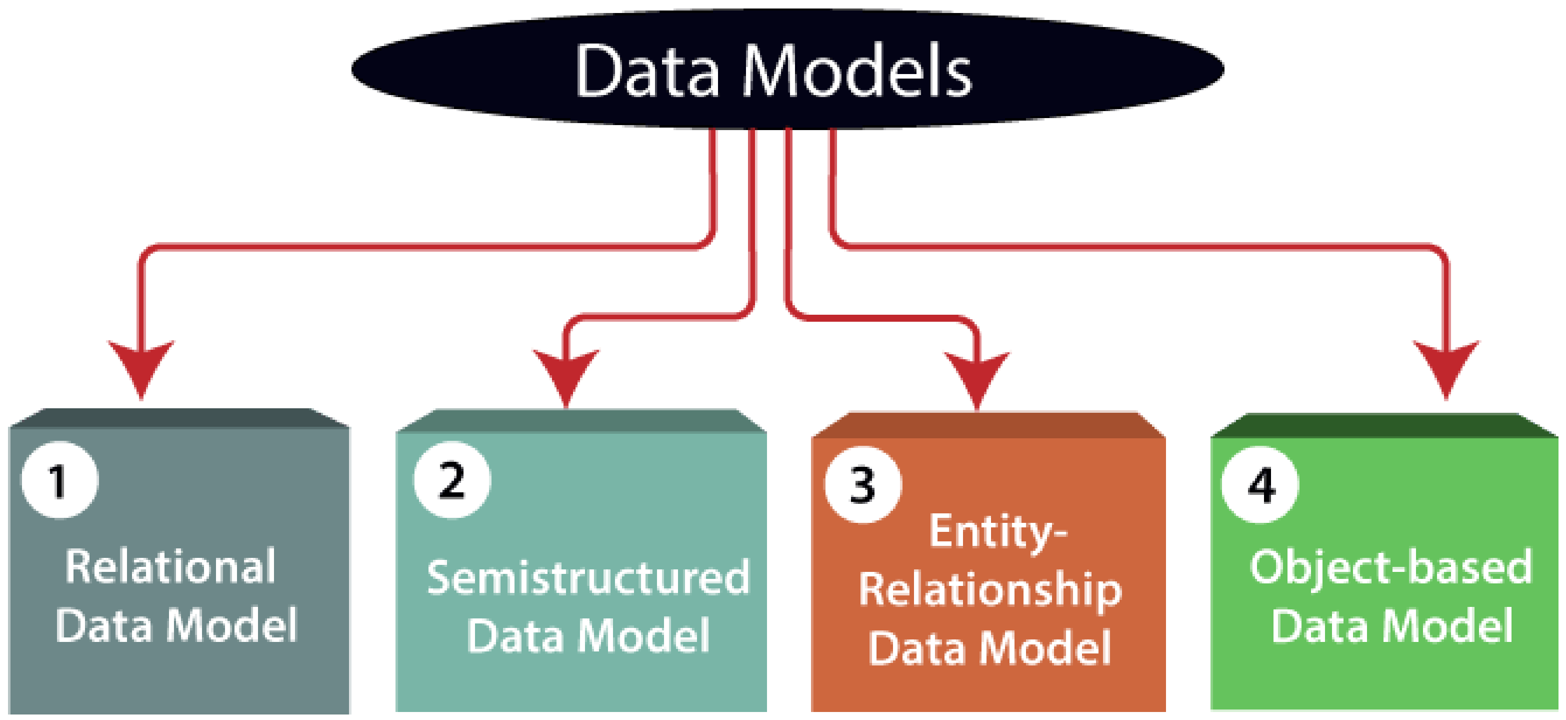
DATA MODELS

- Data Model is the modeling of the ***data description***, ***data semantics***, and ***consistency constraints*** of the data.

- It provides the conceptual tools for describing the design of a database at each level of data abstraction.

DATA MODELS

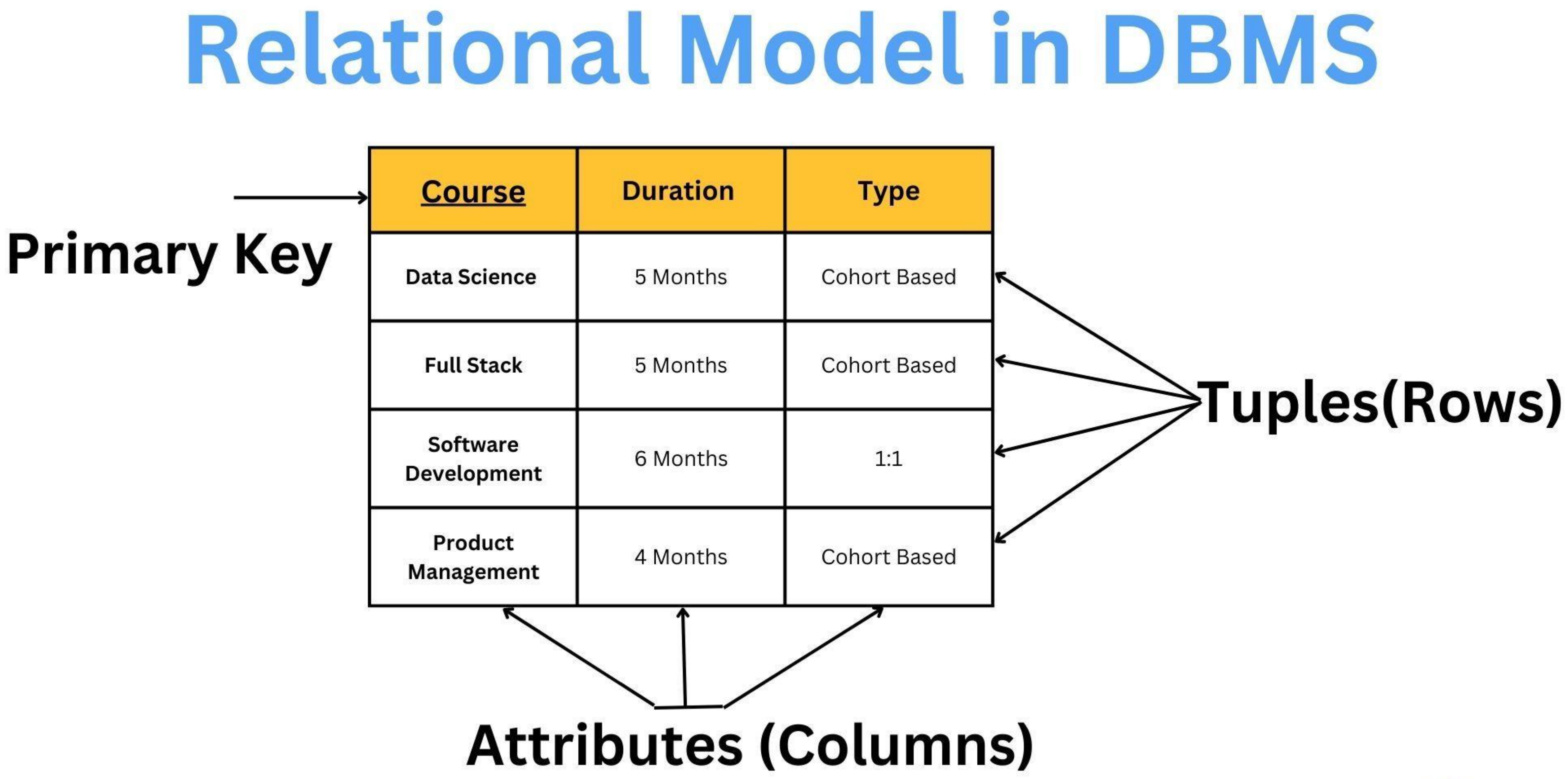
- There are following four data models used for understanding the structure of the database:


Data Models
1 Relational Data Model
2 Semistructured Data Model
3 Entity-Relationship Data Model
4 Object-based Data Model

DATA MODELS

- **RELATIONAL DATA MODEL**
- This type of model designs the data in the form of ***rows*** and ***columns*** within a table. Thus, a relational model uses tables for representing data and ***in-between relationships***.
- Tables are also called ***relations***.
- The relational data model is the widely used model which is primarily used by commercial data processing applications.

DATA MODELS

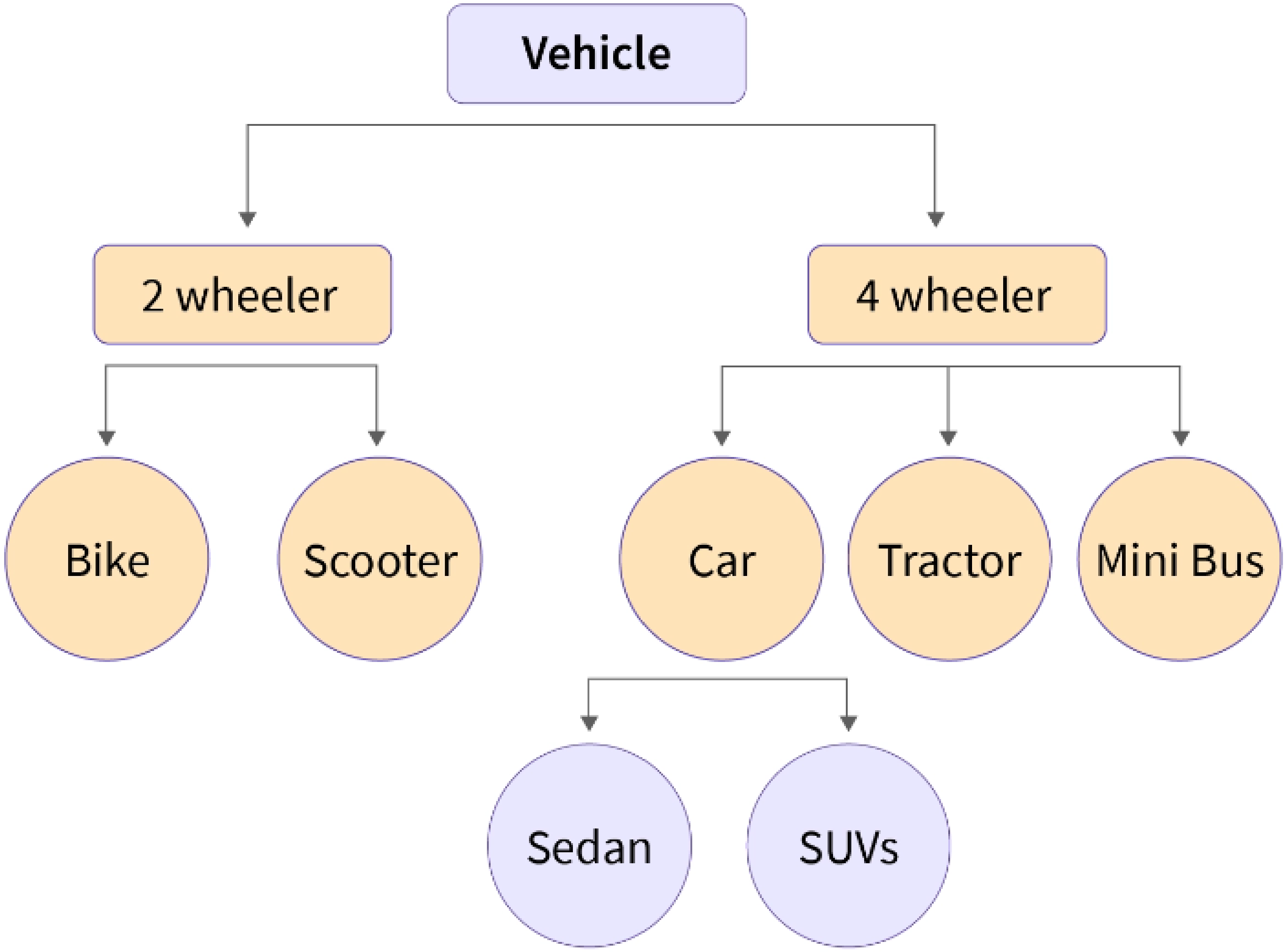


Course	Duration	Type
Data Science	5 Months	Cohort Based
Full Stack	5 Months	Cohort Based
Software Development	6 Months	1:1
Product Management	4 Months	Cohort Based

DATA MODELS

- **SEMISTRUCTURED DATA MODEL**
- The semistructured data model allows the ***data specifications*** at places where the ***individual data items*** of the ***same type*** may have ***different attributes*** sets.
- The ***Extensible Markup Language***, also known as ***XML***, is widely used for representing the semistructured data.

DATA MODELS

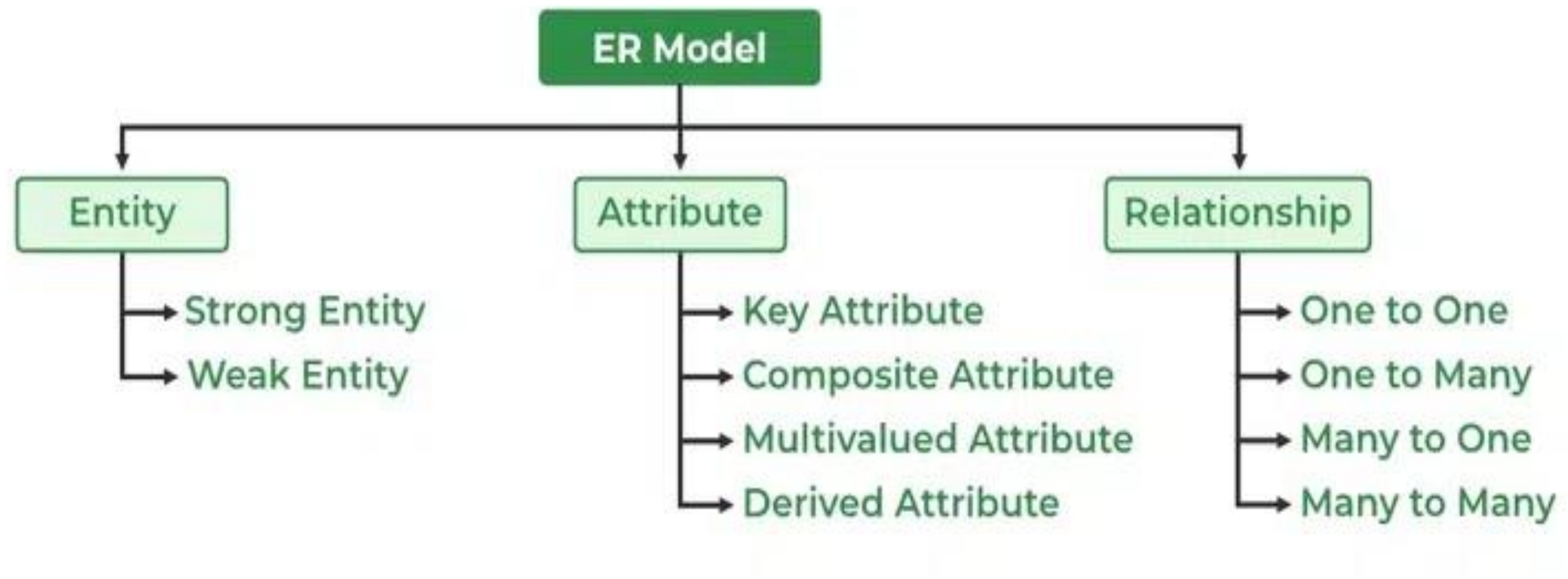

Vehicle
2 wheeler
4 wheeler
Bike
Scooter
Car
Tractor
Mini Bus
Sedan
SUVs

DATA MODELS

- **ENTITY-RELATIONSHIP DATA MODEL**
- An ER model is the logical representation of data as ***objects*** and ***relationships*** among them.
- These objects are known as ***entities***, and ***relationship*** is an ***association*** among these entities.
- A ***set of attributes*** describe the ***entities***. For example, student_name, student_id describes the 'student' entity.
- A set of the ***same type of entities*** is known as an '***Entity set***', and the ***set of the same type of relationships*** is known as '***relationship set***'.

