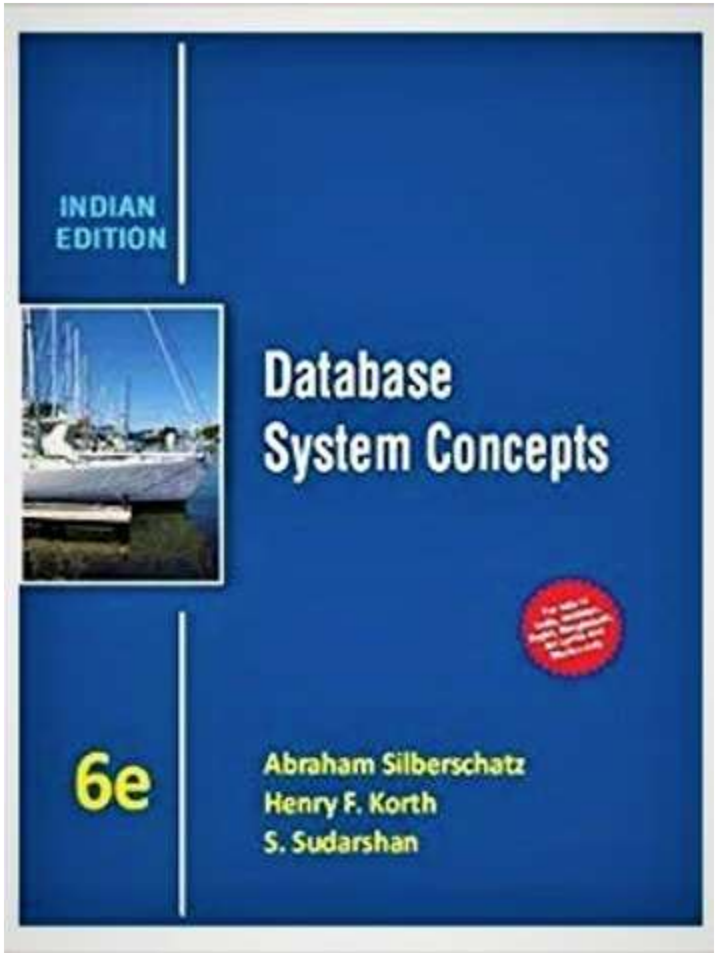
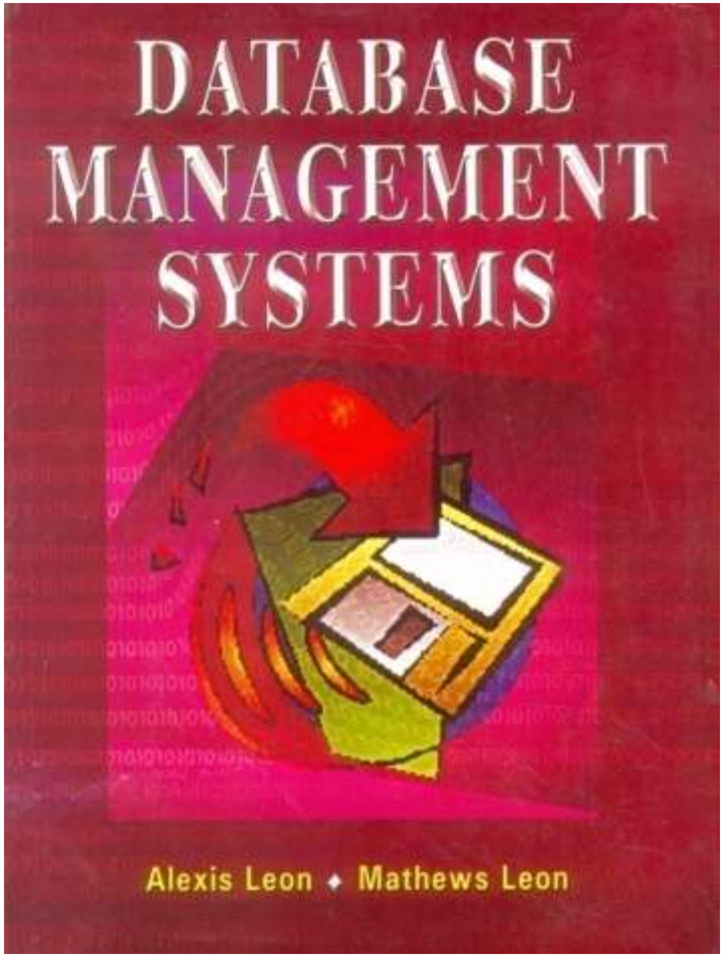
UNIT–I

Introduction: Concept & Overview of DBMS, Three Schema Architecture of DBMS, Database Approach v/s Traditional File Accessing Approach, Advantages of Database Systems, Data Models, Schema and Instances, Data Independence, Data Base Language and Interfaces, Overall Database Structure, Functions of DBA and Designer, Database Users. Entity-Relationship Model: Basic concepts, Design Issues, Mapping Constraints, Keys, EntityRelationship Diagram, Weak Entity Sets and Extended E-R features.ER Diagram to Relational Table conversion.

Reference book





Introduction

- A software package/ system to facilitate the creation and maintenance of a computerized database.
- A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.
- DBMS contains information about a particular enterprise
- A modern database system is a complex software system whose task is to manage a large, complex collection of data.

Database Applications Examples

- **Enterprise Information**
  - Sales: customers, products, purchases
  - Accounting: payments, receipts, assets
  - Human Resources: Information about employees, salaries, payroll taxes.
- **Banking and finance**
  - customer information, accounts, loans, and banking transactions.
  - Credit card transactions
  - Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data
- **Airlines:** reservations, schedules
- **Telecommunication:** records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards
- **Universities:** registration, grades, students record

Characteristics of the Database Approach

- **Self-describing nature of a database system:** A DBMS **catalog** stores the *description* of the database. The description is called **meta-data**). This allows the DBMS software to work with different databases.
- **Data Abstraction:** A data model is used to hide storage details and present the users with a *conceptual view* of the database.
- **Support of multiple views of the data:** Each user may see a different view of the database, which describes *only* the data of interest to that user
- **Sharing of data and multiuser transaction processing :** allowing a set of concurrent users to retrieve and to update the database. Concurrency control within the DBMS guarantees that each **transaction** is correctly executed

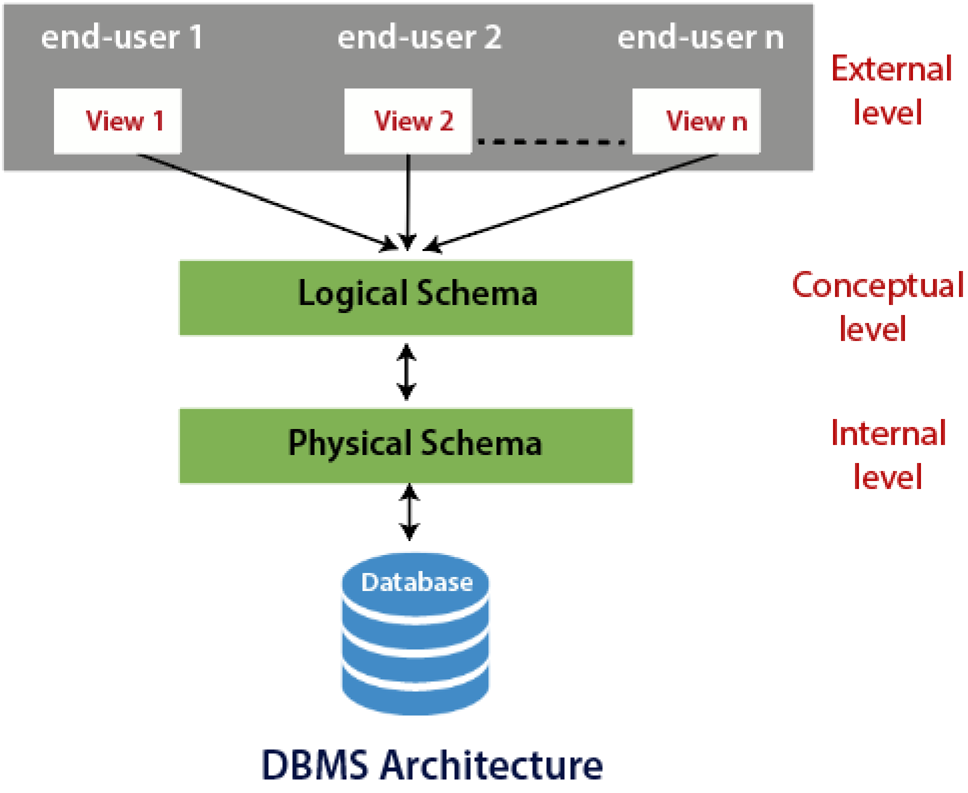
Advantages of Using the Database Approach

- **No redundant data**: Redundancy removed by data normalization. No data duplication saves storage and improves access time.
- **Data Consistency and Integrity**: As we discussed earlier the root cause of data inconsistency is data redundancy, since data normalization takes care of the data redundancy, data inconsistency also been taken care of as part of it
- **Data Security**: It is easier to apply access constraints in database systems so that only authorized user is able to access the data. Each user has a different set of access thus data is secured from the issues such as identity theft, data leaks and misuse of data.
- **Privacy**: Limited access means privacy of data.
- **Easy access to data** – Database systems manages data in such a way so that the data is easily accessible with fast response times.
- **Easy recovery**: Since database systems keeps the backup of data, it is easier to do a full recovery of data in case of a failure.
- **Flexible**: Database systems are more flexible than file processing systems.

Disadvantages of DBMS

- **Cost of Hardware and Software :-**Database systems require sophisticated hardware and software and highly skilled personnel. Training, licensing, and regulation compliance costs are often overlooked when database systems are implemented.
- **Complexity:-**Database systems interface with many different technologies and have a significant impact on a company's resources and culture. Database systems are complex to understand.
- **Frequency Upgrade:-**DBMS vendors frequently upgrade their products by adding new functionality. Such new features often come bundled in new upgrade versions of the software.
- **Currency Maintenance:-**To maximize the efficiency of the database system, you must keep your system current.

Three Schema Architecture of DBMS



1. **External level:-** It is also called **view level**. The reason this level is called "view" is because several users can view their desired data from this level which is internally fetched from database with the help of conceptual and internal level mapping.

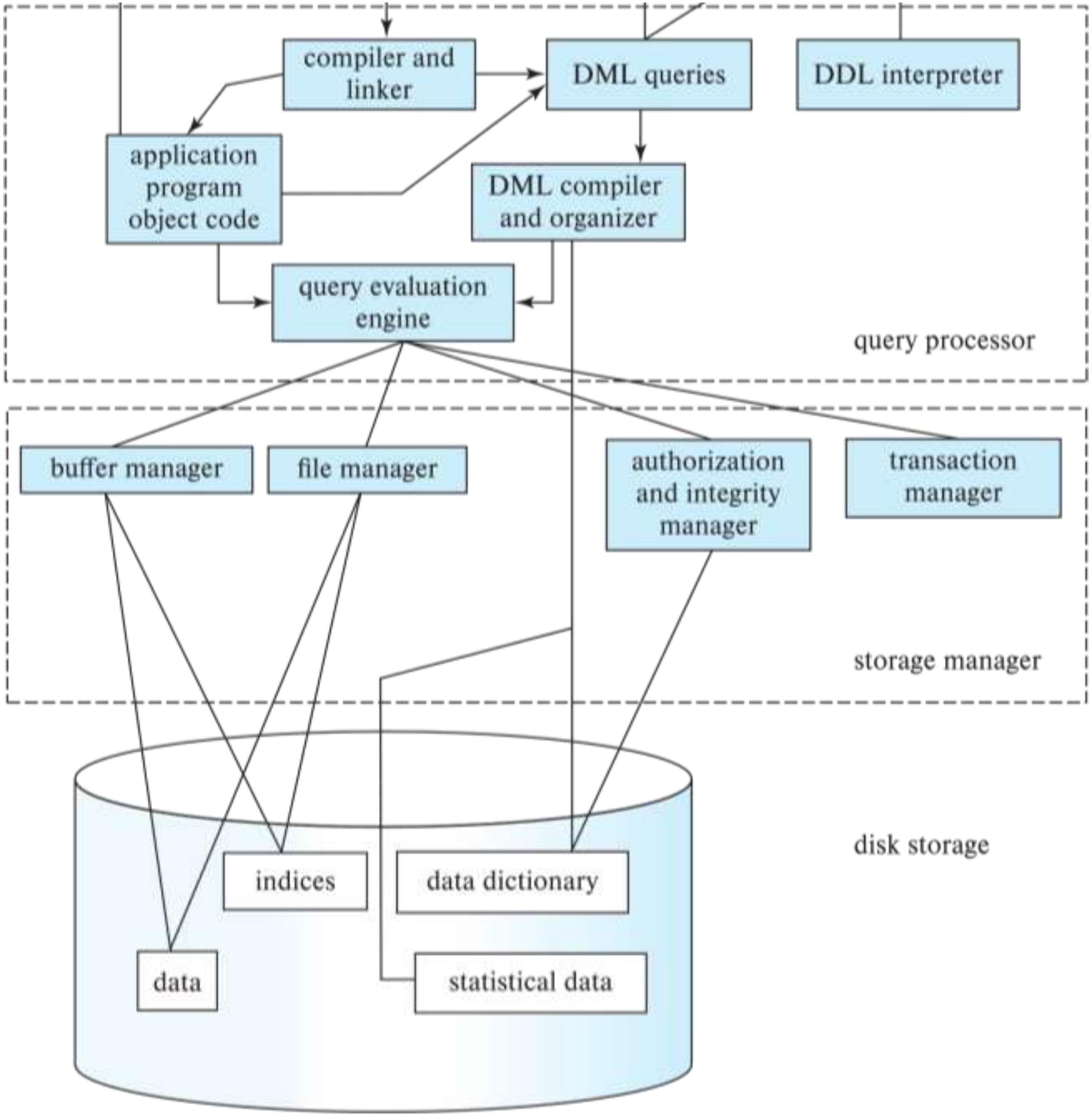
   The view schema describes the end user interaction with database systems.