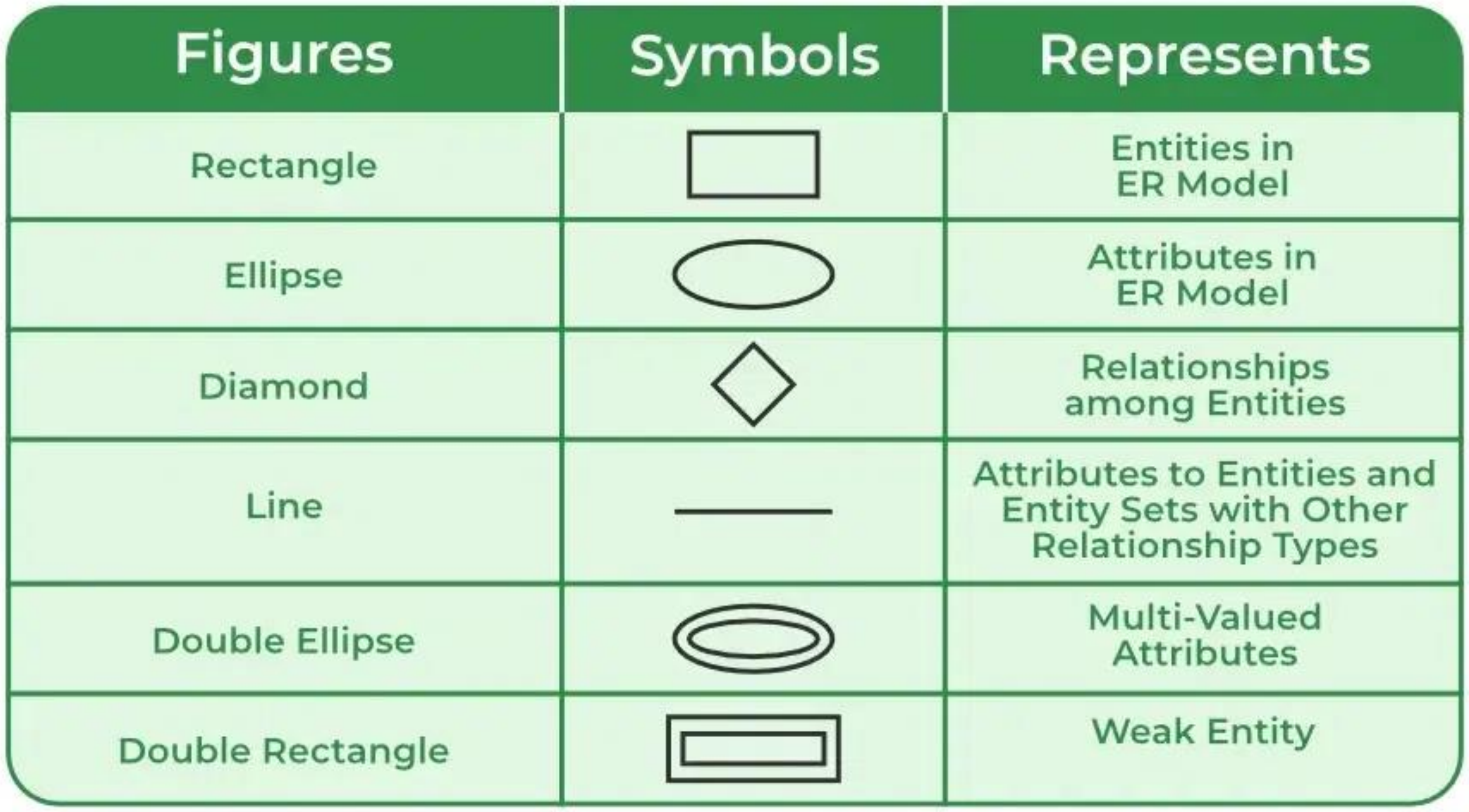
DATA MODELS

Different components of ER model



DATA MODELS

- The Geometric Shapes and their meaning in an E-R Diagram.

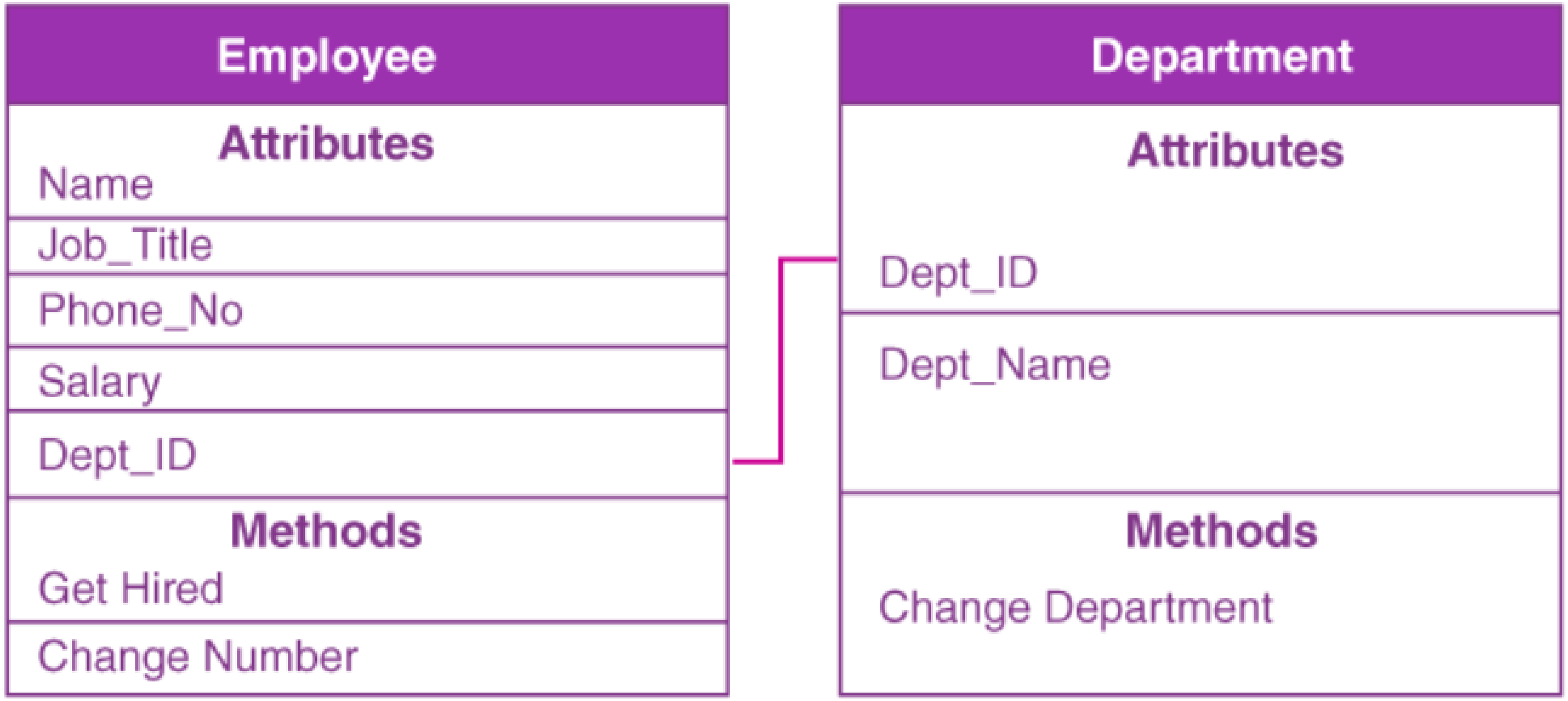


Figures	Symbols	Represents
Rectangle	▭	Entities in ER Model
Ellipse	⬭	Attributes in ER Model
Diamond	◇	Relationships among Entities
Line	—	Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse	◎	Multi-Valued Attributes
Double Rectangle	⧈	Weak Entity

DATA MODELS

- **OBJECT BASED DATA MODEL**
- Object based data model is based upon ***real world situations***. These situations are represented as **objects**, with different **attributes**. All these object have ***multiple relationships*** between them.
- Elements of Object Oriented data model
- **Objects:** The real world entities and situations are represented as objects in the object oriented database model.
- **Attributes and Method:** Every object has certain characteristics. These are represented using Attributes. The behavior of the objects is represented using Methods.
- **Class:** Similar attributes and methods are grouped together using a class. An object can be called as an instance of the class.
- **Inheritance:** A new class can be derived from the original class. The derived class contains attributes and methods of the original class as well as its own.

DATA MODELS


Employee
Attributes
Name
Job_Title
Phone_No
Salary
Dept_ID
Methods
Get Hired
Change Number
Department
Attributes
Dept_ID
Dept_Name
Methods
Change Department

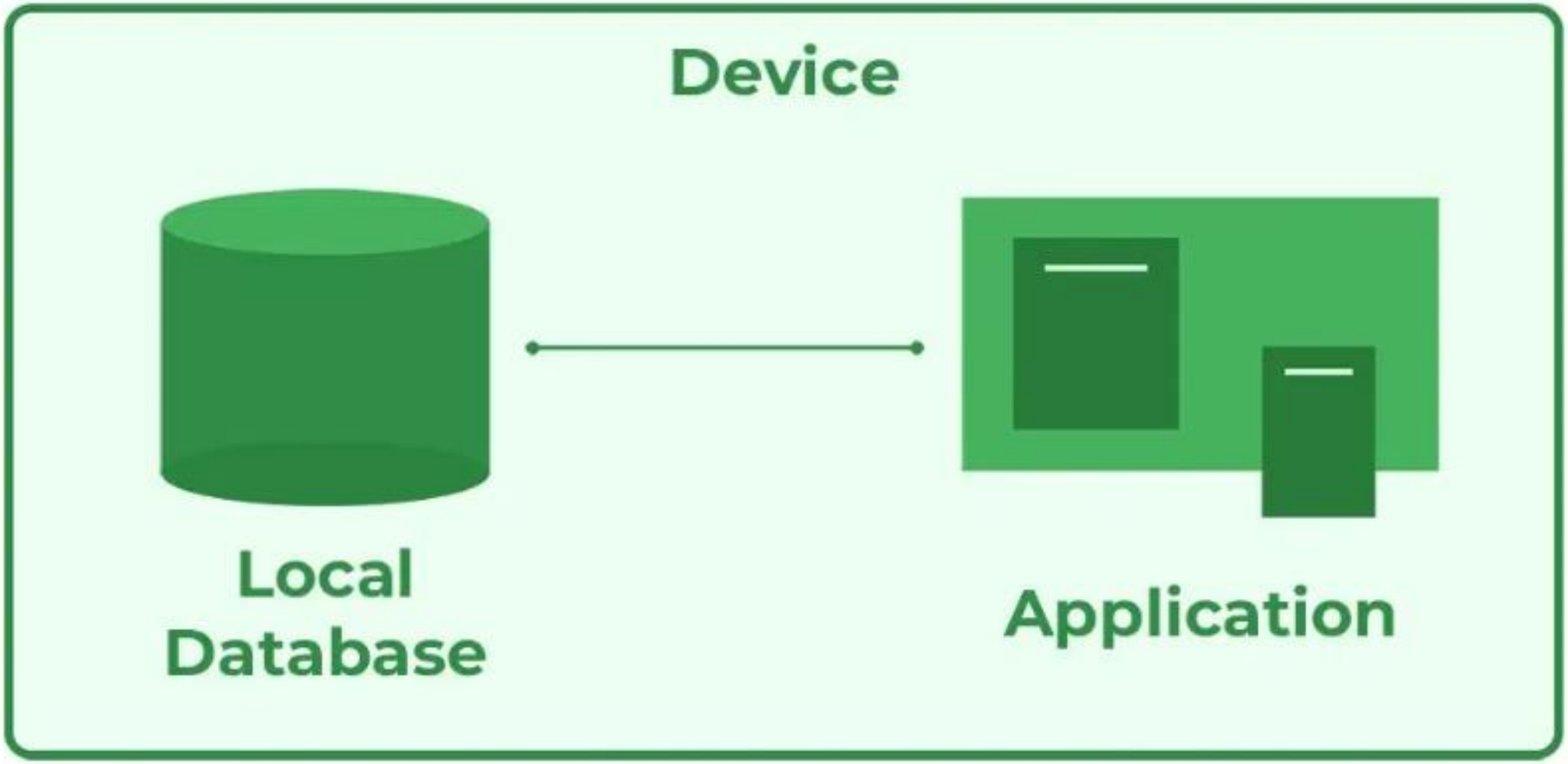
DATABASE ARCHITECTURE

- A Database store a lot of critical information to access data quickly and securely. Hence it is important to select the correct architecture for efficient data management.
- DBMS Architecture helps users to get their requests done while connecting to the database.
- We choose database architecture depending on several factors like the size of the database, number of users, and relationships between the users.
- There are two types of database models that we generally use, are logical model and physical model.

DATABASE ARCHITECTURE

- **Types of DBMS Architecture**
- **1-TIER ARCHITECTURE**
- In 1-Tier Architecture the database is directly available to the user, the user can directly sit on the DBMS and use it that is, the client, server, and Database are all present on the same machine.
- **For Example:** to learn SQL we set up an SQL server and the database on the local system. This enables us to directly interact with the relational database and execute operations.
- The industry won't use this architecture they logically go for 2-Tier and 3-Tier Architecture.

DATABASE ARCHITECTURE



DBMS 1-Tier Architecture

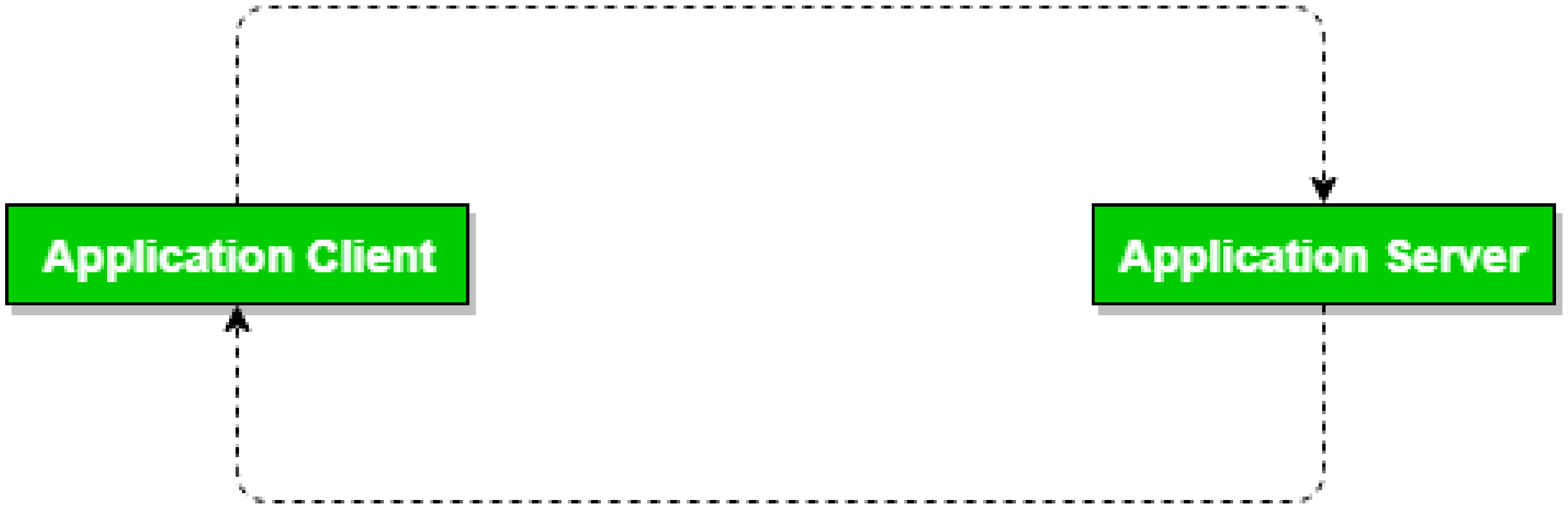
DATABASE ARCHITECTURE

- **ADVANTAGES OF 1-TIER ARCHITECTURE**
- **Simple Architecture:** 1-Tier Architecture is the most simple architecture to set up, as only a single machine is required to maintain it.
- **Cost-Effective:** No additional hardware is required for implementing 1-Tier Architecture, which makes it cost-effective.
- **Easy to Implement:** 1-Tier Architecture can be easily deployed, and hence it is mostly used in small projects.

DATABASE ARCHITECTURE

- **2-TIER ARCHITECTURE**
- The 2-tier architecture is similar to a basic client-server model.
- The application at the client end directly communicates with the database on the server side.
- APIs like ODBC and JDBC are used for this interaction.
- The server side is responsible for providing query processing and transaction management functionalities.
- On the client side, the user interfaces and application programs are run.
- The application on the client side establishes a connection with the server side in order to communicate with the DBMS.

DATABASE ARCHITECTURE



DBMS 2-Tier Architecture

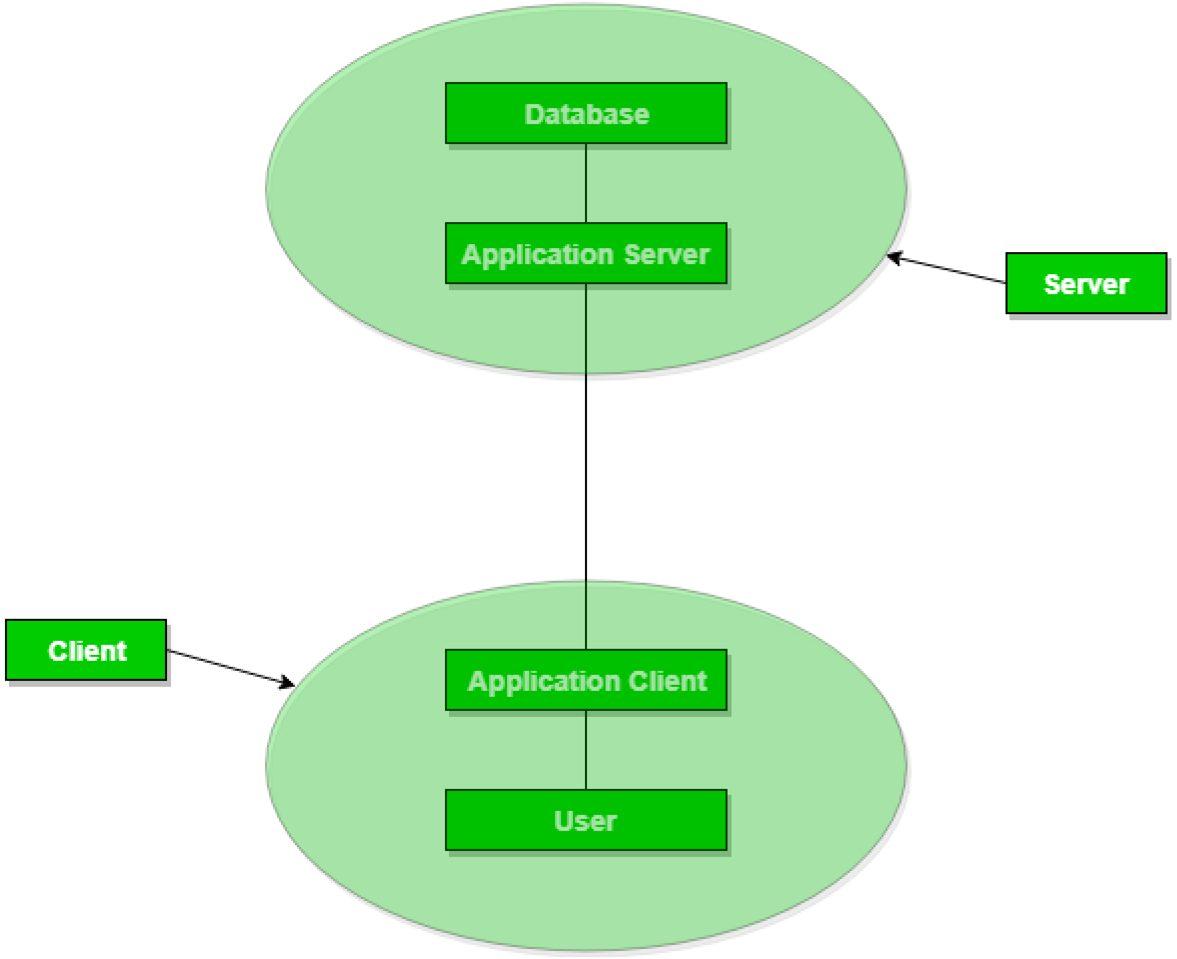
DATABASE ARCHITECTURE

- **ADVANTAGES OF 2-TIER ARCHITECTURE**
- **Easy to Access:** 2-Tier Architecture makes easy access to the database, which makes fast retrieval.
- **Scalable:** We can scale the database easily, by adding clients or by upgrading hardware.
- **Low Cost:** 2-Tier Architecture is cheaper than 3-Tier Architecture and Multi-Tier Architecture.
- **Easy Deployment:** 2-Tier Architecture is easy to deploy than 3-Tier Architecture.
- **Simple:** 2-Tier Architecture is easily understandable as well as simple because of only two components.

DATABASE ARCHITECTURE

- **3-TIER ARCHITECTURE**
- In 3-Tier Architecture, there is another layer between the client and the server.
- The client does not directly communicate with the server.
- Instead, it interacts with an application server which further communicates with the database system and then the query processing and transaction management takes place.
- This intermediate layer acts as a medium for the exchange of partially processed data between the server and the client.
- This type of architecture is used in the case of large web applications.

DATABASE ARCHITECTURE



DBMS 3-Tier Architecture

DATABASE ARCHITECTURE

- **ADVANTAGES OF 3-TIER ARCHITECTURE**
- **Enhanced scalability:** Scalability is enhanced due to distributed deployment of application servers. Now, individual connections need not be made between the client and server.
- **Data Integrity:** 3-Tier Architecture maintains Data Integrity. Since there is a middle layer between the client and the server, data corruption can be avoided/removed.
- **Security:** 3-Tier Architecture Improves Security. This type of model prevents direct interaction of the client with the server thereby reducing access to unauthorized data.

ROLES IN DATABASE ENVIRONMENT

- **DATABASE DESIGN**
- The structure of the database is determined during database design.
- It can be an extremely complex task.
- Need to think of the data first and then application.

ROLES IN DATABASE ENVIRONMENT

- **ROLES IN DATABASE ENVIRONMENT**
- There are 4 distinct types of people:
- Data Administrator
- Database Administrator
- Database Designers
- Application Developers
- End-Users

ROLES IN DATABASE ENVIRONMENT

- **ROLES IN DATABASE ENVIRONMENT**
- **DATA ADMINISTRATORS**
- DA (data administrator) is responsible for the management of the data resource:
- Database planning
- Development & maintenance of standards
- Policies and procedures
- Conceptual/logical database design

ROLES IN DATABASE ENVIRONMENT

- **ROLES IN DATABASE ENVIRONMENT**
- **DATABASE ADMINISTRATORS**
- DBA (Database Administrator) is responsible for the physical realization of the database:
- Physical database design & implementation
- Security & integrity control
- Maintenance of operational system
- Ensuring satisfactory performance of the applications for users

ROLES IN DATABASE ENVIRONMENT

- **ROLES IN DATABASE ENVIRONMENT**
- **DATABASE DESIGNERS**
- Database designers is concerned with:
- Identifying the data
- Identifying relationship between entities and attributes
- Identify the relationships between the data
- Understand the constraints on the data (business rules)

ROLES IN DATABASE ENVIRONMENT

- **ROLES IN DATABASE ENVIRONMENT**
- **APPLICATION DEVELOPERS**
- They worked from the specification produced by systems analysts
- Each program may contain statements that request the DBMS to perform some operation:
  - Retrieving data
  - Insert data
  - Delete data
  - Updating data

ROLES IN DATABASE ENVIRONMENT

- **ROLES IN DATABASE ENVIRONMENT**
- **END USERS**
- Clients of the database
- Can be classified as:
  - Naïve users
  - Typically unaware of the DBMS
  - Sophisticated users
  - Familiar with the structure of the DBMS
  - May use a high-level query language to perform required operation

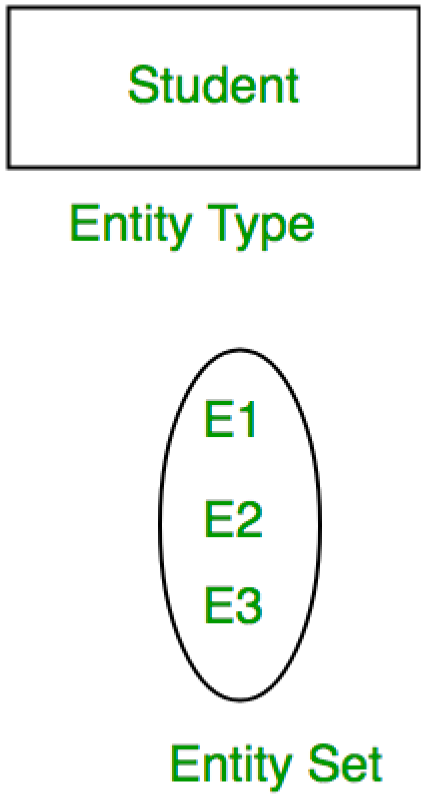
THE ENTITY-RELATIONSHIP MODEL

- **ENTITY:** An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

- **ENTITY SET:** An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set.

THE ENTITY-RELATIONSHIP MODEL

- In ER diagram, Entity Type is represented as:



THE ENTITY-RELATIONSHIP MODEL

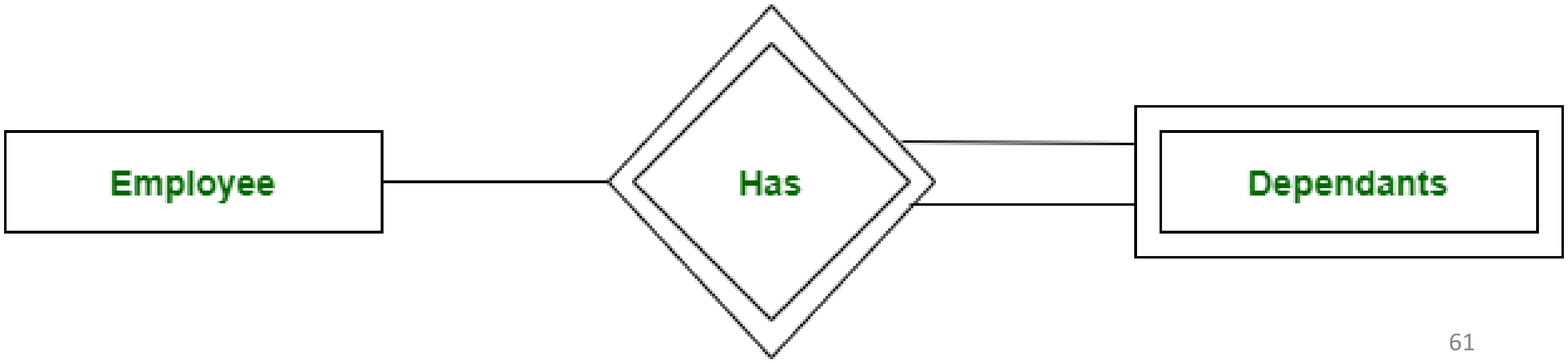
- **1. STRONG ENTITY**
- A Strong Entity is a type of entity that has a key Attribute.
- Strong Entity does not depend on other Entity in the Schema.
- It has a primary key, that helps in identifying it uniquely, and it is represented by a rectangle.
- These are called Strong Entity Types.

THE ENTITY-RELATIONSHIP MODEL

- **2. WEAK ENTITY**
- An Entity type has a key attribute that uniquely identifies each entity in the entity set.
- But some entity type exists for which key attributes can't be defined.
- These are called Weak Entity types.

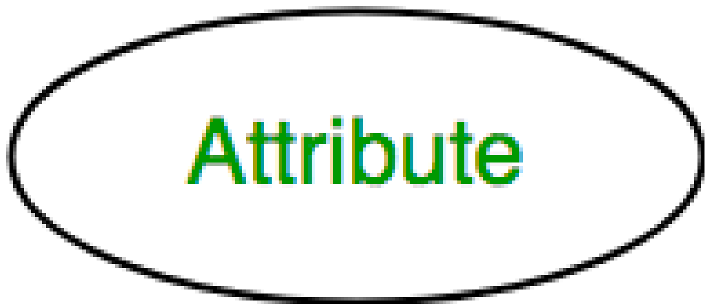
THE ENTITY-RELATIONSHIP MODEL

- **2. WEAK ENTITY**
- A weak entity type is represented by a Double Rectangle. The relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.



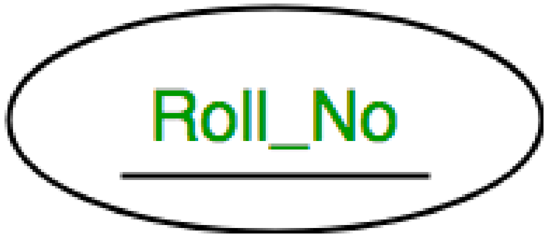
THE ENTITY-RELATIONSHIP MODEL

- **ATTRIBUTES**
- Attributes are the properties that define the entity type.
- For example, Roll_No, Name, DOB, Age, Address, and Mobile_No are the attributes that define entity type Student.



THE ENTITY-RELATIONSHIP MODEL

- **ATTRIBUTES**
- **1. KEY ATTRIBUTE**
- The attribute which uniquely identifies each entity in the entity set is called the key attribute.
- For example, Roll_No will be unique for each student. In ER diagram, the key attribute is represented by an oval with underlying lines.



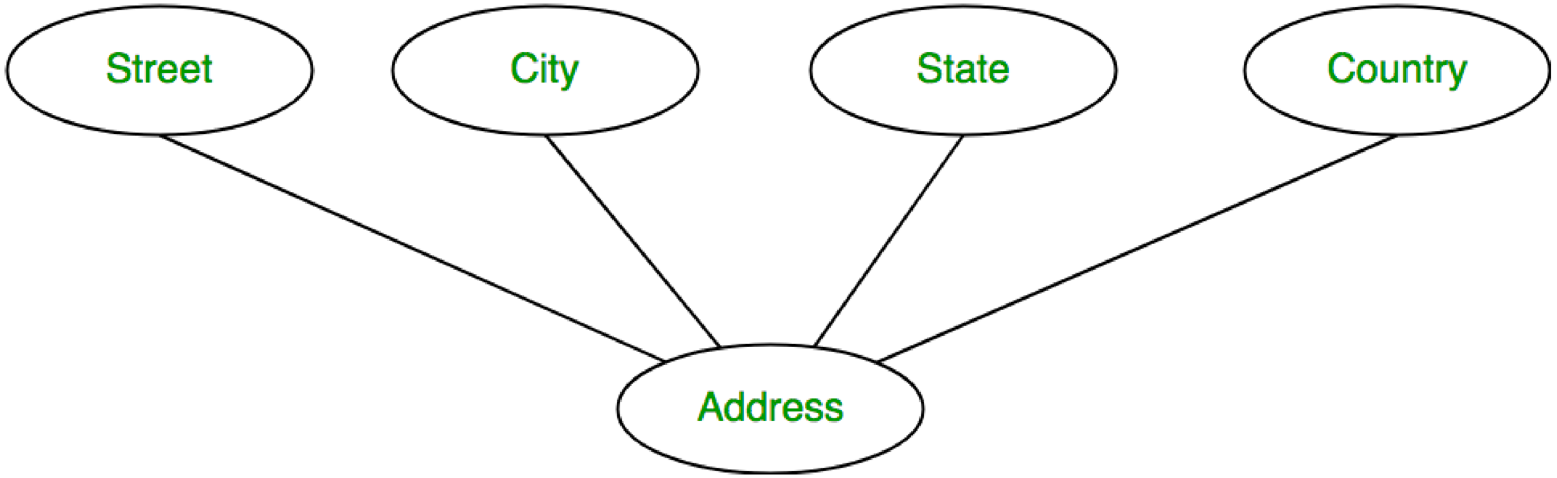
THE ENTITY-RELATIONSHIP MODEL

- **ATTRIBUTES**
- **2. COMPOSITE ATTRIBUTE**
- An attribute composed of many other attributes is called a composite attribute.