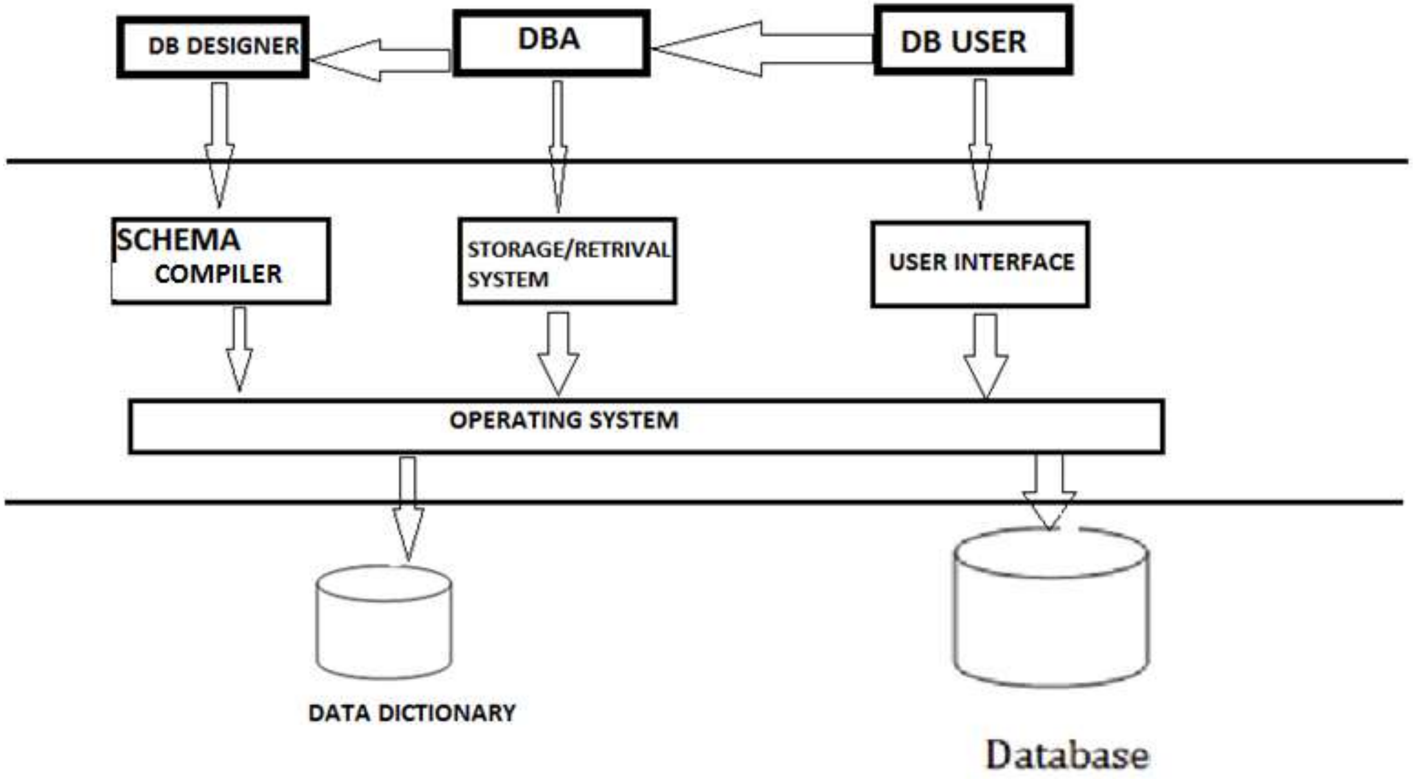
2. **Conceptual level:-**It is also called **logical level**. The whole design of the database such as relationship among data, schema of data etc. are described in this level.

   Programmers and database administrators work at this level.

3. **Internal level:-** This level is also known as physical level. This level describes how the data is actually stored in the storage devices. This level is also responsible for allocating space to the data. This is the lowest level of the architecture.

   The physical level is used to describe complex low-level data structures in detail.


compiler and linker
DML queries
DDL interpreter
application program object code
DML compiler and organizer
query evaluation engine
query processor
buffer manager
file manager
authorization and integrity manager
transaction manager
storage manager
disk storage
indices
data dictionary
data
statistical data


DB DESIGNER
DBA
DB USER
SCHEMA COMPILER
STORAGE/RETRIVAL SYSTEM
USER INTERFACE
OPERATING SYSTEM
DATA DICTIONARY
Database

DB designer:- who implements specific application programs to access the stored data .Determine the requirement of end user , and develop specifications to meet these requirements. The database designer role defines the tables, indexes, views, constraints, triggers, stored procedures, table spaces and storage parameters.

DB user:- Accesses an existing application program to perform daily tasks. A person who ultimately uses or is intended to ultimately use a product. The end user is usually not concerned about the transaction or operations done at various levels.

DBA:- Responsible for authorizing access to the database, monitoring its use and managing all the resources to support the use of the entire database system.

Functions of a **DBA**

1. Schema Definition:

- The DBA definition the logical Schema of the database.A Schema refers to the overall logical structure of the database.
- According to this schema, database will be developed to store required data for an organization.

2. Storage Structure and Access Method Definition:

- The DBA decides how the data is to be represented in the stored database.

3. Assisting Application Programmers:

- The DBA provides assistance to application programmers to develop application programs.

4. Physical Organization Modification:

- The DBA modifies the physical organization of the database to reflex the changing needs of the organization or to improve performance.

5. Approving Data Access:

- The DBA determines which user needs access to which part of the database.
- According to this, various types of authorizations are granted to different users.

6. Monitoring Performance:

- The DBA monitors performance of the system.The DBA ensures that better performance is maintained by making changes in physical or logical schema if required.

7. Backup and Recovery:

- Database should not be lost or damaged.
- The DBA ensures this periodically backing up the database on magnetic tapes or remote servers.
- In case of failure, such as virus attack database is recovered from this backup.

- **Schema:**The overall logical design of the database.

- **Data dictionary:- I**t contains metadata i.e data about the database. The data dictionary is very important as it contains information such as what is in the database, who is allowed to access it, where is the database physically stored etc. The users of the database normally don't interact with the data dictionary, it is only handled by the database administrators.
- **The data dictionary in general contains information about the following −**

➢ Names of all the database tables and their schemas.

➢ Details about all the tables in the database, such as their owners, their security constraints, when they were created etc.

➢ Physical information about the tables such as where they are stored and how.

➢ Table constraints such as primary key attributes, foreign key information etc.

Difference between DBMS and Traditional File System

- DBMS is very expensive but, the traditional file system is cheap.
- DBMS is good for the large system but, the traditional file system is good for a small system having a small number of items.
- DBMS required lots of effort for designing but, the traditional file system is very low design efforts.
- DBMS is highly secured but, the traditional file system is not secure.
- DBMS is data sharable but, the traditional file system is isolated data sharable.
- DBMS is flexible but, the traditional file system has a lack of flexibility and has many limitations.
- DBMS has no integrity but, the traditional file system has an integrity problem.
- DBMS has a complex backup system but, the traditional file system has a simple backup system.
- DBMS removed data redundancy but, the traditional file system has data redundancy.

Data Models

Data Model is a logical structure of Database. It describes the design of database to reflect entities, attributes, relationship among data, constrains etc.

Types of Data Models

- **Object based logical Models** – Describe data at the conceptual and view levels.

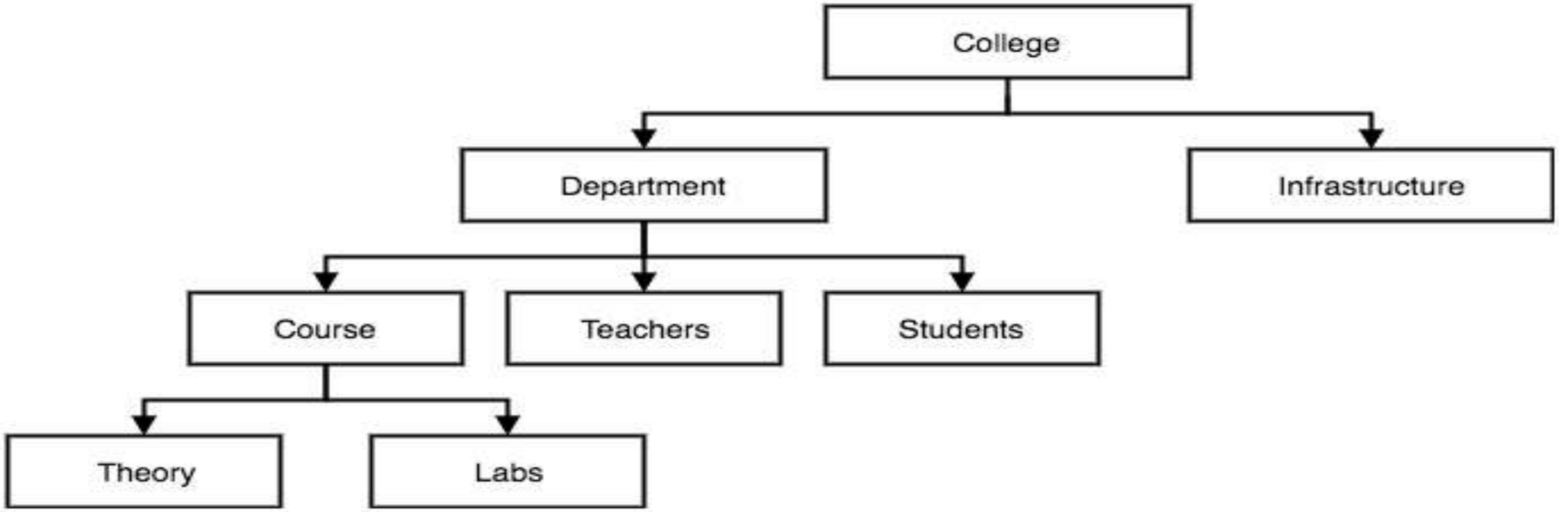
1. E-R Model
2. Object oriented Model

- **Record based logical Models** – Like Object based model, they also describe data at the conceptual and view levels. These models specify logical structure of database with records, fields and attributes.

1. Relational Model
2. Hierarchical Model
3. Network Model

Hierarchical Model

- This database model organises data into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.
- In hierarchical model, data is organised into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.

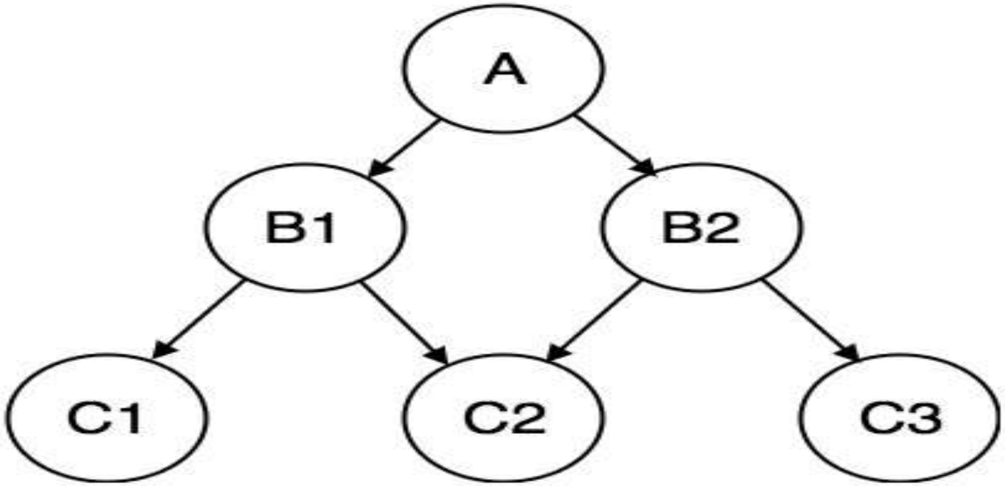


Advantages and disadvantages

- **Advantages of the hierarchical model**
- It promotes data sharing.
- There is a parent/child relationship due to which its concepts are simple.
- It provides database security.
- It takes 1 to many relationships.
- **Disadvantages of the hierarchical model**
- It is not flexible
- It does not have a data definition and data manipulation languages.
- It requires knowledge of physical data storage for complex implementation.

Network Model

- This is an extension of the Hierarchical model. In this model data is organised more like a graph, and are allowed to have more than one parent node.
- In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships.
- This was the most widely used database model, before Relational Model was introduced.



Advantages and disadvantages

Advantages of a network model

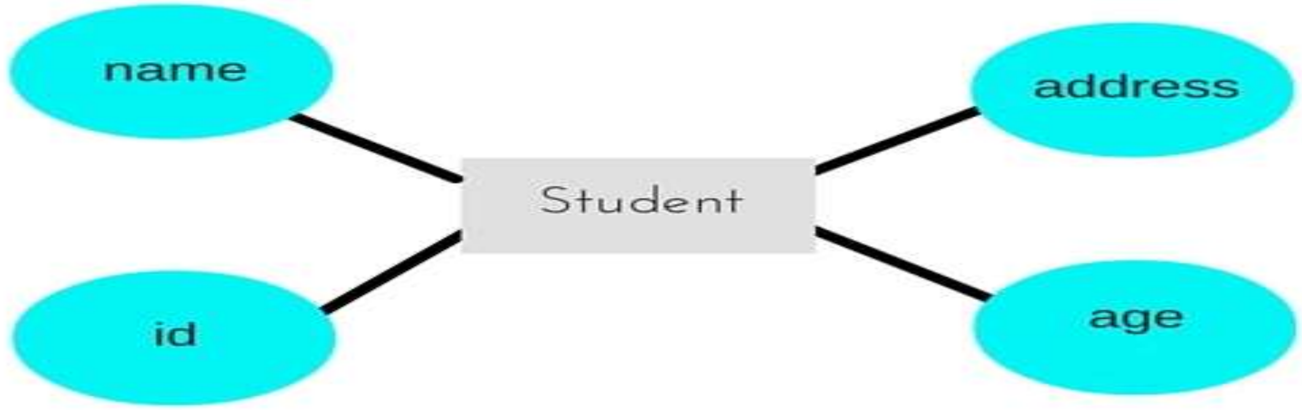
- Its concept is as simple as the hierarchical model.
- There is more than one parent/child relationship.
- Data can be accessed easily in it.
- It provides data integrity.
- It contains a data definition language (DDL) and data manipulation language (DML).

Disadvantages of the network model

- Its database structure is very complex (difficult) because all the records in it are maintained using pointers.
- Changes in its structure require changes in all **programs**.

Entity-relationship Model

- In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.
- Different entities are related using relationships.
- E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.
- This model is good to design a database, which can then be turned into tables in relational model.
- Let's take an example, If we have to design a School Database, then **Student** will be an **entity** with **attributes** name, age, address etc. As **Address** is generally complex, it can be another **entity** with **attributes** street name, pincode, city etc, and there will be a relationship between them.



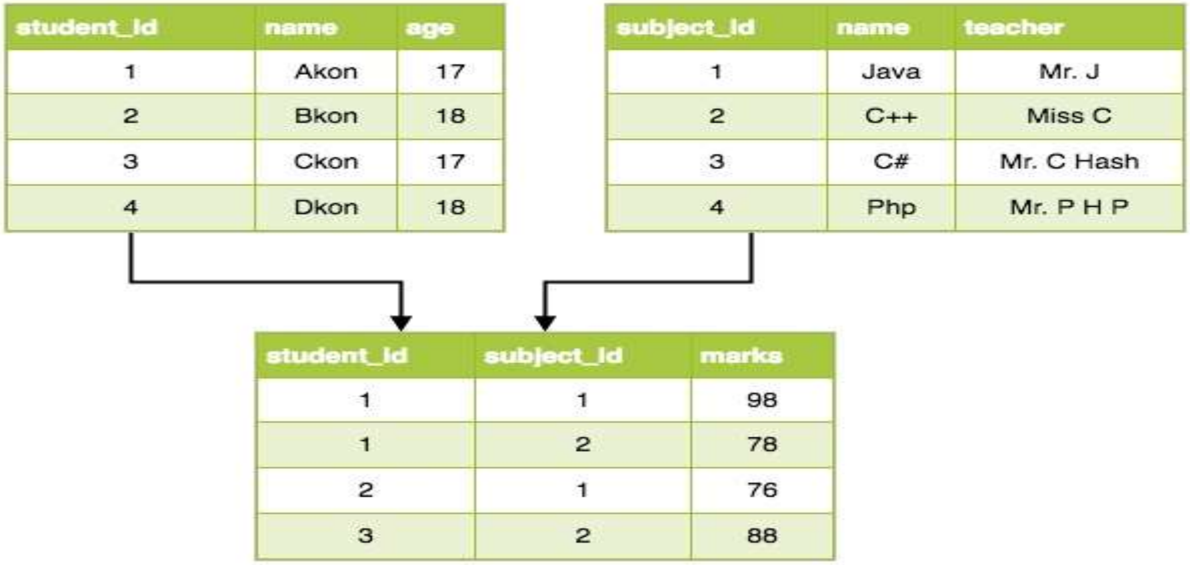
Advantages and disadvantages

Advantages of the E-R model

- The E-R model is very simple if we know the relationship between entities and attributes.
- This model is presented as a diagram. With which we can understand easily.
- There is no data manipulation.
- Its design is of a high level.
- **Disadvantages of the E-R model**
- There is limited relationship representation.
- There is no data manipulation language.

Relational Model

- In this model, data is organised in two-dimensional **tables** and the relationship is maintained by storing a common field.
- This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model. we can say the only database model used around the world.
- The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.
- Hence, tables are also known as **relations** in relational model.



student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

Advantages and disadvantages

Advantages of the relational model

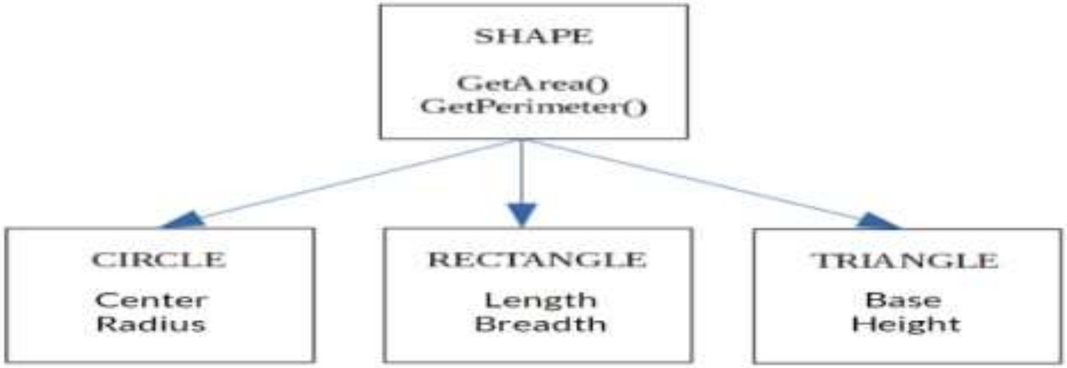
- It is very flexible, it can easily make any kind of change.
- In this, the data is kept in tables, so its concept is very simple.
- It provides data integrity. That is, no user can access the database without the owner’s permission.

Disadvantages of the relational model

- It requires powerful hardware computers, storage devices, and software.
- It is very easy to use but when a user stores data in it incorrectly then it becomes very bad DBMS.
- This is a very simple model, due to its simplicity, some users create their own database, causing the problem of data inconsistency, data duplication.

Object oriented Model

- In this both data and their relationship are organised or contained in a single structure known as object.
- Object includes information about relationship between the facts within the object,as well as information about relationship with other objects.
- It is also said to be semantic data model.
- An object is the abstraction of the real world entity and an object represents only one occurrence of entity.
- Attributes: It describes the property of an object.
- For example:-
- Shape, Circle, Rectangle and Triangle are all objects in this model. Circle has the attributes Center and Radius.Rectangle has the attributes Length and Breath. Triangle has the attributes Base and Height . The objects Circle, Rectangle and Triangle inherit from the object Shape.



Advantages and disadvantages

Advantages of the object-oriented model

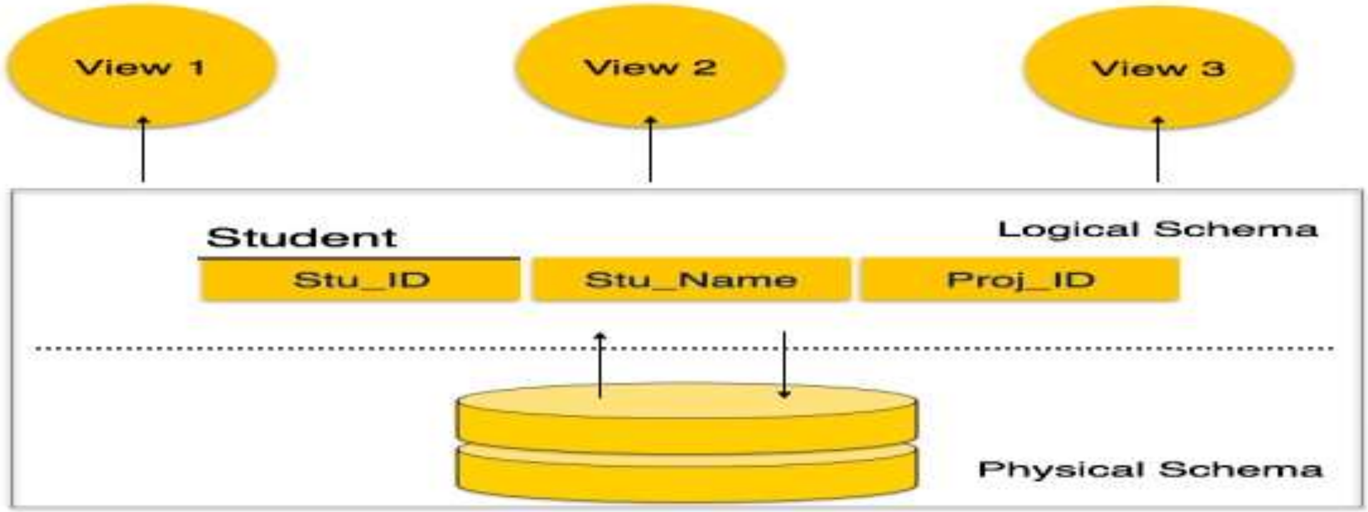
- It supports inheritance which increases data integrity.
- It improves performance.

Disadvantages of the object-oriented model

- It requires a powerful system due to which the transaction is very slow.
- It is a very complex model.
- To use it, one has to learn it first.
- There is very little security in it.

Database Schema

- A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated.
- A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database. It’s the database designers who design the schema to help programmers understand the database and make it useful.
- A database schema can be divided broadly into two categories −
- **Physical Database Schema** − This schema related to the actual storage of data and its form of storage like files, etc. It defines how the data will be stored in a secondary storage.
- **Logical Database Schema** − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.



Database Instance

- It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.
- A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by particularly following all the validations, constraints, and conditions that the database designers have imposed.

Data Independence

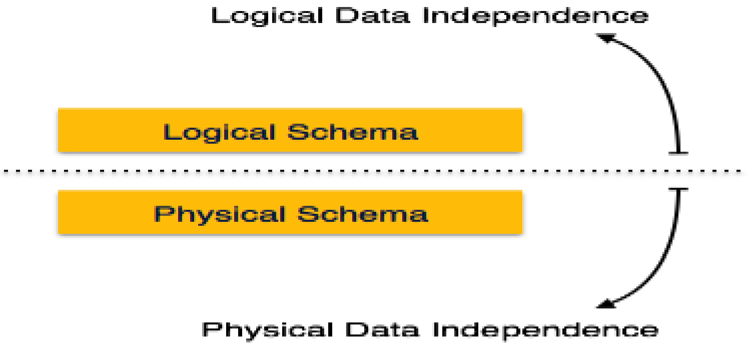
- A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a highly complex job.
- Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.

Logical Data Independence

- Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.

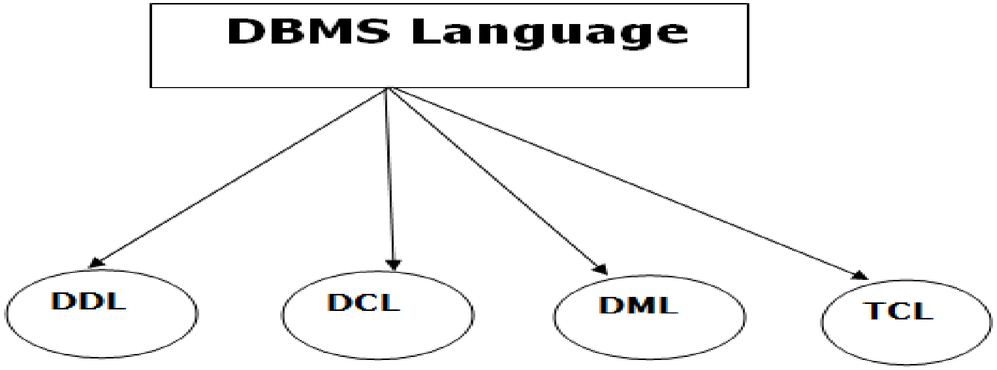
Physical Data Independence

- All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.
- For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.



Database Language

- A DBMS has appropriate languages and interfaces to express database queries and updates.
- Database languages can be used to read, store and update the data in the database.
- Types of Database Language



DDL(Data Definition Language)

- **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
- It is used to create schema, tables, indexes, constraints, etc. in the database.
- Using the DDL statements, you can create the skeleton of the database.
- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Some DDL commands :-

- **Create:** It is used to create objects in the database.
- **Alter:** It is used to alter the structure of the database.
- **Drop:** It is used to delete objects from the database.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.

DML(Data Manipulation Language)

- **DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.DML also known as query language

some tasks that come under DML:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.

DCL(Data Control Language)

- **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
- The DCL execution is transactional. It also has rollback parameters.

some tasks that come under DCL:

- **Grant:** It is used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user.

TCL(Transaction Control Language)

- TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

some tasks that come under TCL:

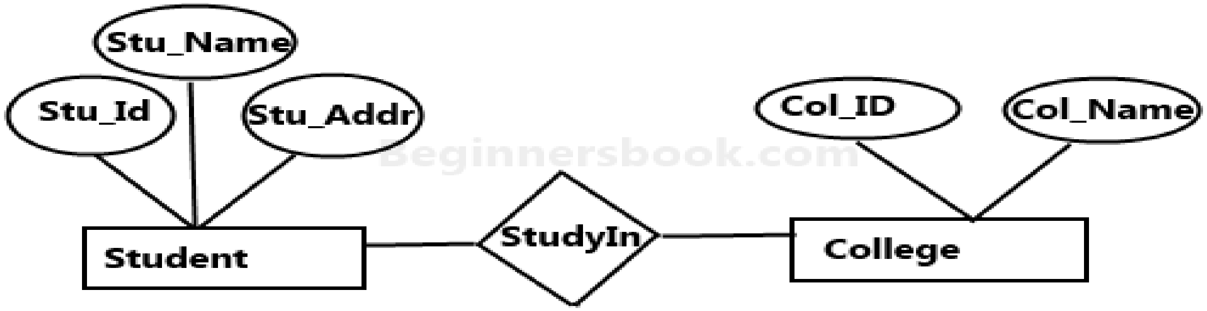
- **Commit:** It is used to save the transaction on the database.
- **Rollback:** It is used to restore the database to original since the last Commit.