- For example, the Address attribute of the student Entity type consists of Street, City, State, and Country.

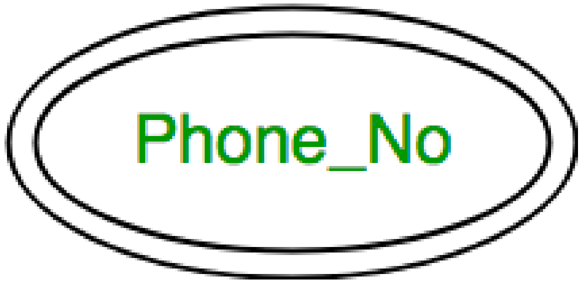
THE ENTITY-RELATIONSHIP MODEL

- **ATTRIBUTES**
- **2. COMPOSITE ATTRIBUTE**
- In ER diagram, the composite attribute is represented by an oval comprising of ovals.



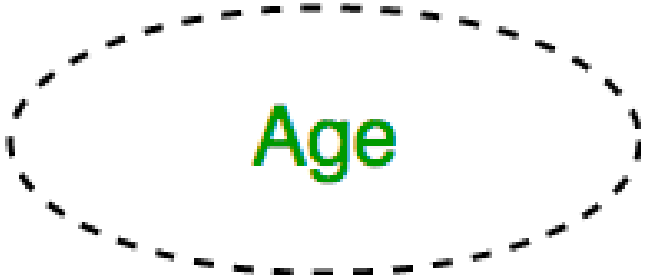
THE ENTITY-RELATIONSHIP MODEL

- **ATTRIBUTES**
- **3. MULTIVALUED ATTRIBUTE**
- An attribute consisting of more than one value for a given entity.
- For example, Phone_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.



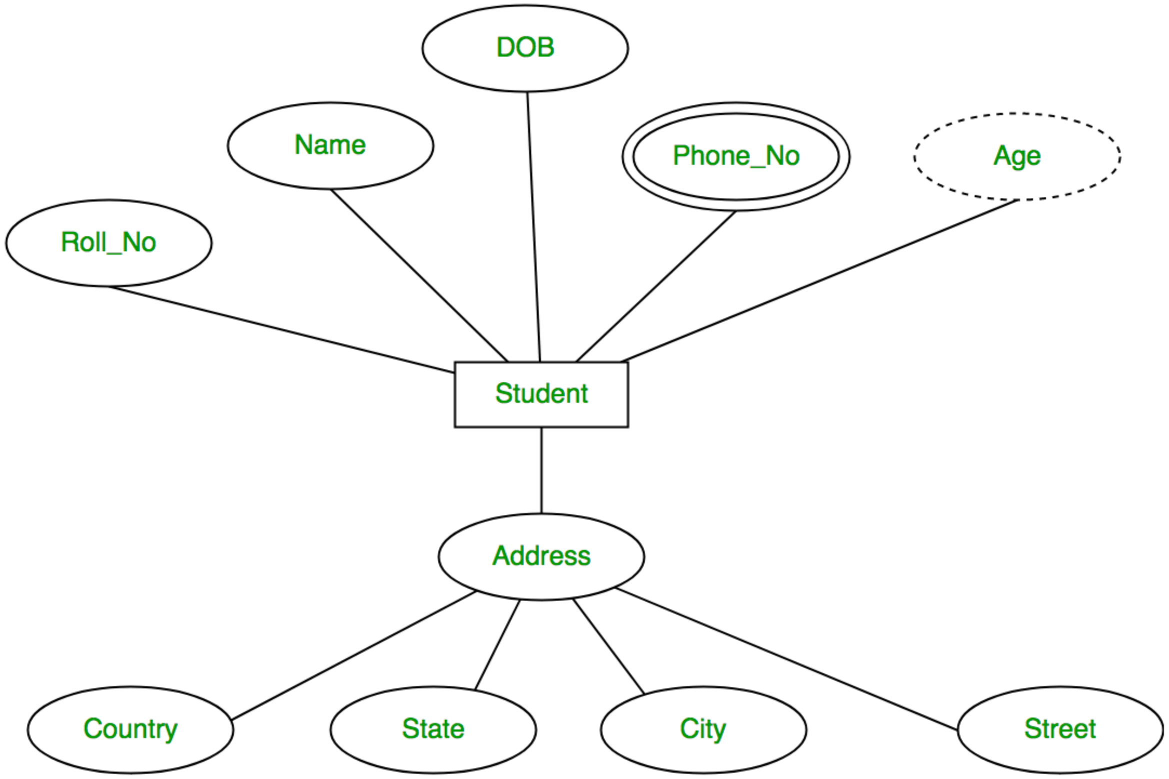
THE ENTITY-RELATIONSHIP MODEL

- **ATTRIBUTES**
- **4. DERIVED ATTRIBUTE**
- An attribute that can be derived from other attributes of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, the derived attribute is represented by a dashed oval.



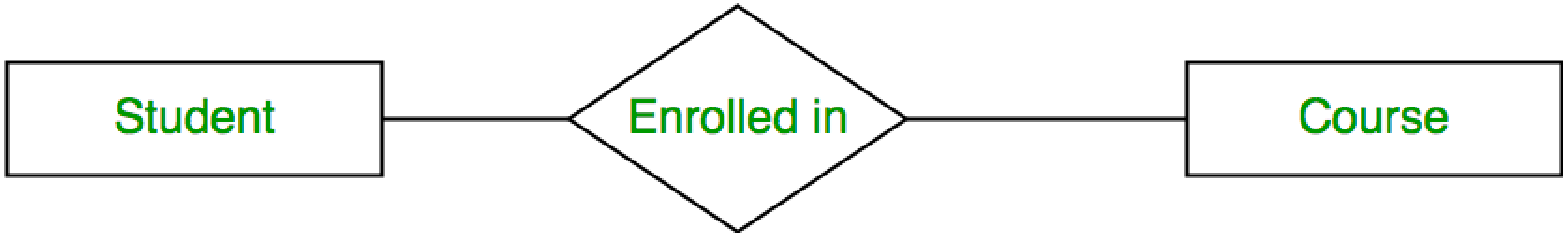
THE ENTITY-RELATIONSHIP MODEL

- **ATTRIBUTES**
- The Complete Entity Type Student with its Attributes can be represented as:



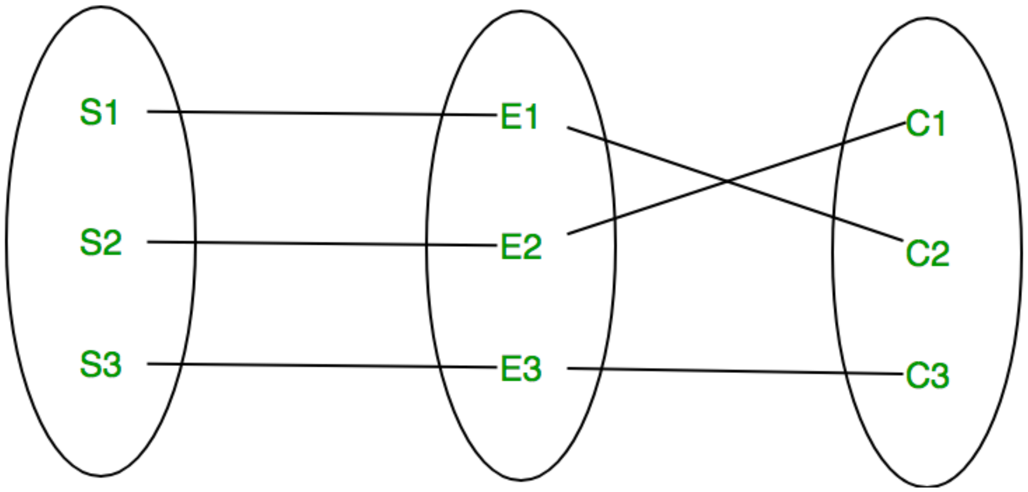
THE ENTITY-RELATIONSHIP MODEL

- **RELATIONSHIP TYPE AND RELATIONSHIP SET**
- A Relationship Type represents the association between entity types.
- For example, ‘Enrolled in’ is a relationship type that exists between entity type Student and Course.
- In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.



THE ENTITY-RELATIONSHIP MODEL

- **RELATIONSHIP TYPE AND RELATIONSHIP SET**
- A set of relationships of the same type is known as a relationship set.
- The following relationship set depicts S1 as enrolled in C2, S2 as enrolled in C1, and S3 as registered in C3.

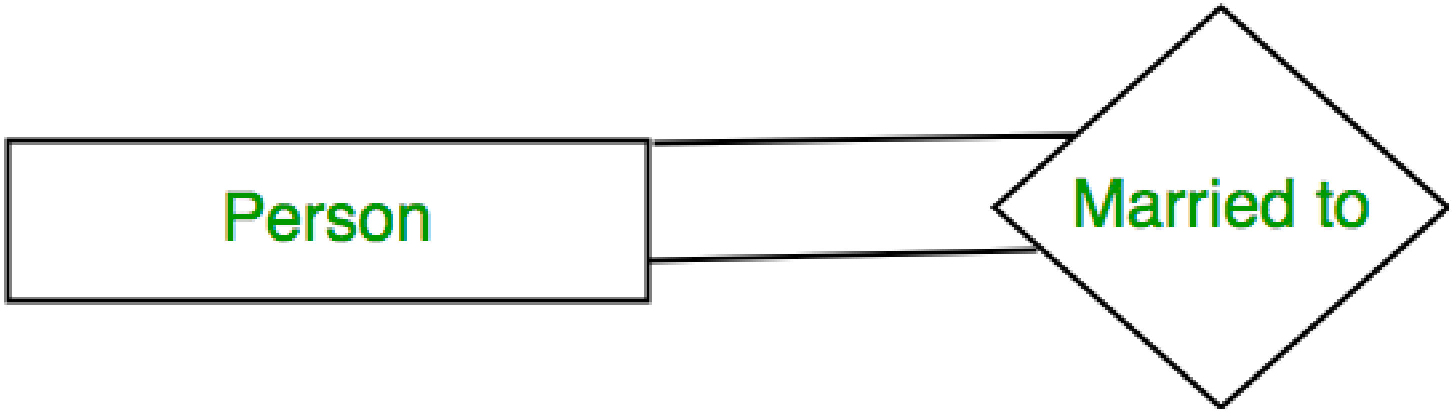


THE ENTITY-RELATIONSHIP MODEL

- **DEGREE OF RELATIONSHIP SET**
- ***The number of different entity sets participating in a relationship set is called the degree of a relationship set.***

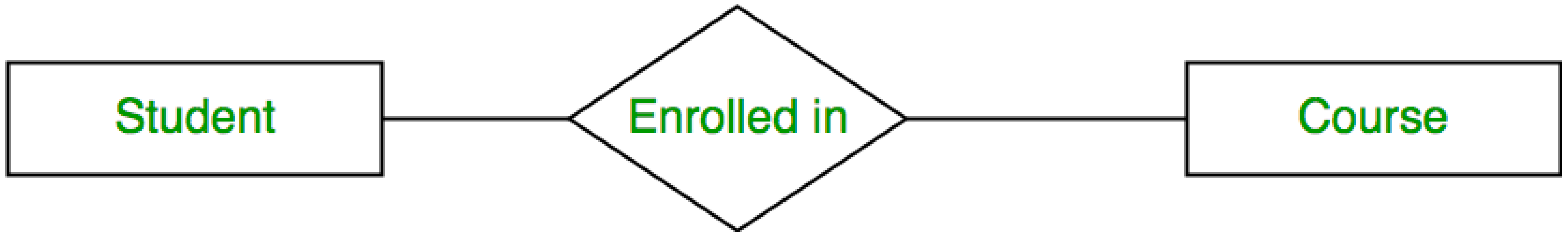
THE ENTITY-RELATIONSHIP MODEL

- **DEGREE OF RELATIONSHIP SET**
- **1. Unary Relationship:** When there is ***only ONE entity set*** participating in a relation, the relationship is called a unary relationship.
- For example, one person is married to only one person.



THE ENTITY-RELATIONSHIP MODEL

- **DEGREE OF RELATIONSHIP SET**
- **2. Binary Relationship:** When there are ***TWO entities set*** participating in a relationship, the relationship is called a binary relationship.
- For example, a Student is enrolled in a Course.



THE ENTITY-RELATIONSHIP MODEL

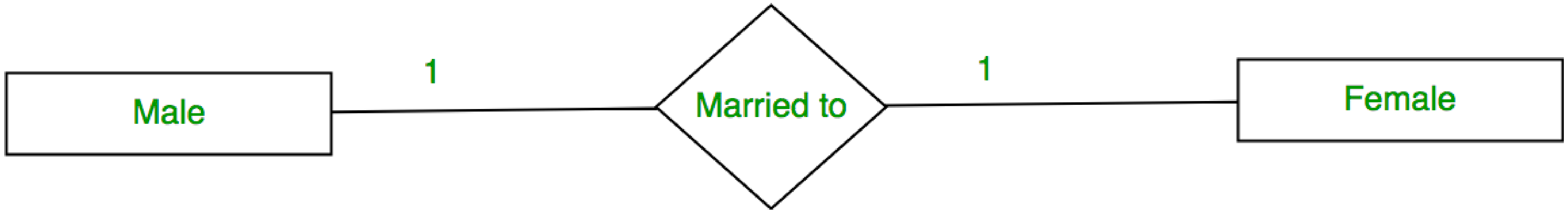
- **DEGREE OF RELATIONSHIP SET**
- **2. n-ary Relationship:** When there are n entities set participating in a relation, the relationship is called an n-ary relationship.

THE ENTITY-RELATIONSHIP MODEL

- **CARDINALITY**
- ***The number of times an entity of an entity set participates in a relationship set is known as cardinality.***

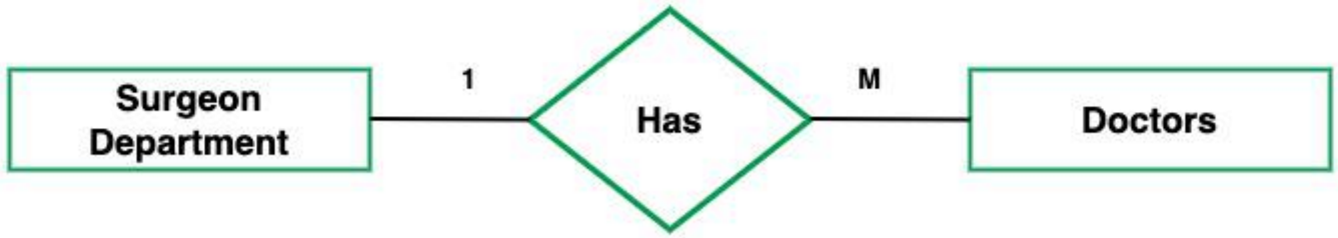
THE ENTITY-RELATIONSHIP MODEL

- **CARDINALITY**
- **1. One-to-One:** When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one.
- Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one.



THE ENTITY-RELATIONSHIP MODEL

- **CARDINALITY**
- **2. One-to-Many:** In one-to-many mapping as well where each entity can be related to more than one relationship and the total number of tables that can be used in this is 2

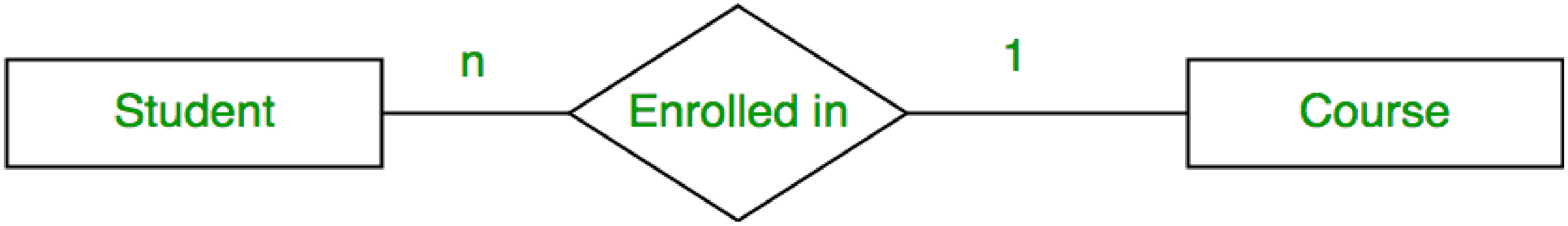


THE ENTITY-RELATIONSHIP MODEL

- **CARDINALITY**
- **3. Many-to-One:** When entities in one entity set can take part only ***once*** in the relationship set and entities in other entity sets can take part ***more than once*** in the relationship set, ***cardinality is many to one.***
- Let us assume that a student can take only one course but one course can be taken by many students.
- So the cardinality will be n to 1.
- It means that for one course there can be n students but for one student, there will be only one course.

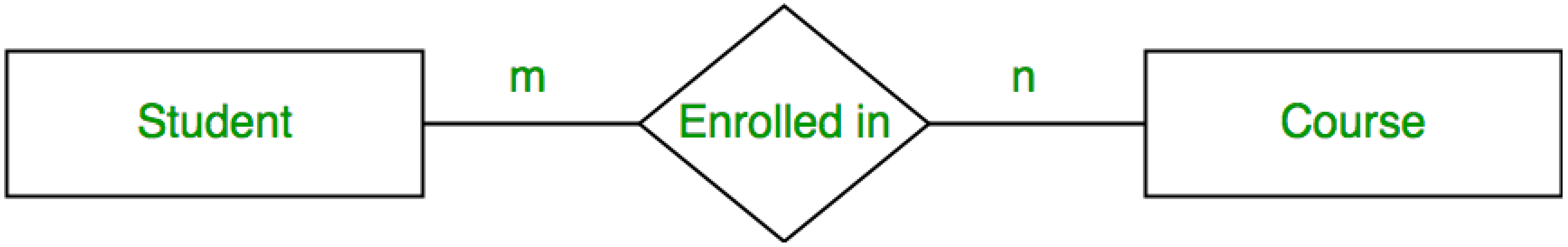
THE ENTITY-RELATIONSHIP MODEL

- **CARDINALITY**
- **3. Many-to-One**
- The total number of tables that can be used in this is 3.



THE ENTITY-RELATIONSHIP MODEL

- **CARDINALITY**
- **4. Many-to-One:** When entities in all entity sets can take part more than once in the relationship cardinality is many to many.
- Let us assume that a student can take more than one course and one course can be taken by many students.
- The total number of tables that can be used in this is 3.

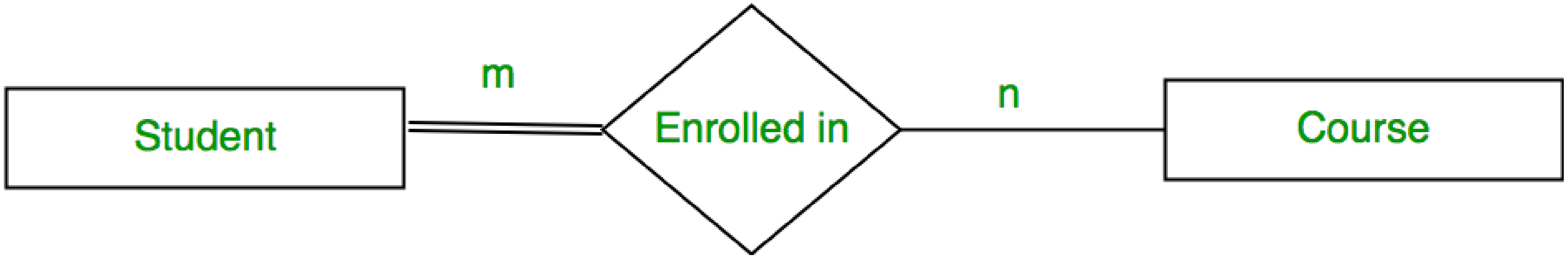


THE ENTITY-RELATIONSHIP MODEL

- **PARTICIPATION CONSTRAINT**
- Participation Constraint is applied to the ***entity participating in the relationship set.***
- **1. Total Participation** – Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of students will be total. Total participation is shown by a double line in the ER diagram.
- **2. Partial Participation** – The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the students, the participation in the course will be partial.

THE ENTITY-RELATIONSHIP MODEL

- **PARTICIPATION CONSTRAINT**
- The diagram depicts the ‘Enrolled in’ relationship set with Student Entity set having total participation and Course Entity set having partial participation.



THE ENTITY-RELATIONSHIP DIAGRAM

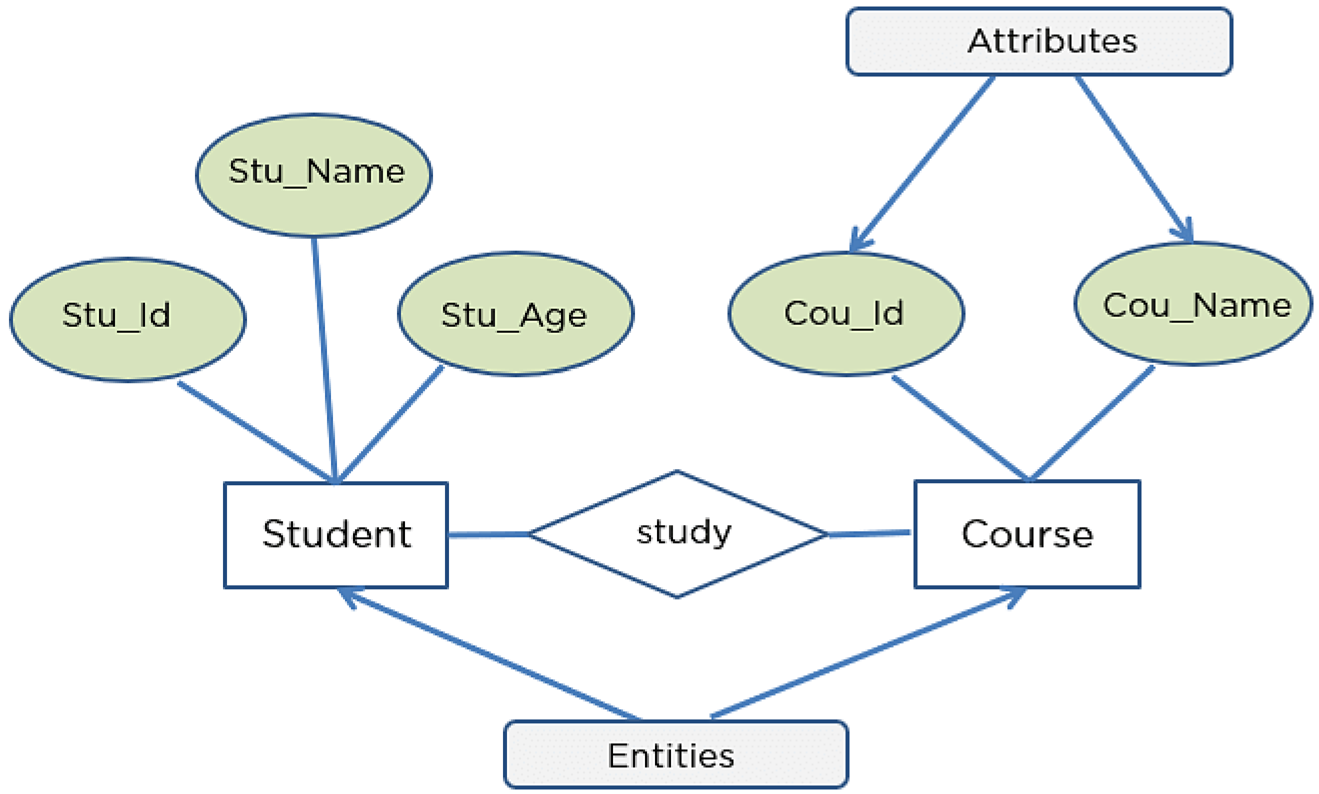
- **HOW TO DRAW ER DIAGRAM?**
- First, identify all the Entities. Embed all the entities in a rectangle and label them properly.
- Identify relationships between entities and connect them using a diamond in the middle, illustrating the relationship. Do not connect relationships with each other.
- Connect attributes for entities and label them properly.
- Eradicate any redundant entities or relationships.
- Make sure your ER Diagram supports all the data provided to design the database.
- Effectively use colors to highlight key areas in your diagrams.

THE ENTITY-RELATIONSHIP DIAGRAM

- **Draw an ER diagram student studying one course**
- Entities : Student and Course
- Attributes of Student and Course.
- Relationship between Student and Course.

THE ENTITY-RELATIONSHIP DIAGRAM

- **Draw an ER diagram for student studying one course**

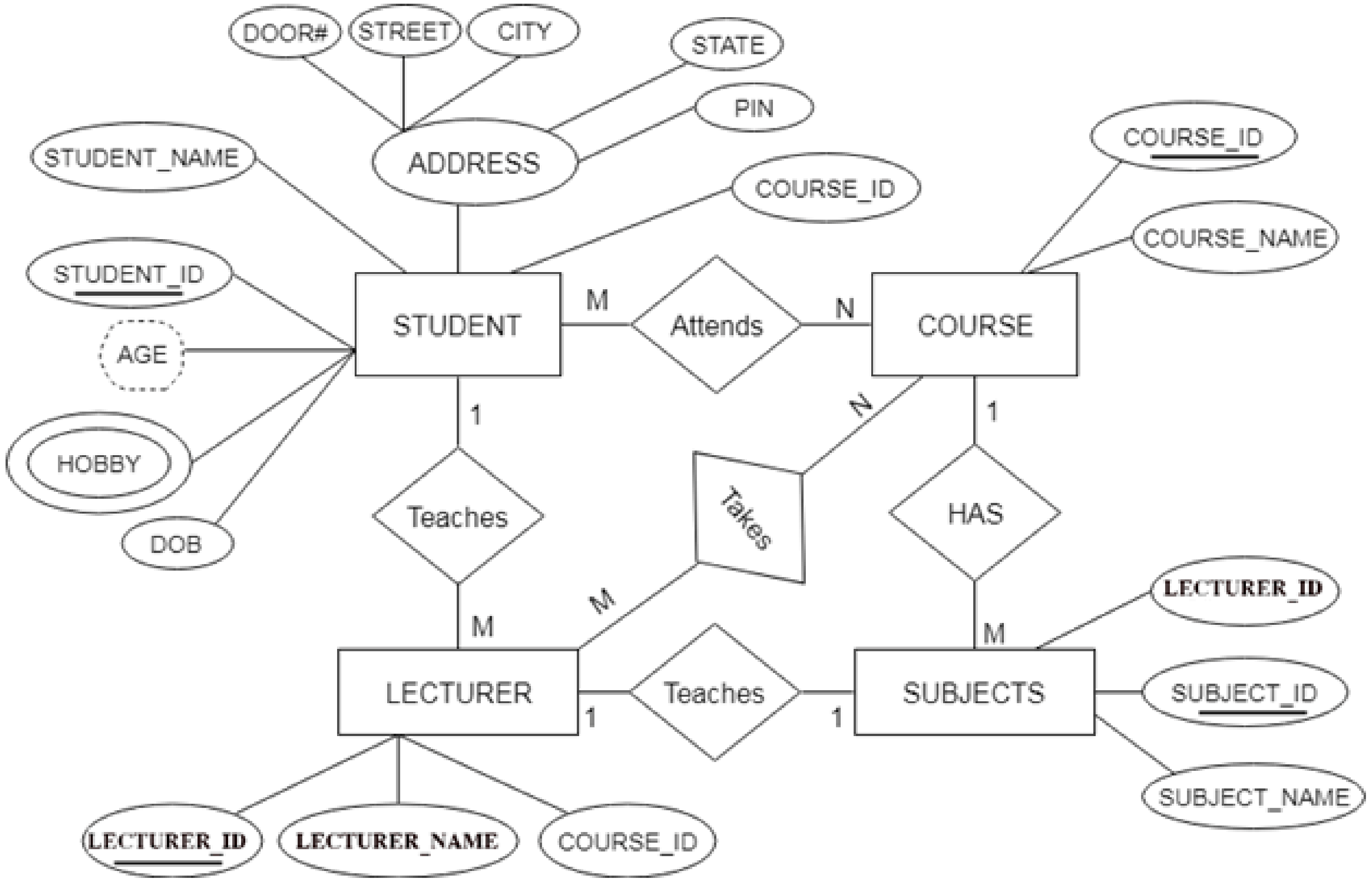


THE ENTITY-RELATIONSHIP DIAGRAM

- **Draw an ER diagram for student studying multiple courses taught by multiple lectures.**
- Entities : Student, Teacher and Course
- Attributes of Student, Teacher and Course.
- Relationship between Student, Teacher and Course including cardinality and participation constraints.