Interfaces in DBMS

- A database management system (DBMS) interface is a user interface which allows for the ability to input queries to a database without using the query language itself.
- User-friendly interfaces provide by DBMS may include the following:
- **Menu-Based Interfaces for Web Clients or Browsing .**
- Forms-Based Interfaces.
- Graphical User Interfaces.
- Natural Language Interfaces.
- Interfaces for the DBA.

ER model

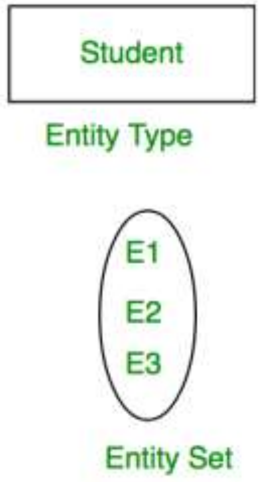
- An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.
- In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time. Student entity has attributes such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as Col_ID & Col_Name.



Sample E-R Diagram

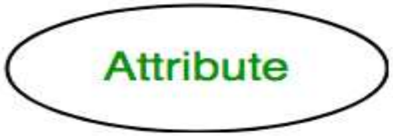
Entity

- 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.
- **Representation:- Rectangle**
- An Entity is an object of Entity Type and set of all entities is called as entity set. e.g.; E1 is an entity having Entity Type Student and set of all students is called Entity Set. In ER diagram, Entity Type is represented as

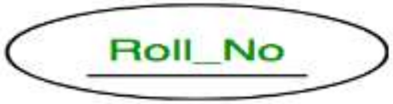


Attribute

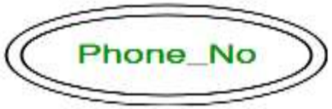
- Attributes are the **properties which define the entity type**. For example, Roll_No, Name, DOB, Age, Address, Mobile_No are the attributes which defines entity type Student.
- **Representation:- oval**



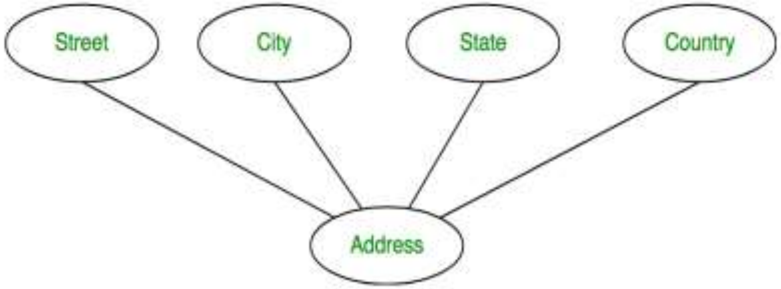
- **Key Attribute –**
  The attribute which **uniquely identifies each entity** in the entity set is called key attribute. For example, Roll_No will be unique for each student.
- **Representation:- oval with underlying lines.**



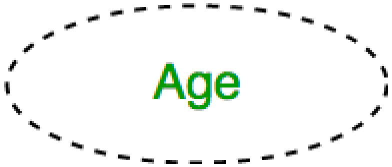
- **Multivalued Attribute –**
  An attribute consisting **more than one value** for a given entity. For example, Phone_No (can be more than one for a given student)
- **Representation:- double oval**



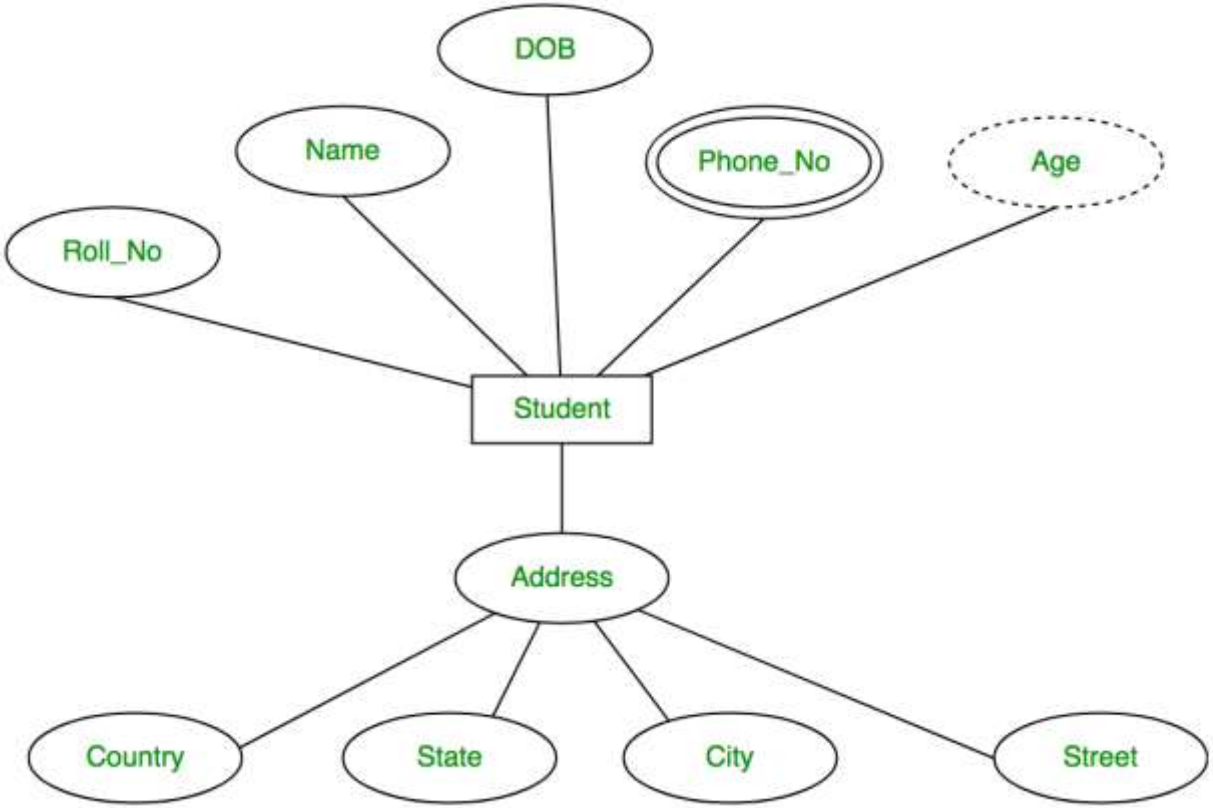
- **Composite Attribute –**
  An attribute **composed of many other attribute** is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country.
- **Representation:- oval comprising of ovals**



- **Derived Attribute –**
  An attribute which can be **derived from other attributes** of the entity type is known as derived attribute. e.g.; Age (can be derived from DOB).
- **Representation:- dashed oval**.

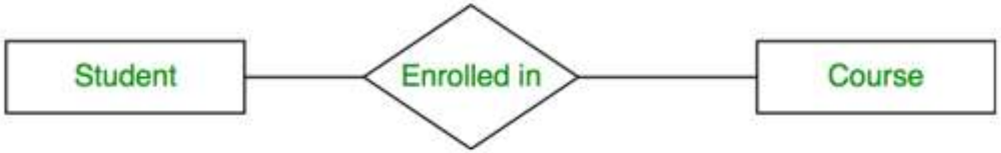


- The complete entity type **Student** with its attributes can be represented as:

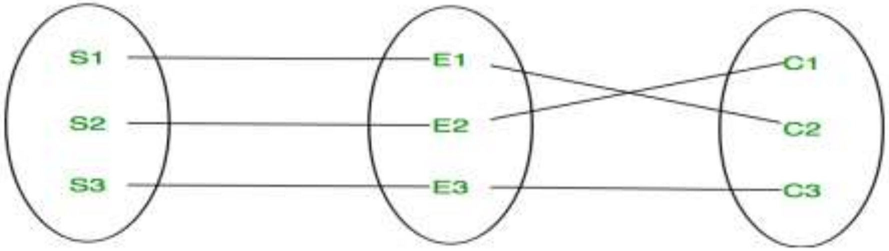


Relationship Type and Relationship Set:

- 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.
- represented :- diamond and connecting the entities with lines.

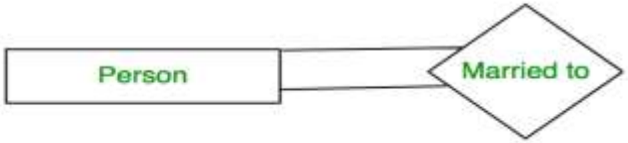


- A set of relationships of same type is known as relationship set. The following relationship set depicts S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled in C3.

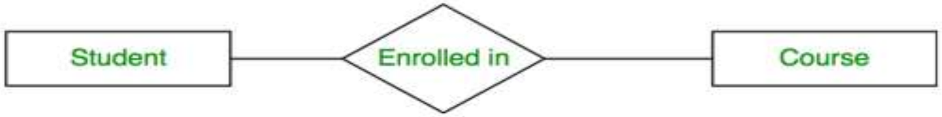


Degree of a relationship set

- The number of different entity sets **participating in a relationship** set is called as degree of a relationship set.
- **Unary Relationship –**
  When there is **only ONE entity set participating in a relation**, the relationship is called as unary relationship. For example, one person is married to only one person.

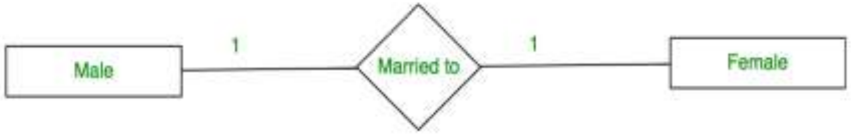


- **Binary Relationship –**
  When there are **TWO entities set participating in a relation**, the relationship is called as binary relationship.For example, Student is enrolled in Course.

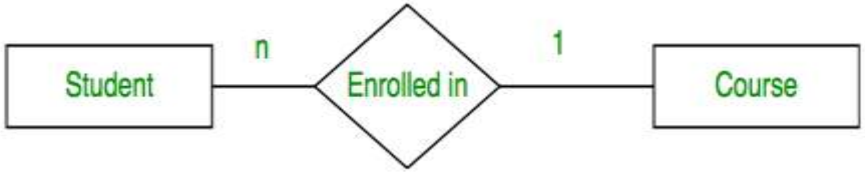


Cardinality:

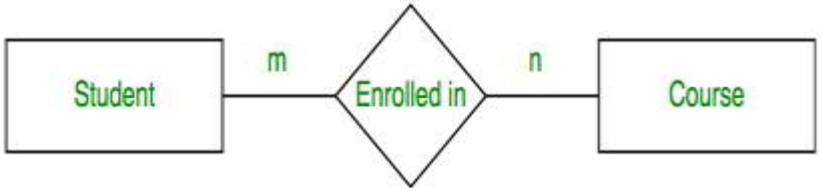
- The **number of times an entity of an entity set participates in a relationship** set is known as cardinality. Cardinality can be of different types:
- **One to one –** When each entity in each entity set can take part **only once in the relationship**, the cardinality is one to one. Let us assume that a male can married to one female and a female can married to one male. So the relationship will be one to one.



- **Many to one –** When entities in one entity set **can take part only once in the relationship set and entities in other entity set 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.

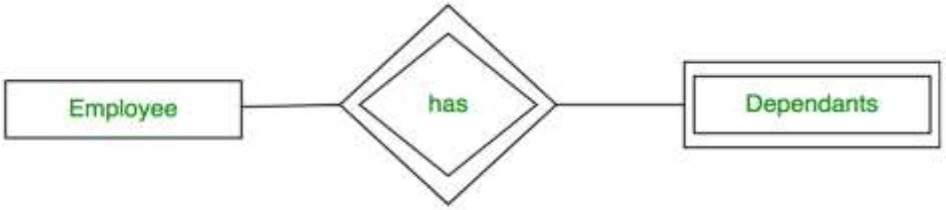


- **Many to many –** When entities in all entity sets can **take part more than once in the relationship** cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.



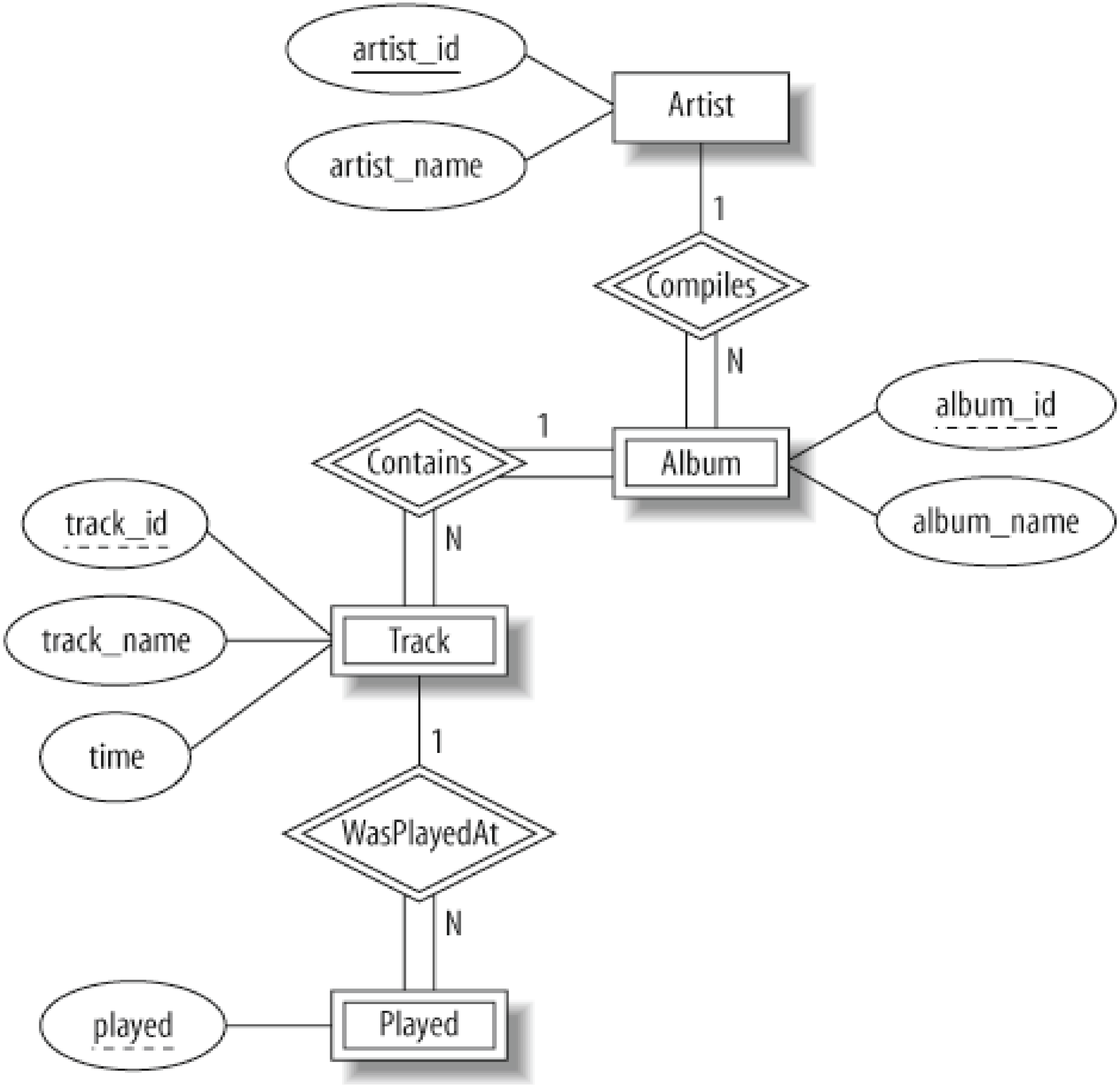
Weak Entity Type and Identifying Relationship:

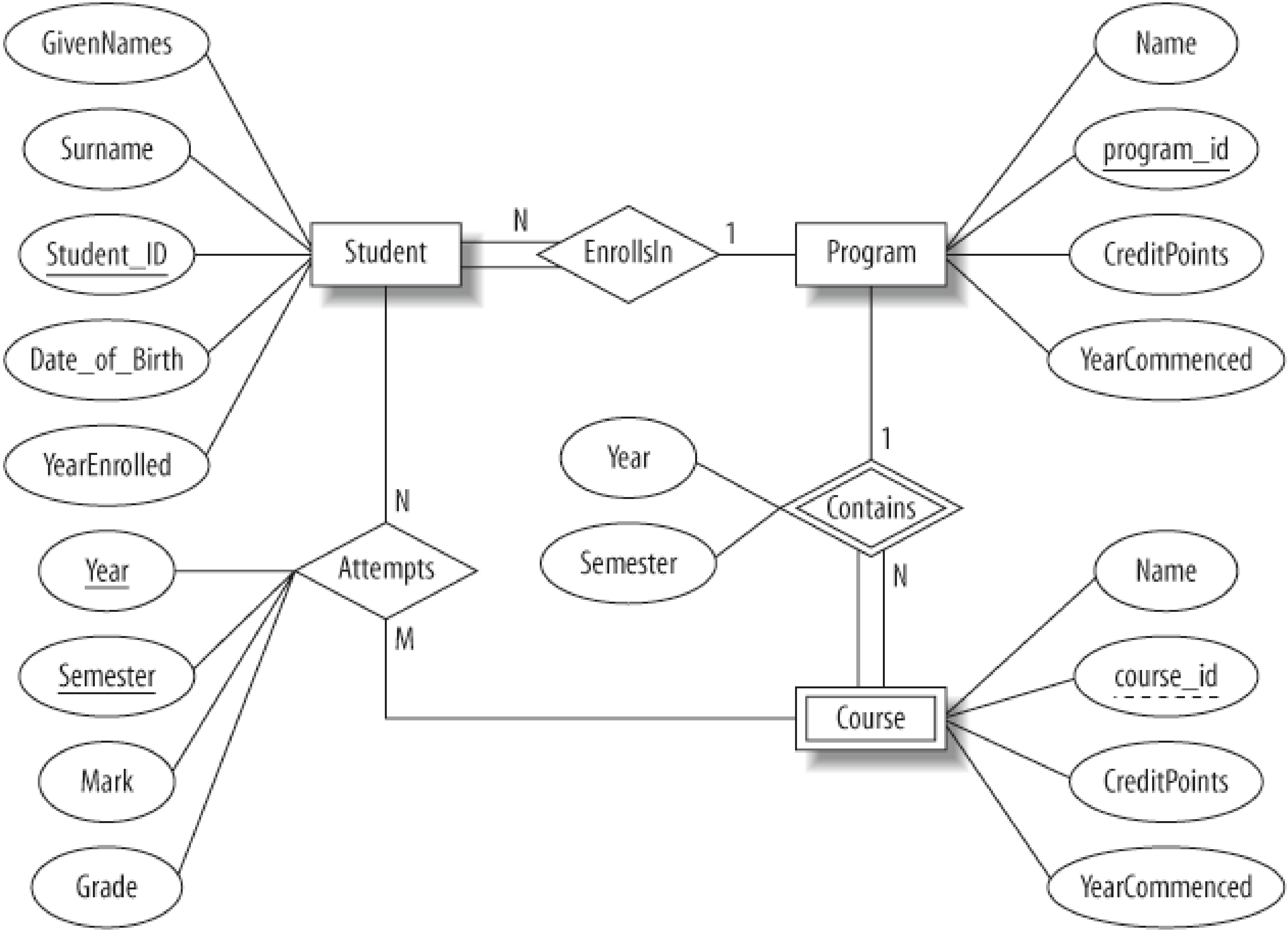
- As discussed before, an entity type has a key attribute which uniquely identifies each entity in the entity set. But there exists **some entity type for which key attribute can’t be defined**. These are called Weak Entity type.
- For example, A company may store the information of dependants (Parents, Children, Spouse) of an Employee. But the dependents don’t have existence without the employee. So Dependent will be weak entity type and Employee will be Identifying Entity type for Dependant.
- A weak entity type is represented by a double rectangle. The participation of weak entity type is always total. The relationship between weak entity type and its identifying strong entity type is called identifying relationship and it is represented by double diamond.

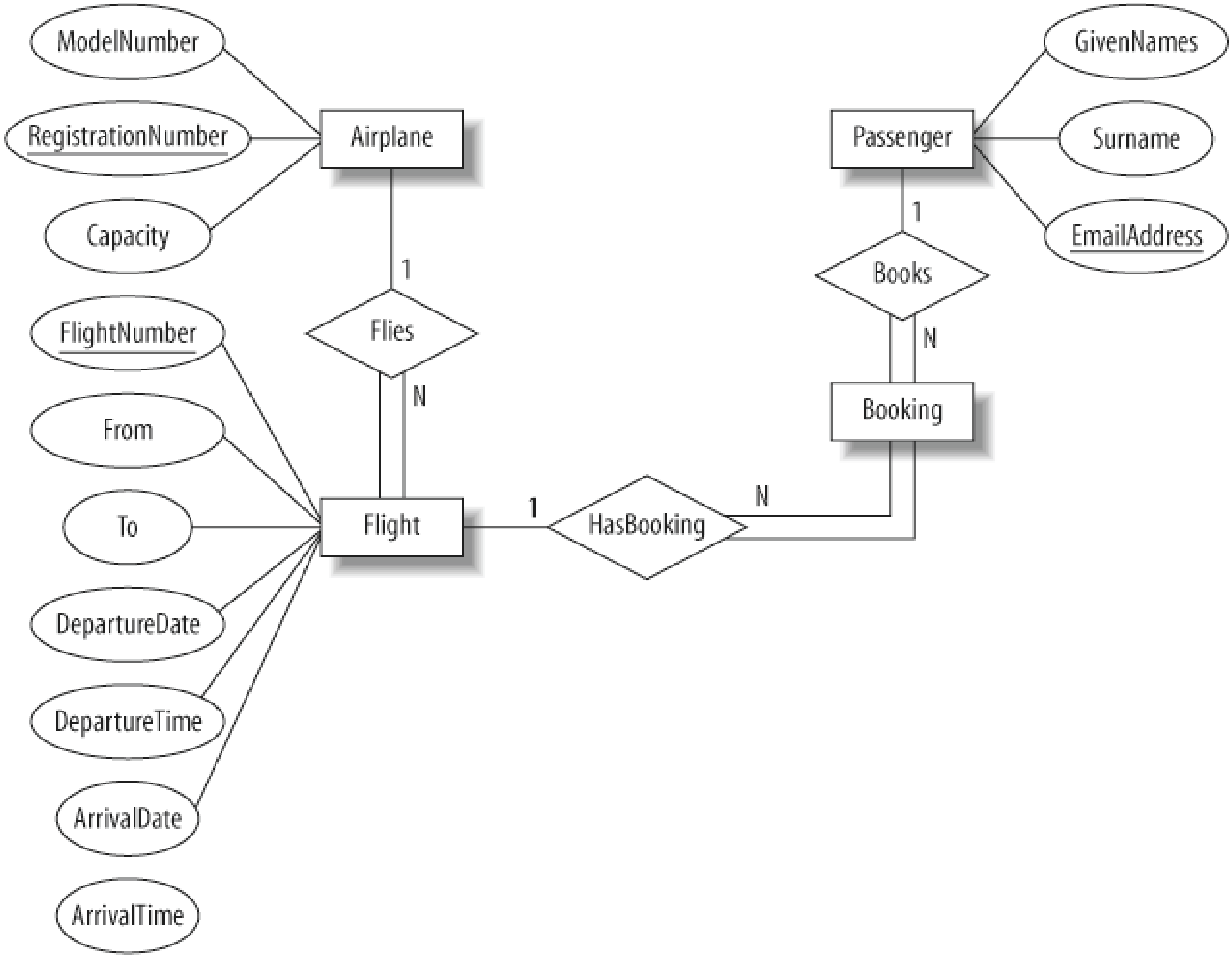


ER examples

- The music database is designed to store details of a music collection, including the albums in the collection, the artists who made them, the tracks on the albums, and when each track was last played.
- The university database captures the details of students, courses, and grades for a university.
- The flight database stores an airline timetable of flight routes, times, and the plane types.


artist_id
artist_name
Artist
1
Compiles
N
album_id
album_name
Album
1
Contains
N
track_id
track_name
time
Track
1
WasPlayedAt
N
played
Played


GivenNames
Surname
Student_ID
Date_of_Birth
YearEnrolled
Student
N
EnrollsIn
1
Program
Name
program_id
CreditPoints
YearCommenced
N
Attempts
M
Year
Semester
Mark
Grade
1
Contains
N
Year
Semester
Course
Name
course_id
CreditPoints
YearCommenced

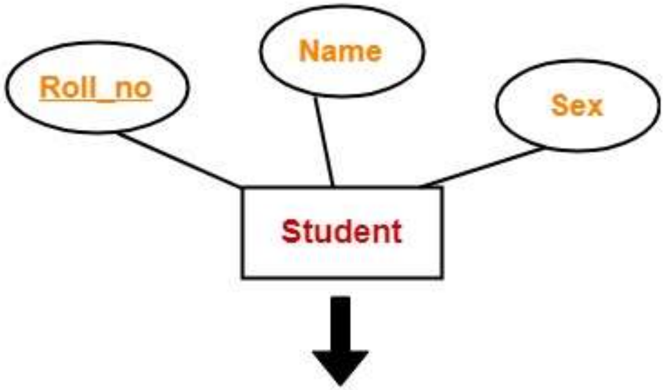

ModelNumber
RegistrationNumber
Capacity
Airplane
1
Flies
N
FlightNumber
From
To
DepartureDate
DepartureTime
ArrivalDate
ArrivalTime
Flight
1
HasBooking
N
Booking
N
Books
1
Passenger
GivenNames
Surname
EmailAddress

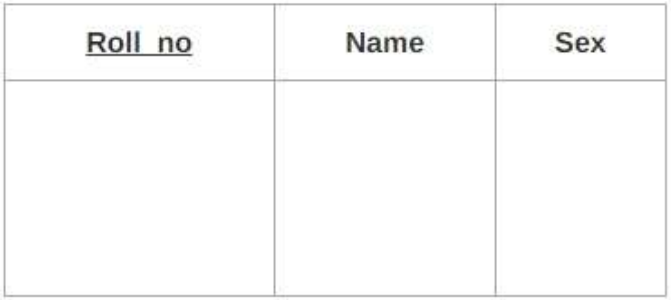
Converting ER Diagrams to Tables-

- Following rules are used for converting an ER diagram into the tables-

Rule-01: For Strong Entity Set With Only Simple Attributes-

- A strong entity set with only simple attributes will require only one table in relational model.
- Attributes of the table will be the attributes of the entity set.
- The primary key of the table will be the key attribute of the entity set.

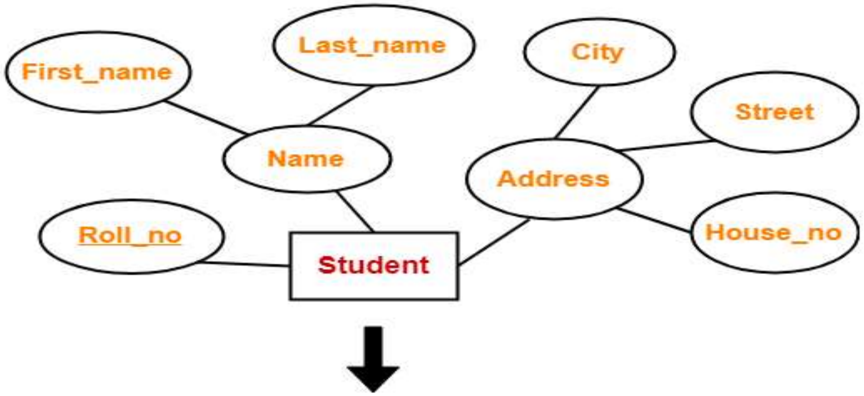


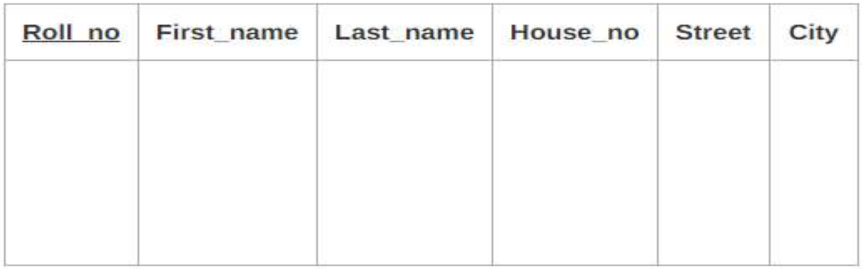


Roll_no	Name	Sex

Schema : Student (Roll_no , Name , Sex)

- **Rule-02: For Strong Entity Set With Composite Attributes-**
- A strong entity set with any number of composite attributes will require only one table in relational model.
- While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.