THE ENTITY-RELATIONSHIP DIAGRAM

- **Draw an ER diagram for student studying multiple courses taught by multiple lectures.**

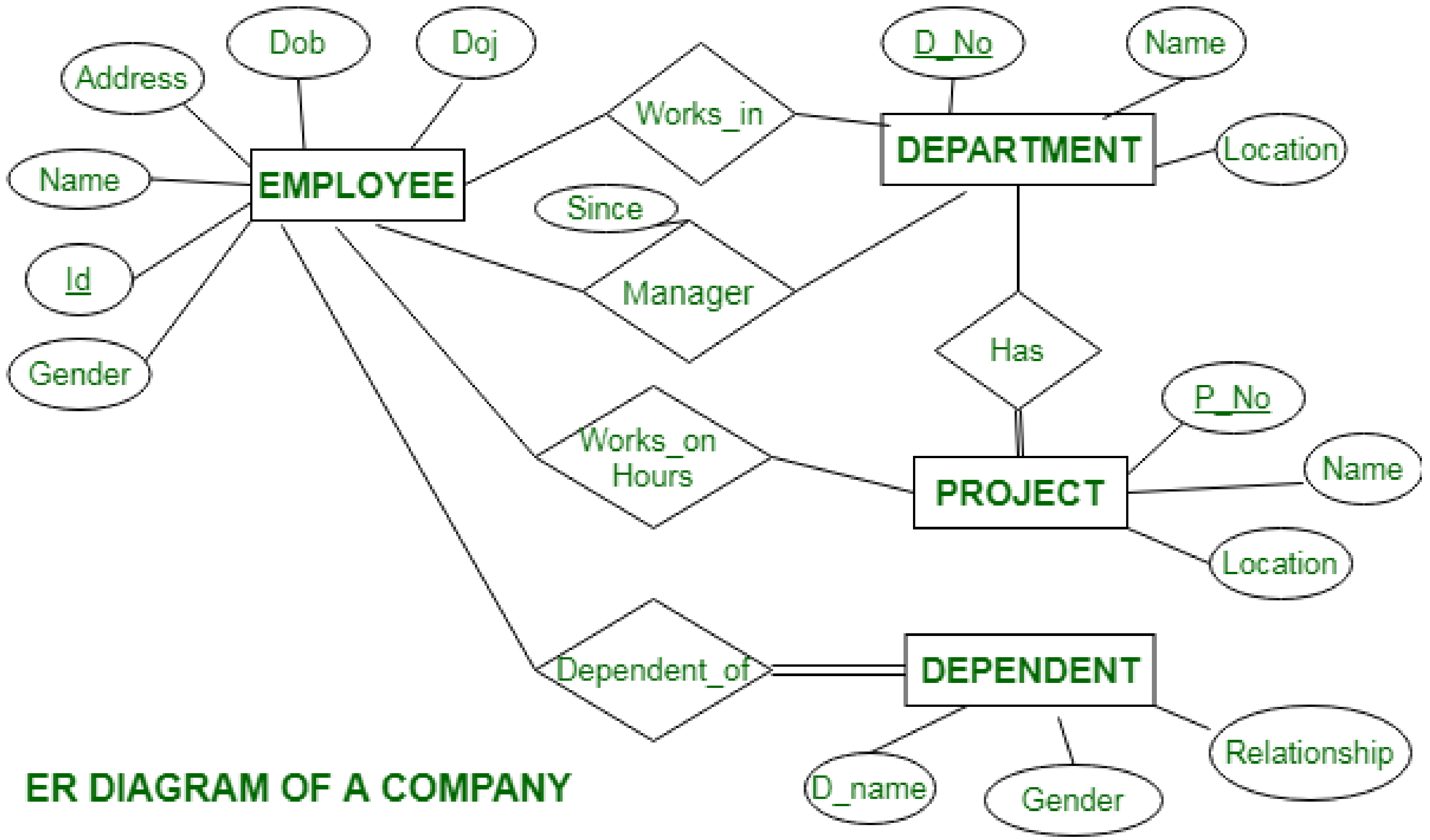


THE ENTITY-RELATIONSHIP DIAGRAM

- **Draw an ER diagram for a company database.**
- Entities : DEPARTMENT, PROJECT, EMPLOYEE, DEPENDENT
- Attributes of DEPARTMENT, PROJECT, EMPLOYEE, DEPENDENT.
- Relationship between DEPARTMENT, PROJECT, EMPLOYEE, DEPENDENT including cardinality and participation constraints.

THE ENTITY-RELATIONSHIP DIAGRAM

- **Draw an ER diagram for a company database.**



THE ENTITY-RELATIONSHIP DIAGRAM

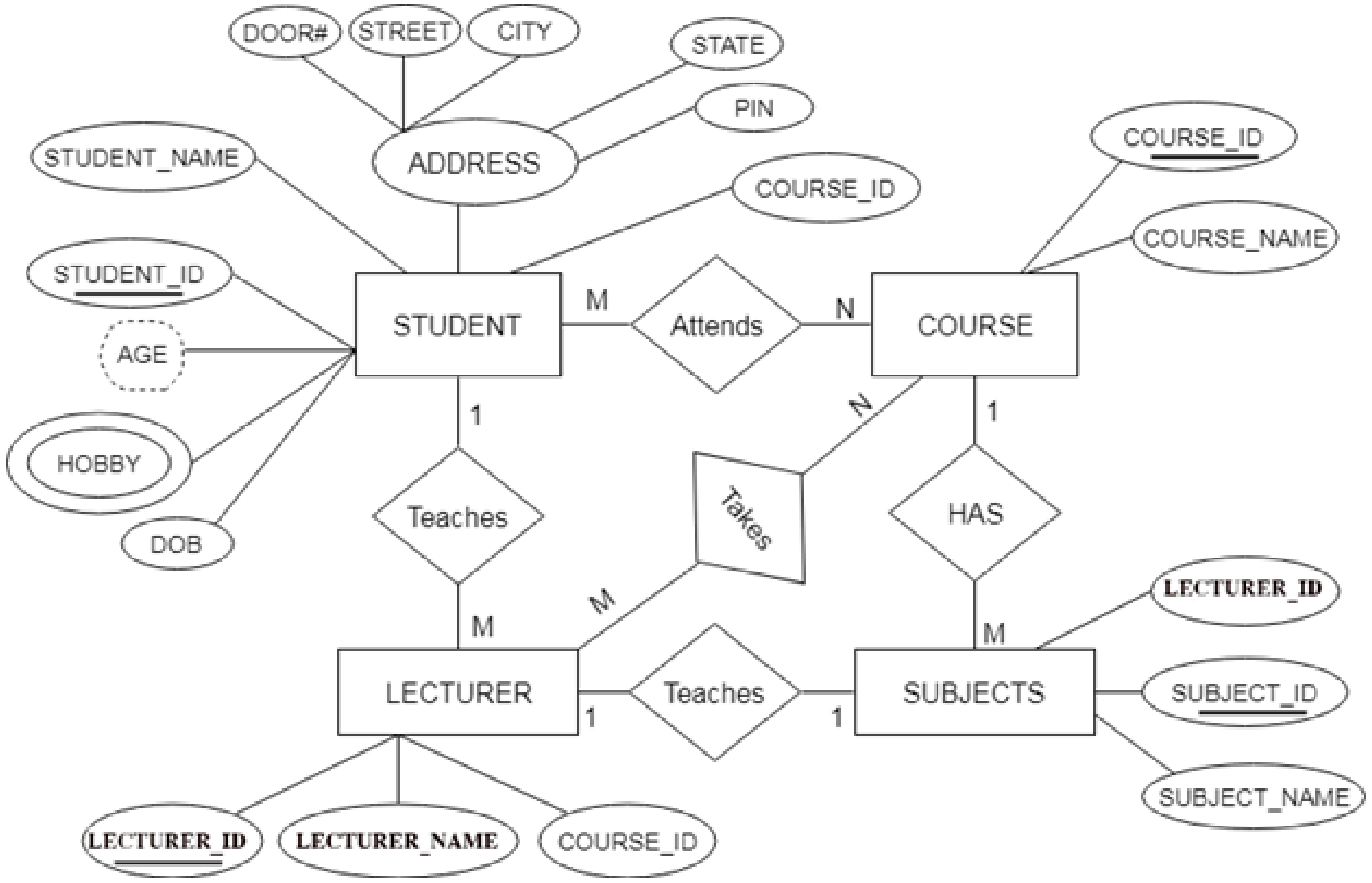
- **Draw an ER diagram for representing student admission process.**

REDUCTION TO RELATIONAL SCHEMAS

- **ER MODEL TO RELATIONAL MODEL.**
- The database can be represented using the notations, and these notations can be reduced to a collection of tables.
- In the database, every entity set or relationship set can be represented in tabular form.

REDUCTION TO RELATIONAL SCHEMAS

- **Draw an ER diagram for student studying multiple courses taught by multiple lectures.**



REDUCTION TO RELATIONAL SCHEMAS

- There are some points for converting the ER diagram to the table:
- **Entity type becomes a table.**
  - In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.
- **All single-valued attribute becomes a column for the table.**
  - In the STUDENT entity, STUDENT_NAME and STUDENT_ID form the column of STUDENT table. Similarly, COURSE_NAME and COURSE_ID form the column of COURSE table and so on.

REDUCTION TO RELATIONAL SCHEMAS

- **A key attribute of the entity type represented by the primary key.**
  - In the given ER diagram, COURSE_ID, STUDENT_ID, SUBJECT_ID, and LECTURE_ID are the key attribute of the entity.

- **The multivalued attribute is represented by a separate table**.
  - In the student table, a hobby is a multivalued attribute. So it is not possible to represent multiple values in a single column of STUDENT table. Hence we create a table STUD_HOBBY with column name STUDENT_ID and HOBBY. Using both the column, we create a composite key.

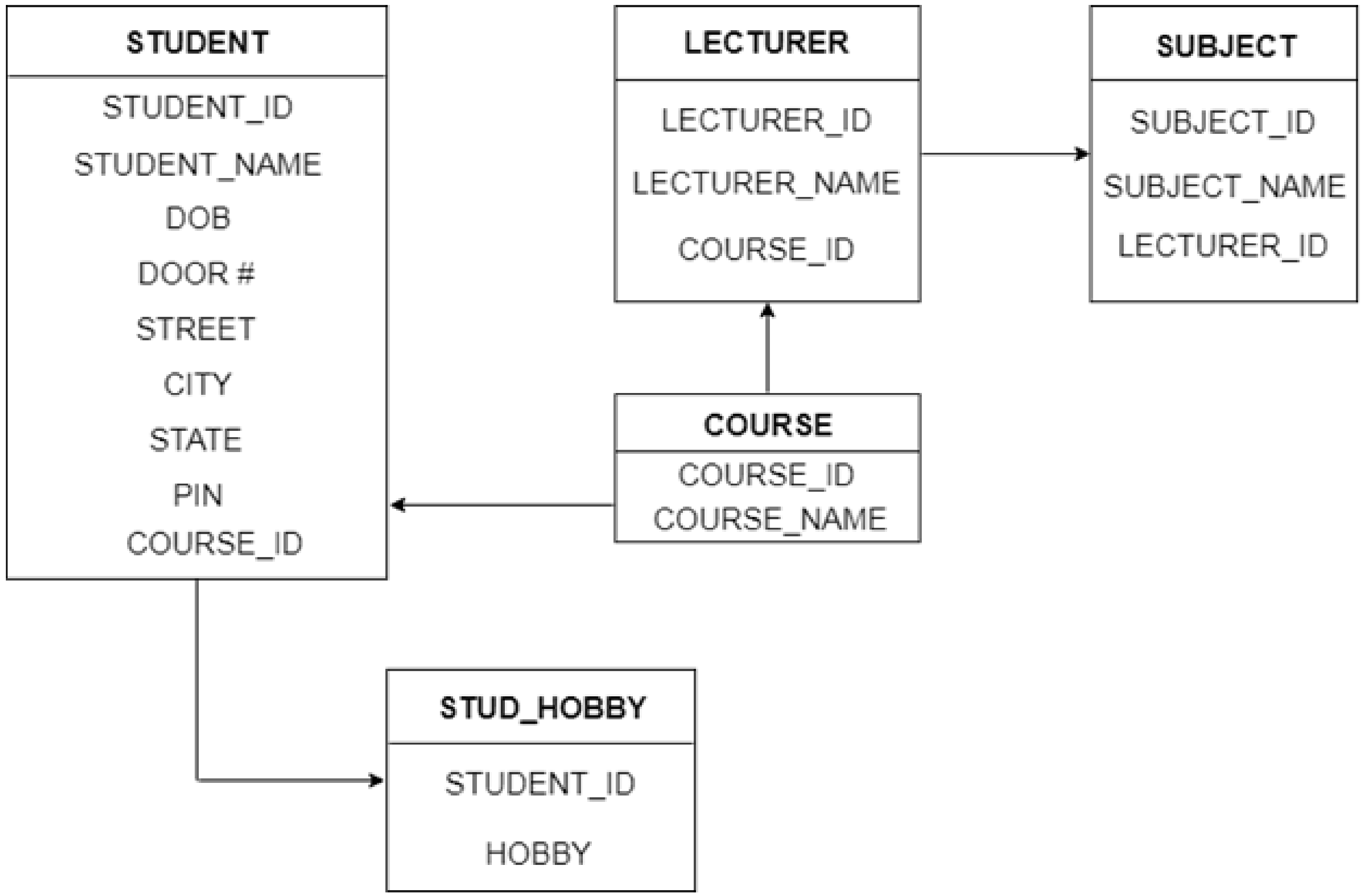
REDUCTION TO RELATIONAL SCHEMAS

- **Composite attribute represented by components.**
  - In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an individual column.

- **Derived attributes are not considered in the table.**
  - In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating the difference between current date and Date of Birth.

REDUCTION TO RELATIONAL SCHEMAS

- **Table structure for the given ER diagram is as below:**



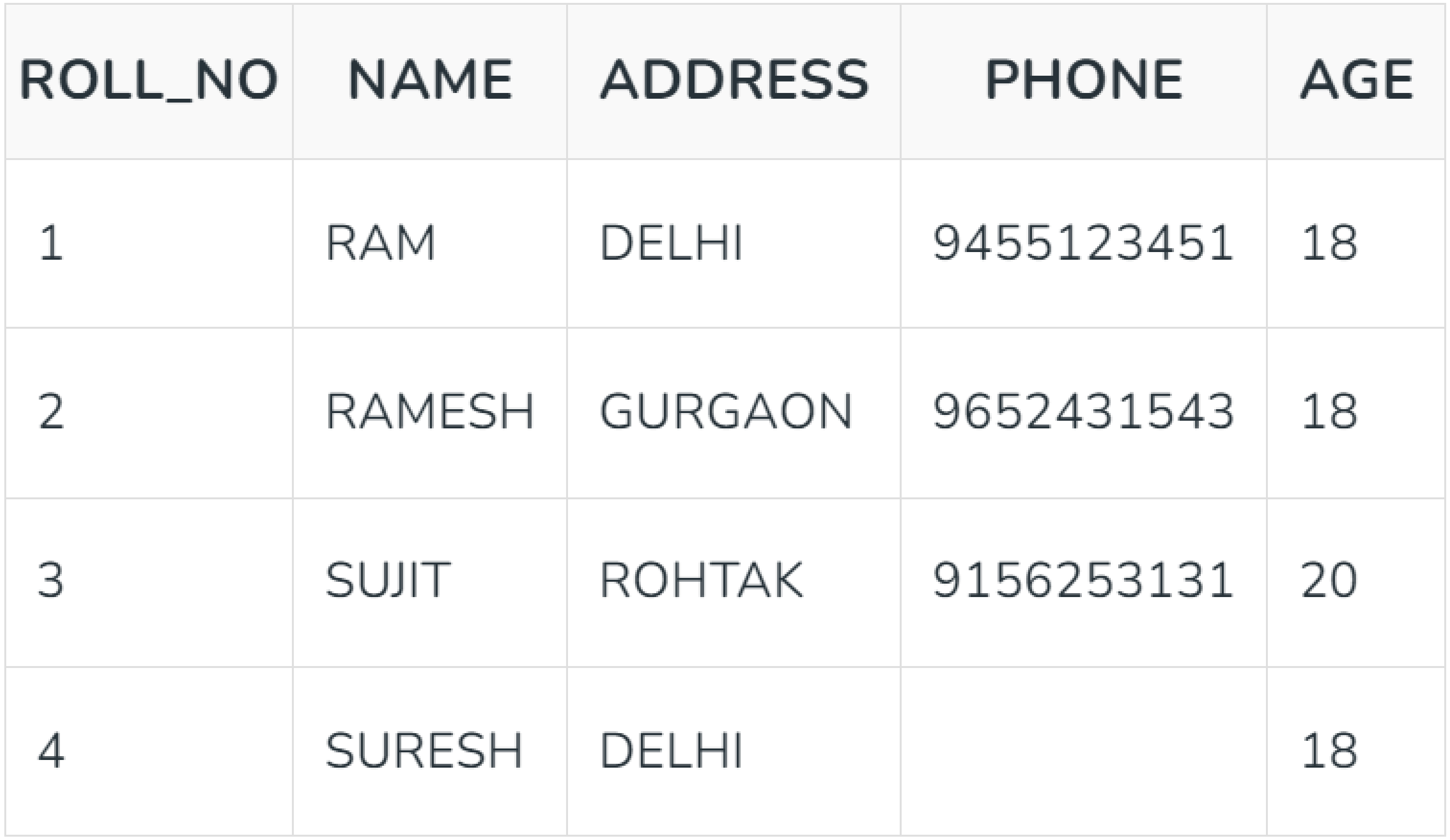
INTRODUCTION TO RELATIONAL MODEL

- ***The relational model represents how data is stored in Relational Databases.***

- A relational database consists of a collection of tables, each of which is assigned a unique name.