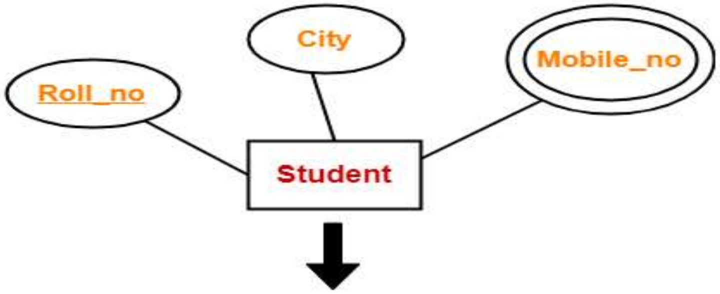




Roll_no	First_name	Last_name	House_no	Street	City

Schema : Student (Roll_no , First_name , Last_name , House_no , Street , City)

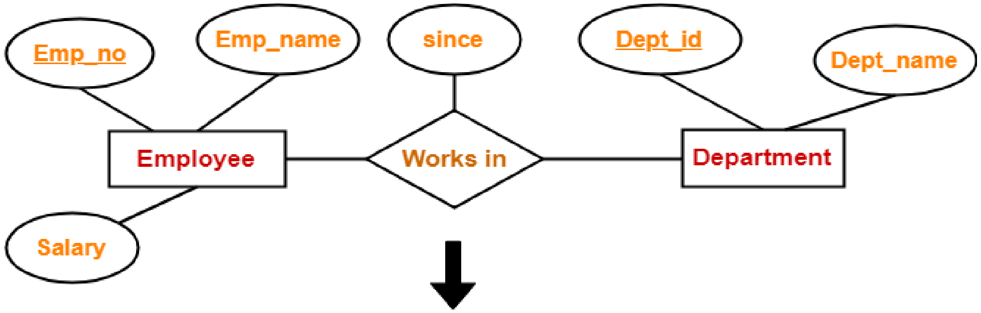
- **Rule-03: For Strong Entity Set With Multi Valued Attributes-**
-
- A strong entity set with any number of multi valued attributes will require two tables in relational model.
- One table will contain all the simple attributes with the primary key.
- Other table will contain the primary key and all the multi valued attributes.

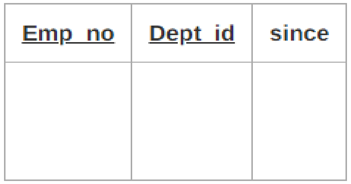


Roll_no	City

Roll_no	Mobile_no

- **Rule-04: Translating Relationship Set into a Table-**
- A relationship set will require one table in the relational model.
- Attributes of the table are-
- Primary key attributes of the participating entity sets
- Its own descriptive attributes if any.
- Set of non-descriptive attributes will be the primary key.



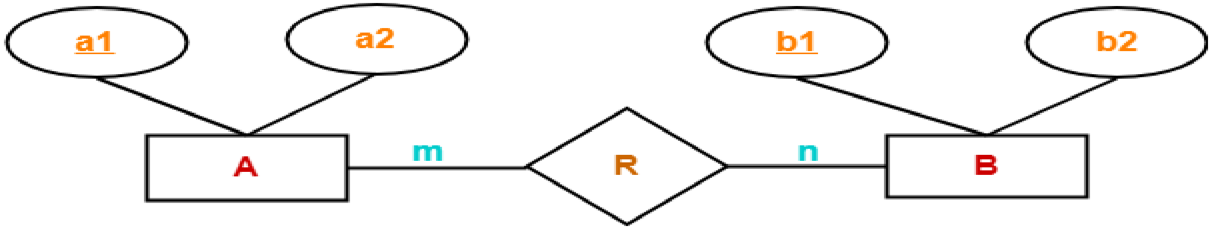


Emp_no	Dept_id	since

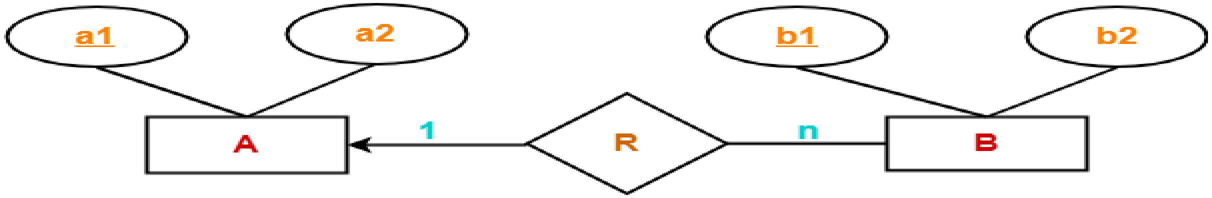
Schema : Works in (Emp_no , Dept_id , since)

- **NOTE-**
- If we consider the overall ER diagram, three tables will be required in relational model-
- One table for the entity set “Employee”
- One table for the entity set “Department”
- One table for the relationship set “Works in”
- **Rule-05: For Binary Relationships With Cardinality Ratios-**
- The following four cases are possible-
- **Case-01:** Binary relationship with cardinality ratio m:n
- **Case-02:** Binary relationship with cardinality ratio 1:n
- **Case-03:** Binary relationship with cardinality ratio m:1
- **Case-04:** Binary relationship with cardinality ratio 1:1

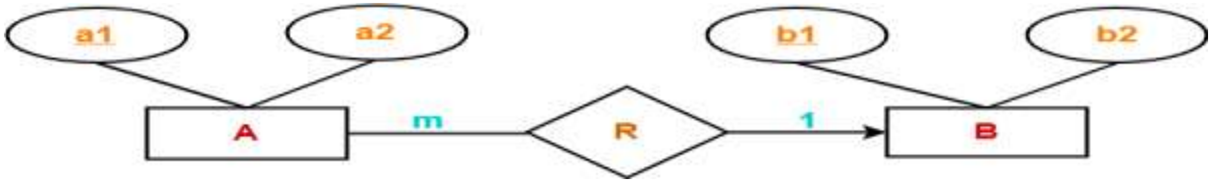
- **Case-01: For Binary Relationship With Cardinality Ratio m:n**
- Here, three tables will be required-
- A (a1 , a2)
- R (a1 , b1)
- B (b1 , b2)



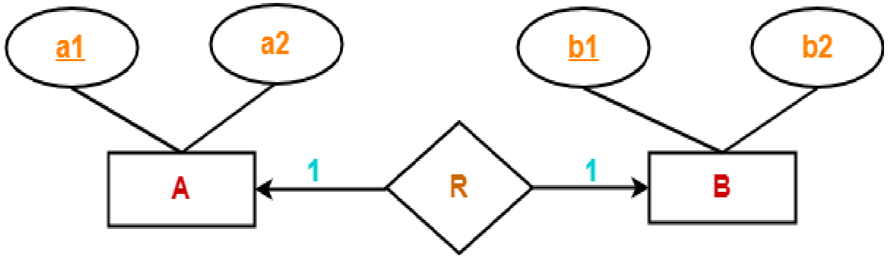
- **Case-02: For Binary Relationship With Cardinality Ratio 1:n**
- Here, two tables will be required-
- A (a1 , a2)
- BR (a1 , b1 , b2)
- **NOTE-** Here, combined table will be drawn for the entity set B and relationship set R.



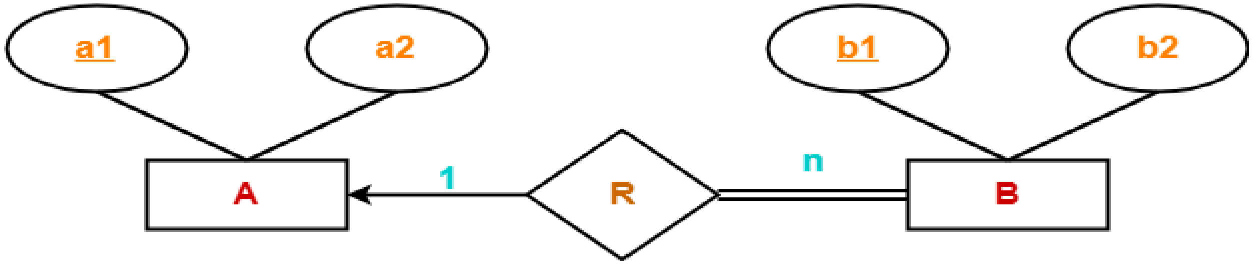
- **Case-03: For Binary Relationship With Cardinality Ratio m:1**
- Here, two tables will be required-
- AR (a1 , a2 , b1)
- B (b1 , b2)
- **NOTE-** Here, combined table will be drawn for the entity set A and relationship set R.



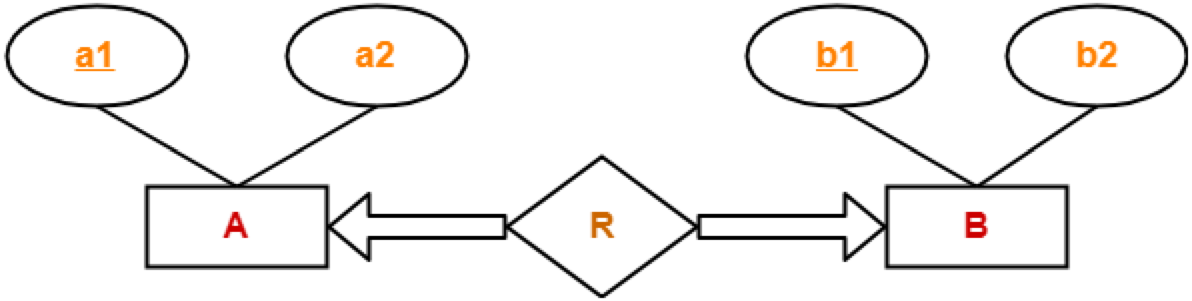
- **Case-04: For Binary Relationship With Cardinality Ratio 1:1**
- Here, two tables will be required. Either combine ‘R’ with ‘A’ or ‘B’
- **Way-01:**
- AR (a1 , a2 , b1)
- B (b1 , b2)
- **Way-02:**
- A (a1 , a2)
- BR (a1 , b1 , b2)



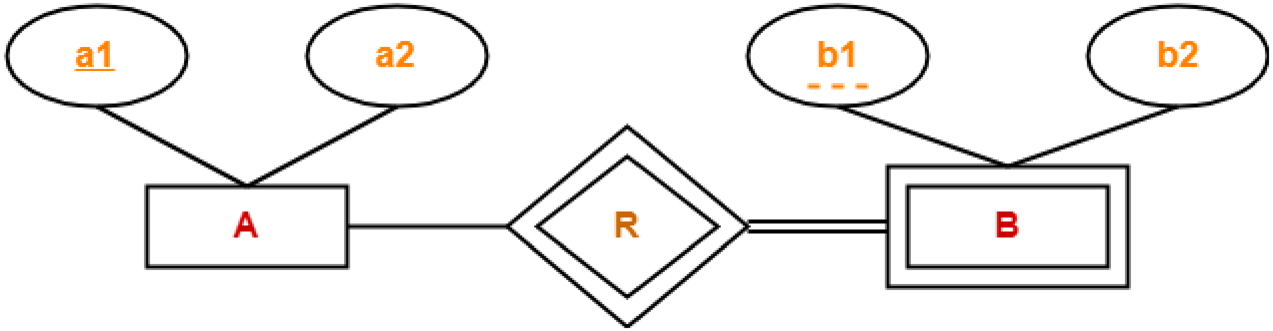
- **Rule-06: For Binary Relationship With Both Cardinality Constraints and Participation Constraints-**
- Cardinality constraints will be implemented as discussed in Rule-05.
- Because of the total participation constraint, foreign key acquires **NOT NULL** constraint i.e. now foreign key can not be null.
- **Case-01: For Binary Relationship With Cardinality Constraint and Total Participation Constraint From One Side-**
- Because cardinality ratio = 1 : n , so we will combine the entity set B and relationship set R.
- Then, two tables will be required-
- A (a1 , a2)
- BR (a1 , b1 , b2)
- Because of total participation, foreign key a1 has acquired NOT NULL constraint, so it can’t be null now.



- **Case-02: For Binary Relationship With Cardinality Constraint and Total Participation Constraint From Both Sides-**
-
- If there is a key constraint from both the sides of an entity set with total participation, then that binary relationship is represented using only single table.
- Here, Only one table is required.
- ARB (a1 , a2 , b1 , b2)



- **Rule-07: For Binary Relationship With Weak Entity Set-**
- Weak entity set always appears in association with identifying relationship with total participation constraint.
- Here, two tables will be required-
- A (a1 , a2)
- BR (a1 , b1 , b2)



Enhanced ER Model

- Todays time the complexity of the data is increasing so it becomes more and more difficult to use the traditional ER model for database modeling.
- To reduce this complexity of modeling we have to make improvements or enhancements were made to the existing ER model to make it able to handle the complex application in a better way.
- Enhanced entity-relationship diagrams are advanced database diagrams very similar to regular ER diagrams .
- In addition to ER model concepts EE-R includes −

➢ Subclasses and Super classes.

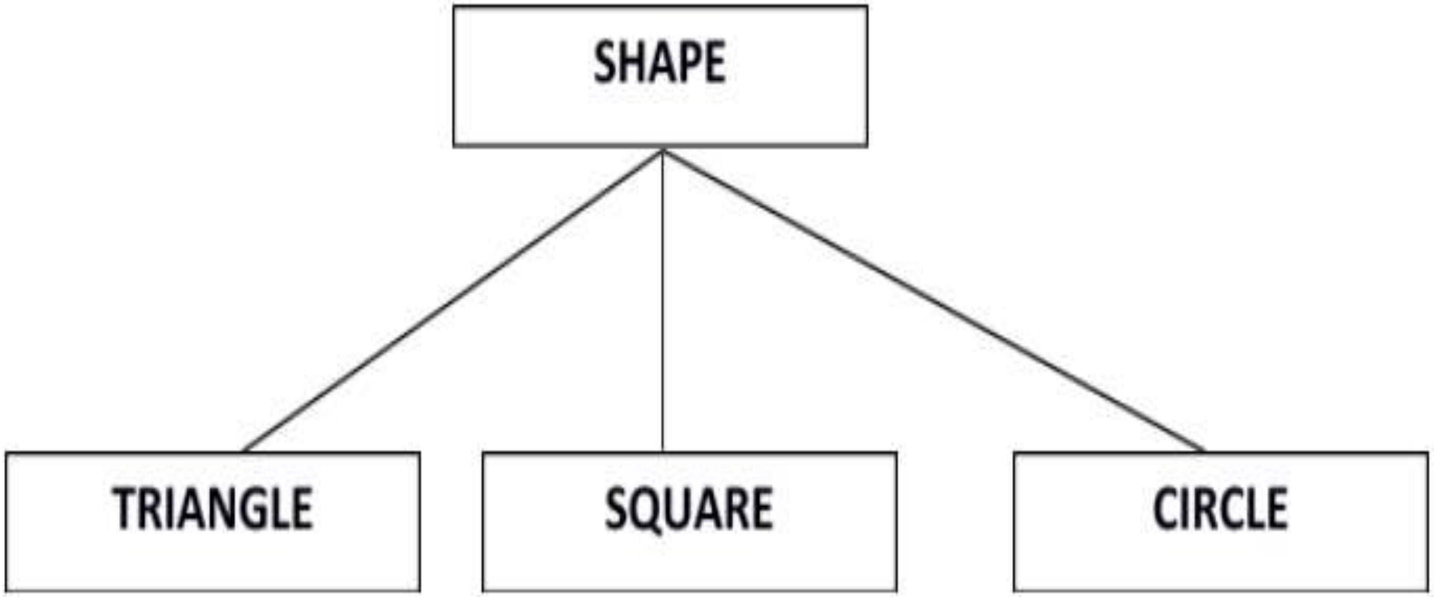
➢ Specialization and Generalization.

➢ Category or union type.

➢ Aggregation.

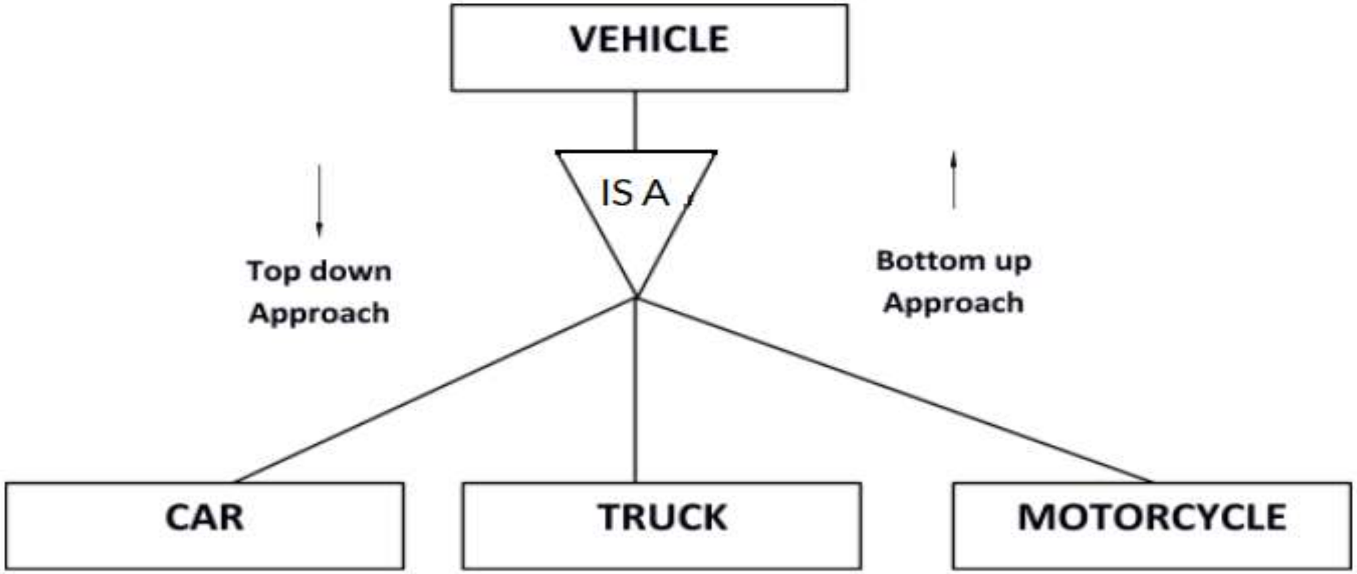
Subclasses and Super class

- Super class is an entity that can be divided into further subtype.
- For **example** − consider Shape super class.
- super class shape has sub groups: Triangle, Square and Circle.
- Sub classes are the group of entities with some unique attributes . Sub class inherits the properties and attributes from super class.



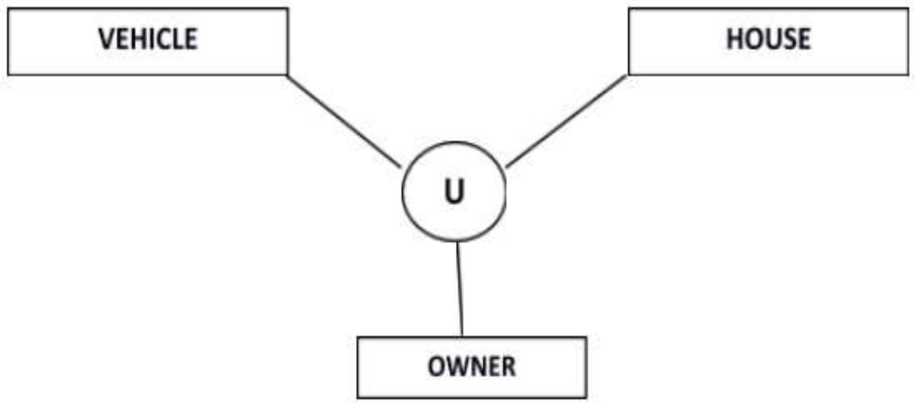
Specialization and Generalization

- Generalization is a process of generalizing an entity which contains generalized attributes or properties of generalized entities.
- It is a Bottom up process i.e. consider we have 3 sub entities Car, Truck and Motorcycle. Now these three entities can be generalized into one super class named as Vehicle.
- Specialization is a process of identifying subsets of an entity that share some different characteristic. It is a top down approach in which one entity is broken down into low level entity.
- In above example Vehicle entity can be a Car, Truck or Motorcycle.



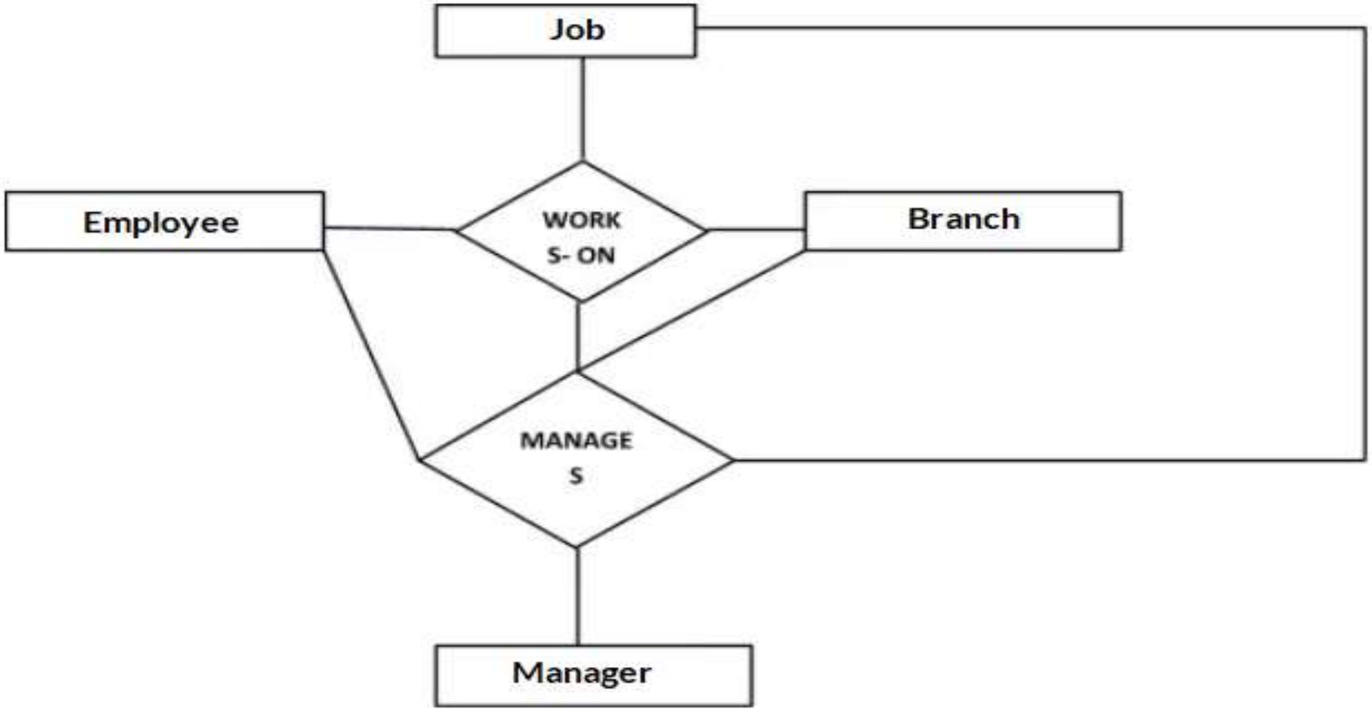
Category or Union

- Relationship of one super or sub class with more than one super class.
- Owner is the subset of two super class: Vehicle and House.



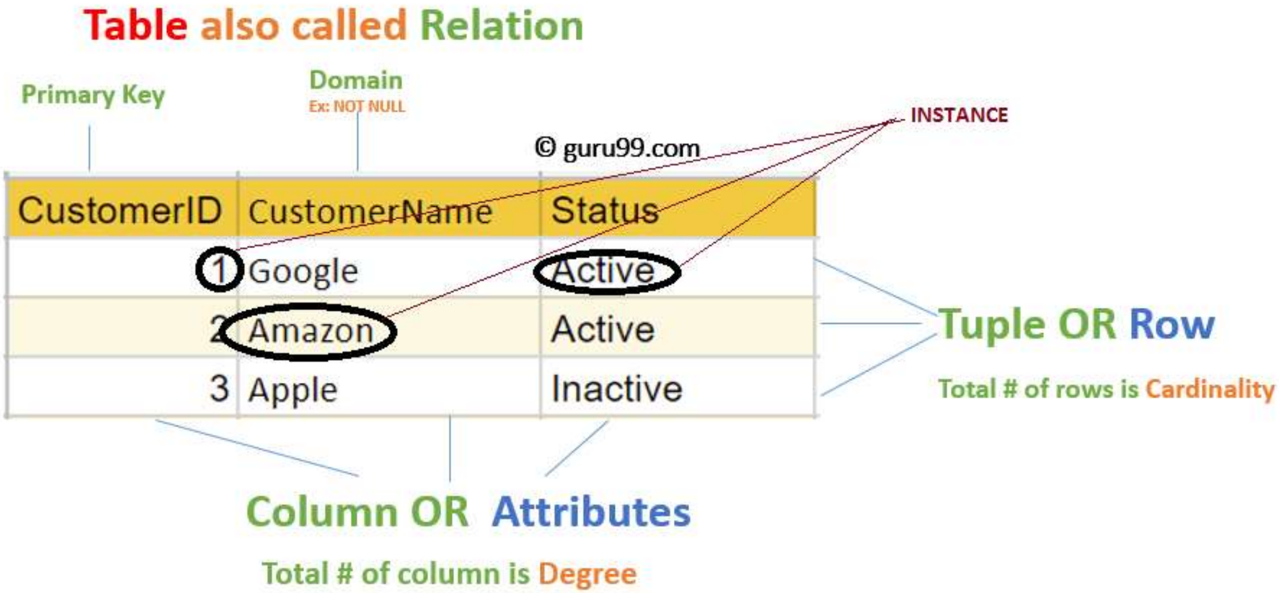
Aggregation

- Represents relationship between a whole object and its component.
- Consider a ternary relationship Works_On between Employee, Branch and Manager. Now the best way to model this situation is to use aggregation, So, the relationship-set, Works_On is a higher level entity-set. Such an entity-set is treated in the same manner as any other entity-set. We can create a binary relationship, Manager, between Works_On and Manager to represent who manages what tasks.