INTRODUCTION TO RELATIONAL MODEL

- Consider a relation STUDENT with attributes ROLL_NO, NAME, ADDRESS, PHONE, and AGE shown in the table.



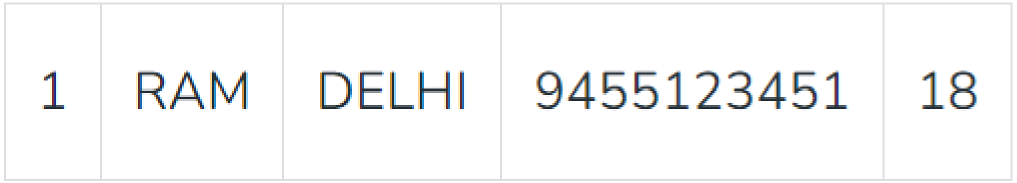
ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI		18

INTRODUCTION TO RELATIONAL MODEL

- **IMPORTANT TERMINOLOGIES**
- **Attribute:** Attributes are the properties that define an entity. e.g.; ROLL_NO, NAME, ADDRESS
- **Relation Schema:** A relation schema defines the structure of the relation and represents the name of the relation with its attributes. e.g.; STUDENT (ROLL_NO, NAME, ADDRESS, PHONE, and AGE) is the relation schema for STUDENT. If a schema has more than 1 relation, it is called Relational Schema.

INTRODUCTION TO RELATIONAL MODEL

- **IMPORTANT TERMINOLOGIES**
- **Tuple:** Each row in the relation is known as a tuple. The above relation contains 4 tuples, one of which is shown as:



1	RAM	DELHI	9455123451	18

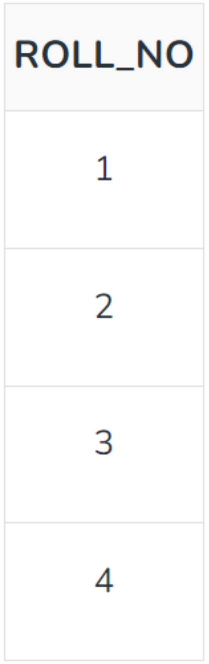
- **Relation Instance:** The set of tuples of a relation at a particular instance of time is called a relation instance. Table 1 shows the relation instance of STUDENT at a particular time. It can change whenever there is an insertion, deletion, or update in the database.

INTRODUCTION TO RELATIONAL MODEL

- **IMPORTANT TERMINOLOGIES**
- **Degree:** The number of attributes in the relation is known as the degree of the relation. The STUDENT relation defined above has degree 5.
- **Cardinality:** The number of tuples in a relation is known as cardinality. The STUDENT relation defined above has cardinality 4.

INTRODUCTION TO RELATIONAL MODEL

- **IMPORTANT TERMINOLOGIES**
- **Column:** The column represents the set of values for a particular attribute. The column ROLL_NO is extracted from the relation STUDENT.



ROLL_NO
1
2
3
4

INTRODUCTION TO RELATIONAL MODEL

- **IMPORTANT TERMINOLOGIES**
- **NULL Values:** The value which is not known or unavailable is called a NULL value. It is represented by blank space. e.g.; PHONE of STUDENT having ROLL_NO 4 is NULL.
- **Relation Key:** These are basically the keys that are used to identify the rows uniquely or also help in identifying tables. These are of the following types.
- Primary Key, Candidate Key, Super Key, Foreign Key, Alternate Key & Composite Key.

INTRODUCTION TO RELATIONAL MODEL

- **CONSTRAINTS IN RELATIONAL MODEL**
- ***While designing the Relational Model, we define some conditions which must hold for data present in the database are called Constraints.***
- These constraints are checked before performing any operation (insertion, deletion, and updation) in the database. If there is a violation of any of the constraints, the operation will fail.

INTRODUCTION TO RELATIONAL MODEL

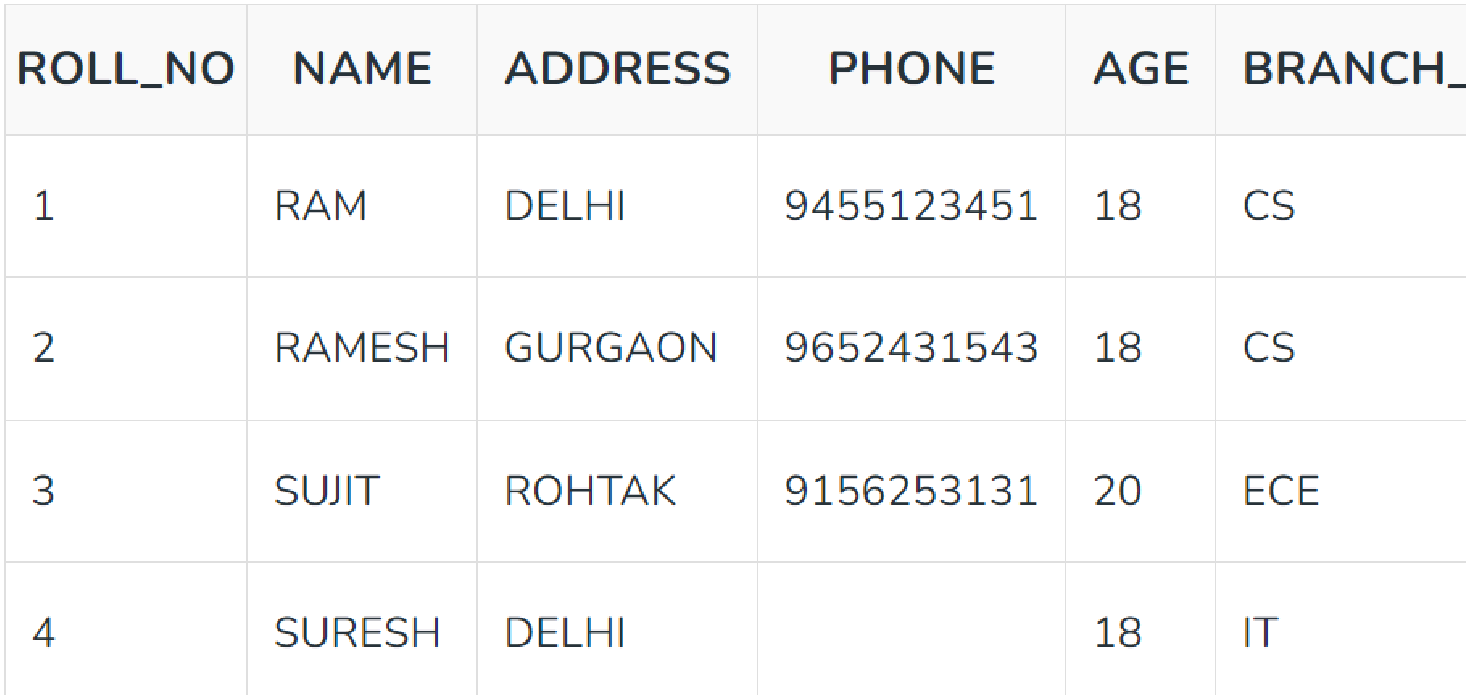
- **CONSTRAINTS IN RELATIONAL MODEL**
- **Domain Constraints**
- These are ***attribute-level constraints***.
- An attribute can only take values that lie inside the domain range.
- e.g.; If a constraint **AGE>0** is applied to **STUDENT** relation, inserting a **negative value** of AGE will result in failure.

INTRODUCTION TO RELATIONAL MODEL

- **CONSTRAINTS IN RELATIONAL MODEL**
- **Referential Integrity**
- When one attribute of a relation can only take values from another attribute of the same relation or any other relation, it is called referential integrity. Let us suppose we have 2 relations

INTRODUCTION TO RELATIONAL MODEL

- **CONSTRAINTS IN RELATIONAL MODEL**
- **Referential Integrity**
- Let us suppose we have 2 relations : **TABLE STUDENT**



ROLL_NO	NAME	ADDRESS	PHONE	AGE	BRANCH_
1	RAM	DELHI	9455123451	18	CS
2	RAMESH	GURGAON	9652431543	18	CS
3	SUJIT	ROHTAK	9156253131	20	ECE
4	SURESH	DELHI		18	IT

INTRODUCTION TO RELATIONAL MODEL

- **CONSTRAINTS IN RELATIONAL MODEL**
- **Referential Integrity**
- Let us suppose we have 2 relations : **TABLE BRANCH**

BRANCH_CODE	BRANCH_NAME
CS	COMPUTER SCIENCE
IT	INFORMATION TECHNOLOGY
ECE	ELECTRONICS AND COMMUNICATION ENGINEERING
CV	CIVIL ENGINEERING

INTRODUCTION TO RELATIONAL MODEL

- **CONSTRAINTS IN RELATIONAL MODEL**
- **Referential Integrity**
- **BRANCH_CODE** of **TABLE STUDENT** can only take the values which are present in **BRANCH_CODE** of **TABLE BRANCH** which is called **referential integrity constraint**.
- The relation which is referencing another relation is called **REFERENCING RELATION** (STUDENT in this case) and the relation to which other relations refer is called **REFERENCED RELATION** (BRANCH in this case).

INTRODUCTION TO RELATIONAL MODEL

- **ANOMALIES IN THE RELATIONAL MODEL**
- ***An anomaly is an irregularity or something which deviates from the expected or normal state.***

- When designing databases, we identify three types of anomalies: Insert, Update, and Delete.

INTRODUCTION TO RELATIONAL MODEL

- **ANOMALIES IN THE RELATIONAL MODEL**
- **Insertion Anomaly in Referencing Relation**
- We can’t insert a row in **REFERENCING RELATION** if referencing attribute’s value is not present in the referenced attribute value.
- e.g.; Insertion of a student with **BRANCH_CODE** ‘***ME***’ in **STUDENT** relation will result in an error because ‘**ME**’ is not present in **BRANCH_CODE** of **BRANCH**.

INTRODUCTION TO RELATIONAL MODEL

- **ANOMALIES IN THE RELATIONAL MODEL**
- **Deletion/ Updation Anomaly in Referenced Relation**
- We can't ***delete*** or ***update*** a row from **REFERENCED RELATION** if the value of **REFERENCED ATTRIBUTE** is used in the value of **REFERENCING ATTRIBUTE**. e.g; if we try to delete a tuple from **BRANCH** having **BRANCH_CODE** '**CS**', it will result in an error because '**CS**' is referenced by **BRANCH_CODE** of **STUDENT**, but if we try to delete the row from **BRANCH** with **BRANCH_CODE CV**, it will be deleted as the value is not been used by referencing relation.

INTRODUCTION TO RELATIONAL MODEL

- **ANOMALIES IN THE RELATIONAL MODEL**
- **On Delete Cascade**
- It will delete the tuples from REFERENCING RELATION if the value used by REFERENCING ATTRIBUTE is deleted from REFERENCED RELATION. e.g.; For, if we delete a row from BRANCH with BRANCH_CODE 'CS', the rows in STUDENT relation with BRANCH_CODE CS (ROLL_NO 1 and 2 in this case) will be deleted.

INTRODUCTION TO RELATIONAL MODEL

- **ANOMALIES IN THE RELATIONAL MODEL**
- **On Update Cascade**
- It will update the REFERENCING ATTRIBUTE in REFERENCING RELATION if the attribute value used by REFERENCING ATTRIBUTE is updated in REFERENCED RELATION. e.g;, if we update a row from BRANCH with BRANCH_CODE ‘CS’ to ‘CSE’, the rows in STUDENT relation with BRANCH_CODE CS (ROLL_NO 1 and 2 in this case) will be updated with BRANCH_CODE ‘CSE’.

INTRODUCTION TO RELATIONAL MODEL

- **ANOMALIES IN THE RELATIONAL MODEL**
- **Super Keys**
- Any set of attributes that allows us to identify unique rows (tuples) in a given relationship is known as super keys. Out of these super keys, we can always choose a proper subset among these that can be used as a primary key. Such keys are known as Candidate keys. If there is a combination of two or more attributes that are being used as the primary key then we call it a Composite key.

INTRODUCTION TO RELATIONAL MODEL

- **ADVANTAGES**
- **Simple model:** Relational Model is simple and easy to use in comparison to other languages.
- **Flexible:** Relational Model is more flexible than any other relational model present.
- **Secure:** Relational Model is more secure than any other relational model.
- **Data Accuracy:** Data is more accurate in the relational data model.
- **Data Integrity:** The integrity of the data is maintained in the relational model.
- **Operations can be Applied Easily:** It is better to perform operations in the relational model.

INTRODUCTION TO RELATIONAL MODEL

- **DISADVANTAGES**
- Relational Database Model is not very good for large databases.
- Sometimes, it becomes difficult to find the relation between tables.
- Because of the complex structure, the response time for queries is high.

RELATIONAL QUERY LANGUAGES

- **INTRODUCTION TO RELATIONAL ALGEBRA**
- ***Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output.***
- It uses operators to perform queries.
- An operator can be either **unary** or **binary**.
- They accept relations as their input and yield relations as their output.
- Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- These are the basic/fundamental operators used in Relational Algebra.
- **Selection(σ)**
- **Projection(π)**
- **Union(U)**
- **Set Difference(-)**
- **Set Intersection(∩)**
- **Rename(ρ)**
- **Cartesian Product(X)**

RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- **SELECTION(σ)**
- Select Operator is denoted by **sigma (σ)** and it is used to find the **tuples (or rows)** in a **relation (or table)** which satisfy the given condition.

- **Syntax**
- **σ Condition/Predicate(Relation/Table name)**

RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- **SELECTION(σ)**

Example:

A	B	C
1	2	4
2	2	3
3	2	3
4	3	4

A	B	C
1	2	4
4	3	4

For the above relation, ***σ(c>3)R*** will select the tuples which have **c** more than **3**.

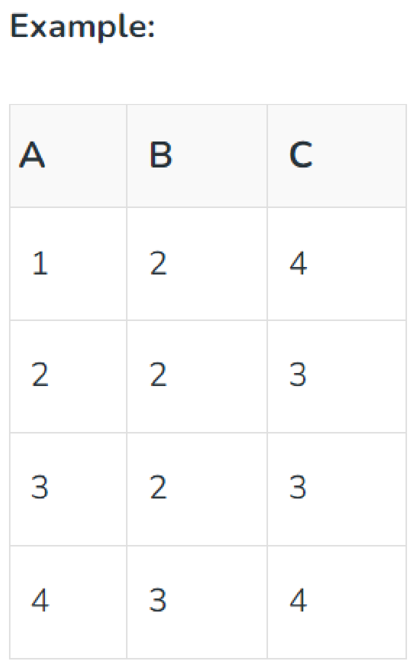
RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- **PROJECTION(π)**
- Project operator is denoted **π** by symbol and it is used to select **desired columns (or attributes)** from a **table (or relation).**

- **Syntax**
- **π column_name1,column_name2,..,column_nameN(table_name)**

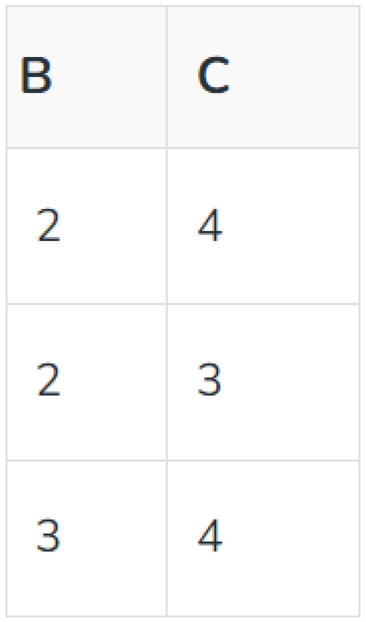
RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- **PROJECTION(π)**
- **Example:** Suppose we want columns B and C from Relation R.
- **π(B,C)R** will show following columns.