Basic

- **Attribute:** Each column in a Table.
- **Tuple** – a single row of a table, which contains a single record.
- **Degree:** The total number of attributes .
- **Cardinality:** Total number of rows .
- **domain** – a attribute has some pre-defined value.
- **Relationship** –a Table name
- **schema**- a structure or a table without values
- **Instance**- a single value



CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

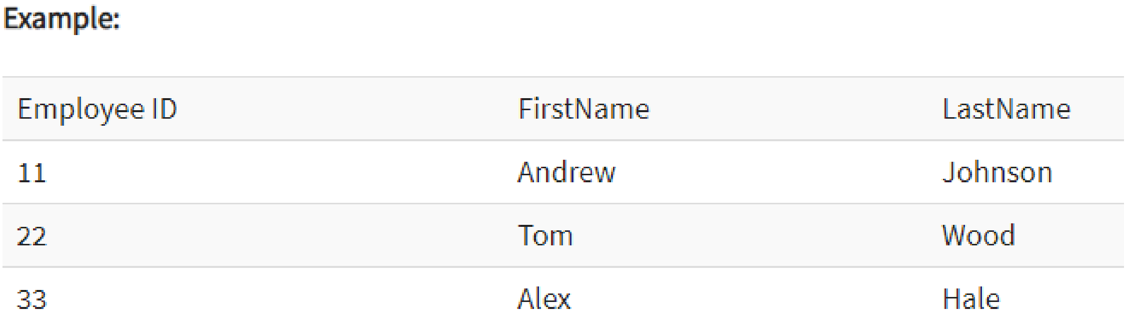
Table also called Relation
Primary Key
Domain
Ex: NOT NULL
INSTANCE
© guru99.com
Tuple OR Row
Total # of rows is Cardinality
Column OR Attributes
Total # of column is Degree

key

- **KEYS in DBMS** is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table).
- They allow you to find the relation between two tables.
- Keys help you uniquely identify a row in a table by a combination of one or more columns in that table.
- **Types of Keys in Database Management System:-**
- **Super Key -**
- **Candidate Key -**
- **Primary Key -**
- **Alternate Key -** .
- **Foreign Key -**
- **Compound Key -**
- **Composite Key -**

Superkey

- A superkey is a group of single or multiple keys which identifies rows in a table.



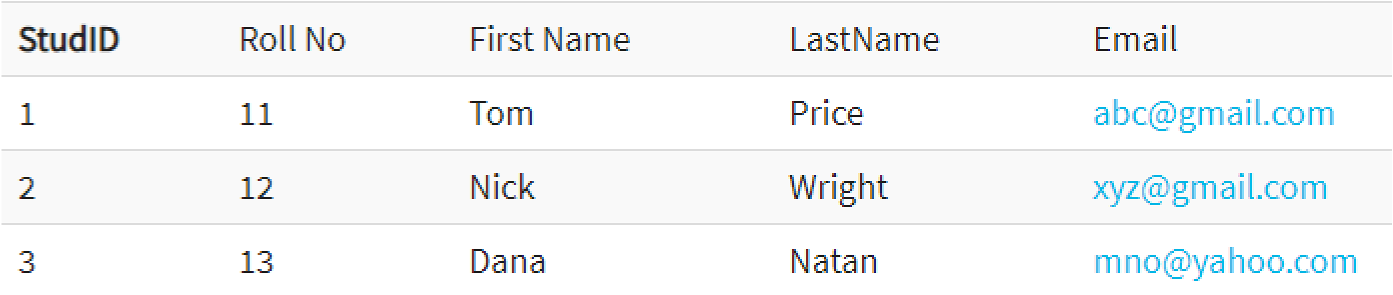
Example:

Employee ID	FirstName	LastName
11	Andrew	Johnson
22	Tom	Wood
33	Alex	Hale

- Super keys :-{employeeid ,first name, lastname, employeeid + firstname, firstname + lastname, employeeid + lastname, employeeid +first name+lastname}

CANDIDATE KEY

- **CANDIDATE KEY** is a set of attributes that uniquely identify tuples in a table.
- Candidate Key is a super key with no repeated attributes.
- The Primary key should be selected from the candidate keys.
- Every table must have at least a single candidate key.
- Candidate key may have multiple attributes
- It should contain minimum fields to ensure uniqueness

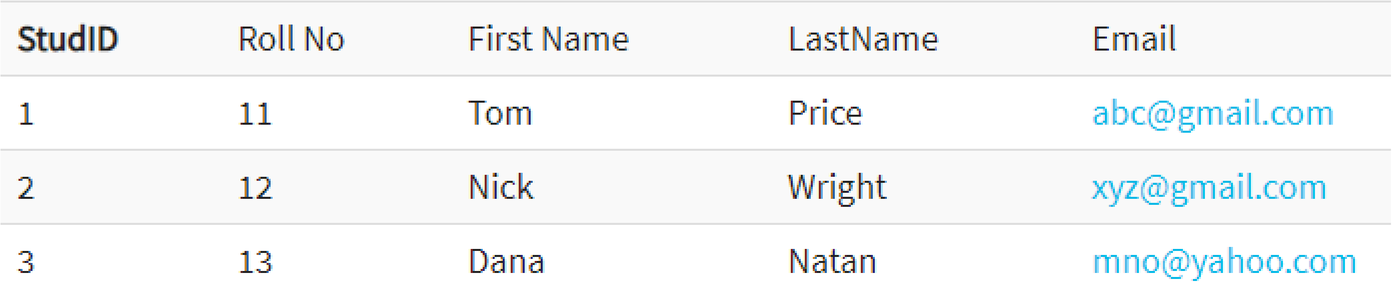


StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

- Candidate key{Studid ,rollno ,email }

PRIMARY KEY

- **PRIMARY KEY** is a column or group of columns in a table that uniquely identify every row in that table.
- The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table.
- A table cannot have more than one primary key.
- The primary key field cannot be null.
- A table can have multiple candidate keys but only a single primary key.
- The Primary key should be selected from the candidate keys.



StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

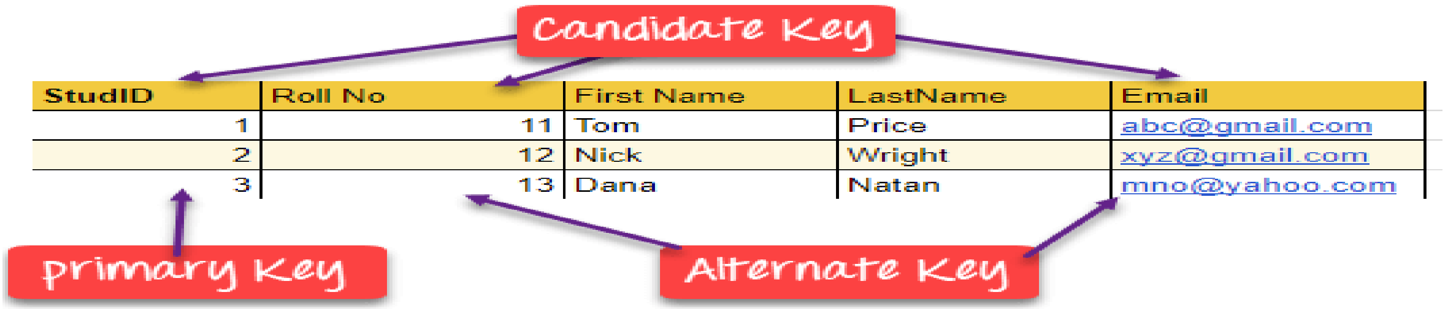
- Primary key {Studid **OR** rollno **OR** email }

ALTERNATE KEYS

- **ALTERNATE KEYS** is a column or group of columns in a table that uniquely identify every row in that table.
- A table can have multiple choices for a primary key but only one can be set as the primary key.
- All the keys which are not primary key are called an Alternate Key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

- In this table StudID, Roll No, Email are qualified to become a primary key. But since StudID is the primary key then Roll No and Email becomes the alternative key.



Candidate Key

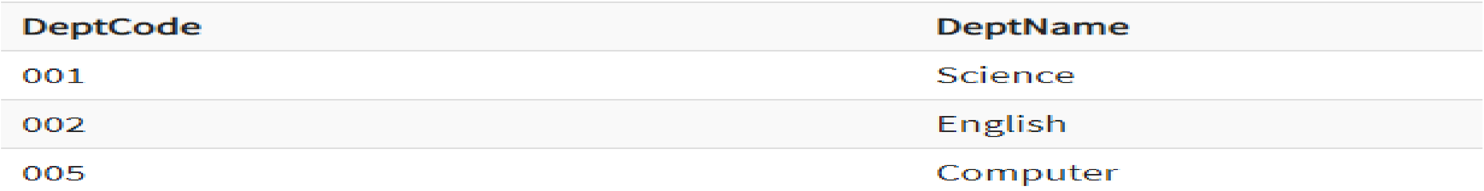
StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

Primary Key

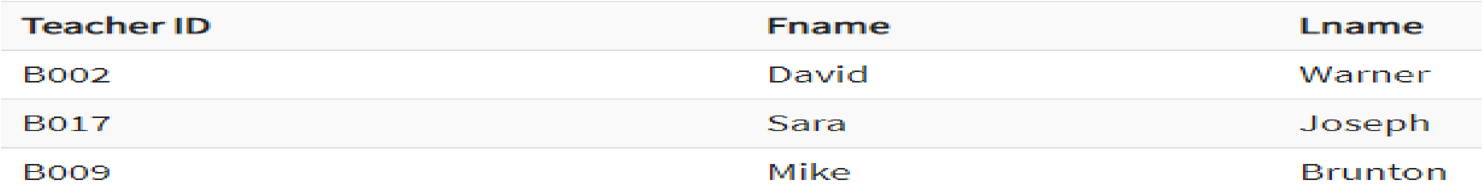
Alternate Key

FOREIGN KEY

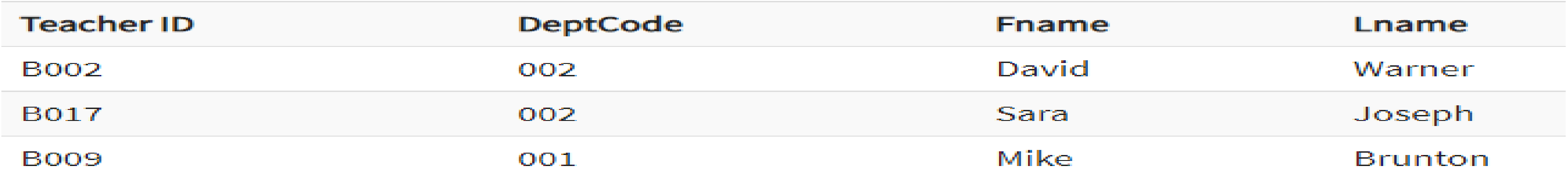
- **FOREIGN KEY** is a column that creates a relationship between two tables.
- The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.
- It acts as a cross-reference between two tables as it references the primary key of another table.
- In the 3rd table, adding the foreign key(Deptcode) , we can create a relationship between the two tables.



DeptCode	DeptName
001	Science
002	English
005	Computer



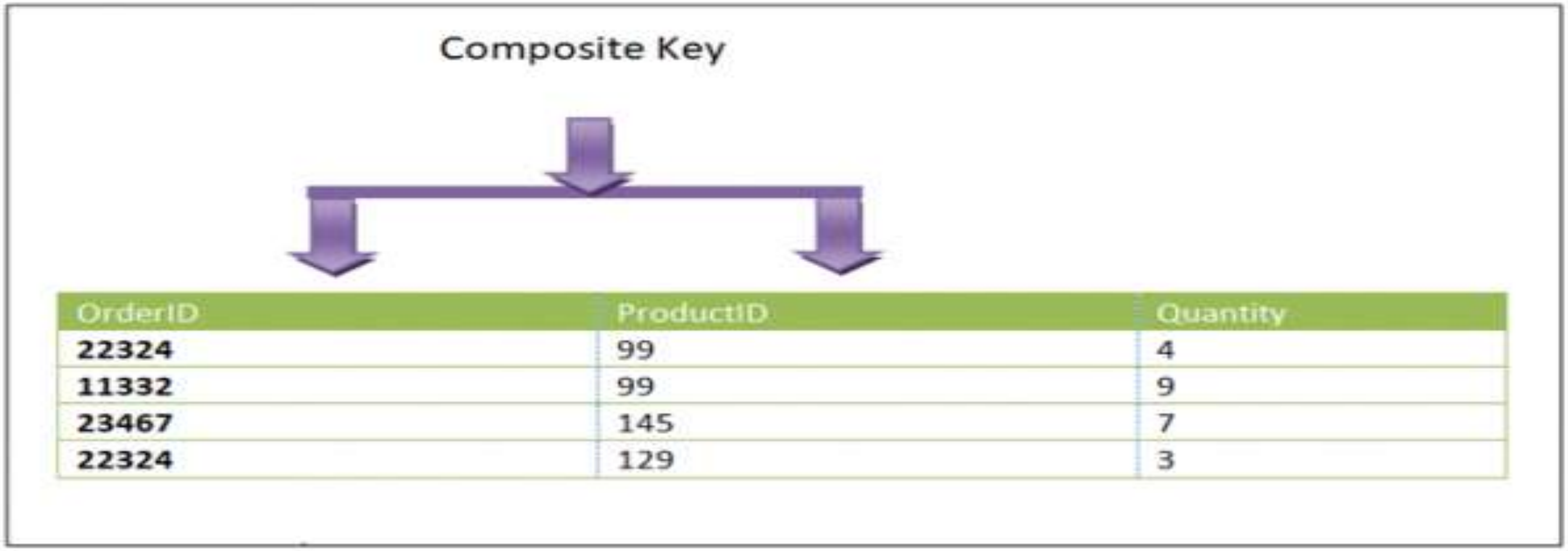
Teacher ID	Fname	Lname
B002	David	Warner
B017	Sara	Joseph
B009	Mike	Brunton



Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

composite key

- A 'combination of two or more' better describes the word 'composite'. Thus, a composite key in DBMS is a candidate key that is composed of two or more attributes and is capable of uniquely identifying a table or a relation.
- Such a key is also known as **Compound Key**, where each attribute creating a key is a foreign key in its own right.
- A primary key having two or more attributes is called composite key.
- Any key such as super key, primary key, candidate key etc. can be called composite key if it has more than one attributes.



OrderID	ProductID	Quantity
22324	99	4
11332	99	9
23467	145	7
22324	129	3