Example:

A	B	C
1	2	4
2	2	3
3	2	3
4	3	4



B	C
2	4
2	3
3	4

RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- **UNION(U)**
- Union operator is denoted by **U** symbol and it is used to select all the **rows (tuples)** from **two tables (relations).**
- Lets say we have two relations R1 and R2 both have same columns and we want to select all the tuples(rows) from these relations then we can apply the union operator on these relations.
- The rows (tuples) that are present in both the tables will only appear once in the union set. In short you can say that there are no duplicates present after the union operation.

RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- **UNION(U)**
- Syntax of Union Operator (∪)
- **table_name1 ∪ table_name2**

RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**

- **UNION(U)**

FRENCH

Student_Name	Roll_Number
Ram	01
Mohan	02
Vivek	13
Geeta	17

GERMAN

Student_Name	Roll_Number
Vivek	13
Geeta	17
Shyam	21
Rohan	25

Student_Name
Ram
Mohan
Vivek
Geeta
Shyam
Rohan

Consider the above table of Students having different optional subjects in their course.

π(Student_Name) FRENCH U π(Student_Name) GERMAN

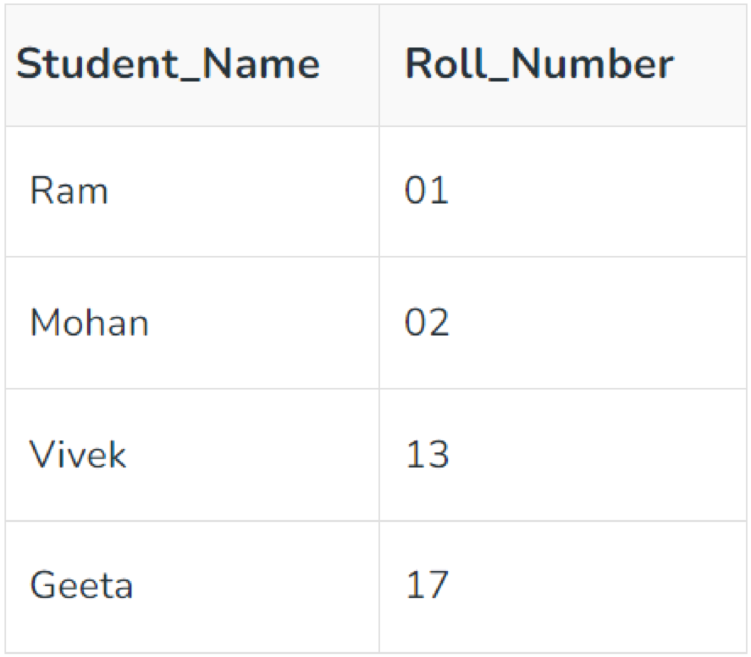
RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- **SET DIFFERENCE(-)**
- Set Difference is denoted by **–** symbol.
- Lets say we have two relations R1 and R2 and we want to select all those tuples(rows) that are present in Relation R1 but not present in Relation R2, this can be done using Set difference R1 – R2.
- **Syntax of Set Difference (-)**
- **table_name1 - table_name2**

RELATIONAL QUERY LANGUAGES

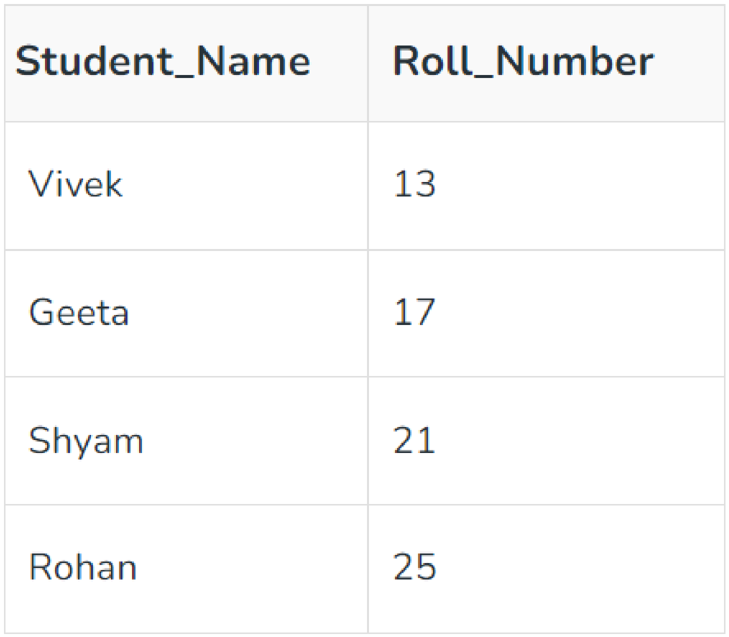
- **FUNDAMENTALS OPERATORS**
- **SET DIFFERENCE(-)**

FRENCH

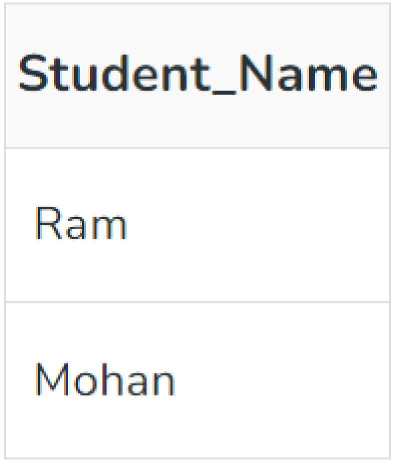


Student_Name	Roll_Number
Ram	01
Mohan	02
Vivek	13
Geeta	17

GERMAN



Student_Name	Roll_Number
Vivek	13
Geeta	17
Shyam	21
Rohan	25



Student_Name
Ram
Mohan

From the above table of **FRENCH** and **GERMAN**, Set Difference is used as follows

π(Student_Name) FRENCH - π(Student_Name) GERMAN

RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- **SET INTERSECTION(∩)**
- Intersection operator is denoted by ∩ symbol and it is used to select **common rows (tuples)** from **two tables (relations).**
- Lets say we have two relations R1 and R2 both have same columns and we want to select all those tuples(rows) that are present in both the relations, then in that case we can apply intersection operation on these two relations R1 ∩ R2.
- Only those rows that are present in both the tables will appear in the result set.

RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- **SET INTERSECTION(∩)**

FRENCH

Student_Name	Roll_Number
Ram	01
Mohan	02
Vivek	13
Geeta	17

GERMAN

Student_Name	Roll_Number
Vivek	13
Geeta	17
Shyam	21
Rohan	25

Student_Name
Vivek
Geeta

From the above table of **FRENCH** and **GERMAN**, Set Intersection is used as follows

π(Student_Name) FRENCH ∩ π(Student_Name) GERMAN

RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- **RENAME(ρ)**
- Rename (**ρ**) operator can be used to rename a **relation** or an **attribute** of a **relation**.

- **Rename (ρ) Syntax:**
- **ρ(new_relation_name, old_relation_name)R**

RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- **CARTESIAN PRODUCT(X)**
- Cartesian Product is denoted by X symbol.
- Lets say we have two relations R1 and R2 then the cartesian product of these two relations (R1 X R2) would combine each tuple of first relation R1 with the each tuple of second relation R2.
- **Syntax of Cartesian product (X)**
- **R1 X R2**

RELATIONAL QUERY LANGUAGES

- **FUNDAMENTALS OPERATORS**
- **CARTESIAN PRODUCT(X)**

A

Name	Age	Sex
Ram	14	M
Sona	15	F
Kim	20	M

B

ID	Course
1	DS
2	DBMS

A X B

Name	Age	Sex	ID	Course
Ram	14	M	1	DS
Ram	14	M	2	DBMS
Sona	15	F	1	DS
Sona	15	F	2	DBMS
Kim	20	M	1	DS
Kim	20	M	2	DBMS

RELATIONAL QUERY LANGUAGES

- **DERIVED OPERATORS**
- These are some of the derived operators, which are derived from the fundamental operators.
- **Natural Join(⋈)**
- **Conditional Join**

RELATIONAL QUERY LANGUAGES

- **DERIVED OPERATORS**
- **NATURAL JOIN(⋈)**
- Natural join is a **binary operator**.
- Natural join between two or more relations will result in a set of all combinations of tuples where they have an equal common attribute.
- **Syntax:**
- **table1_relation ⋈ table2_relation**

RELATIONAL QUERY LANGUAGES

- **DERIVED OPERATORS**
- **NATURAL JOIN(⋈)**

EMP

Name	ID	Dept_Name
A	120	IT
B	125	HR
C	110	Sales
D	111	IT

DEPT

Dept_Name	Manager
Sales	Y
Production	Z
IT	A

EMP ⋈ DEPT

Name	ID	Dept_Name	Manager
A	120	IT	A
C	110	Sales	Y
D	111	IT	A

RELATIONAL QUERY LANGUAGES

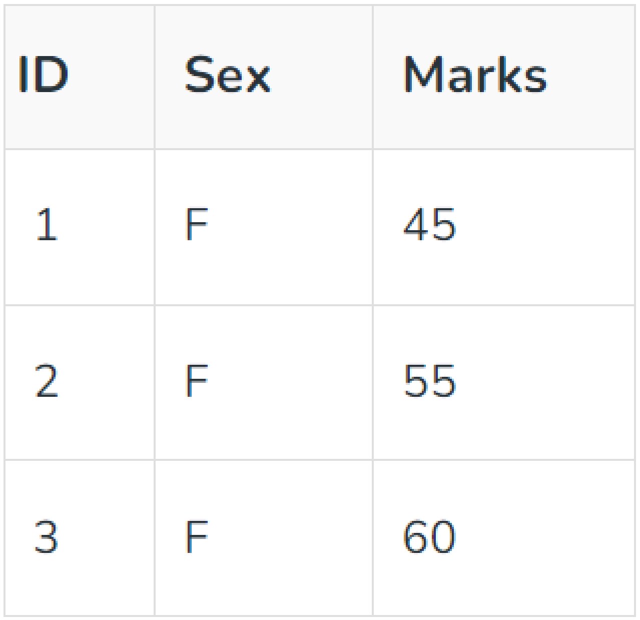
- **DERIVED OPERATORS**
- **CONDITIONAL JOIN**
- Conditional join works similarly to natural join.

- In natural join, by default condition is equal between common attributes while in conditional join we can specify any condition such as greater than, less than, or not equal.

RELATIONAL QUERY LANGUAGES

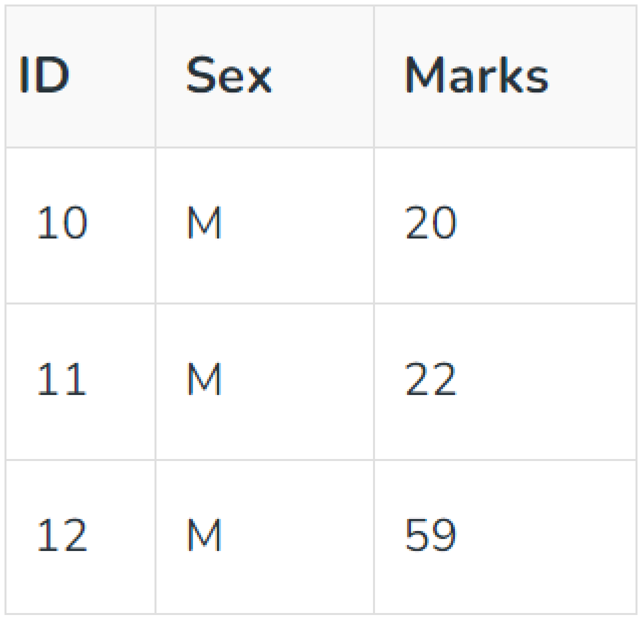
- **DERIVED OPERATORS**
- **CONDITIONAL JOIN**

R



ID	Sex	Marks
1	F	45
2	F	55
3	F	60

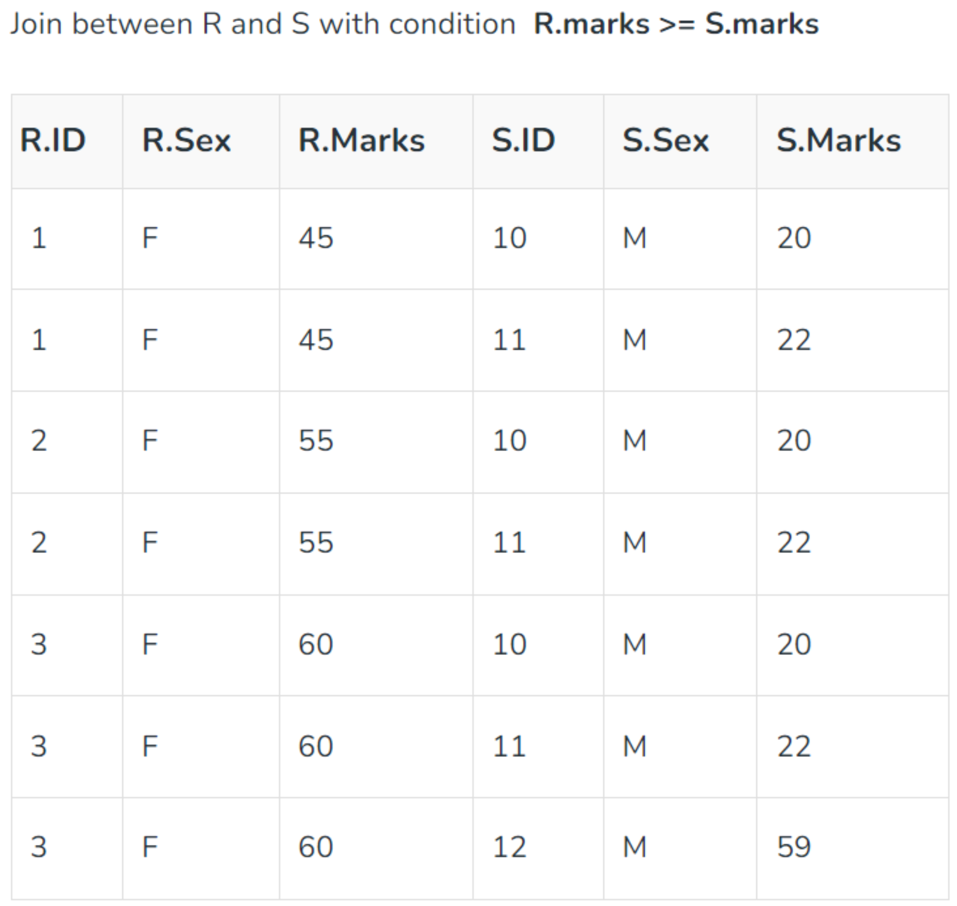
S



ID	Sex	Marks
10	M	20
11	M	22
12	M	59

RELATIONAL QUERY LANGUAGES

- **DERIVED OPERATORS**
- **CONDITIONAL JOIN**



Join between R and S with condition **R.marks >= S.marks**

R.ID	R.Sex	R.Marks	S.ID	S.Sex	S.Marks
1	F	45	10	M	20
1	F	45	11	M	22
2	F	55	10	M	20
2	F	55	11	M	22
3	F	60	10	M	20
3	F	60	11	M	22
3	F	60	12	M	59