DATABASE MANAGEMENMT SYSTEM

UNIT 1

What is DBMS?

Database Management System (DBMS) is software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs that manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software store and retrieve data.
DBMS allows users to create their own databases as per their requirements. The term "DBMS" includes the user of the database and other application programs. It provides an interface between the data and the software application.

Example of a DBMS

Let us see a simple example of a university database. This database is maintaining information concerning students, courses, and grades in a university environment. The database is organized as five files:

- The STUDENT file stores the data of each student
- The COURSE file stores contain data on each course.
- The SECTION stores information about sections in a particular course.
- The GRADE file stores the grades which students receive in the various sections
- The TUTOR file contains information about each professor.

To define DBMS:

- We need to specify the structure of the records of each file by defining the different types of data elements to be stored in each record.
- We can also use a coding scheme to represent the values of a data item.
- Basically, your Database will have 5 tables with a foreign key defined amongst the various tables.

History of DBMS

Here, are the important landmarks from the history of DBMS:

- 1960 – Charles Bachman designed the first DBMS system
- 1970 – Codd introduced IBM’S Information Management System (IMS)
- 1976- Peter Chen coined and defined the Entity-relationship model, also known as the ER model
- 1980 – Relational Model becomes a widely accepted database component
- 1985- Object-oriented DBMS develops.
- 1990s- Incorporation of object-orientation in relational DBMS.
- 1991- Microsoft ships MS access, a personal DBMS, and that displaces all other personal DBMS products.
- 1995: First Internet database applications
- 1997: XML applied to database processing. Many vendors begin to integrate XML into DBMS products.

Characteristics of DBMS

Here are the characteristics and properties of a Database Management System:

- Provides security and removes redundancy
- Self-describing nature of a database system
- Insulation between programs and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing
- Database Management Software allows entities and relations among them to form tables.
- It follows the ACID concept (Atomicity, Consistency, Isolation, and Durability).
- DBMS supports a multi-user environment that allows users to access and manipulate data in parallel.

DBMS vs. Flat File

DBMS	Flat File Management System
Multi-user access	It does not support multi-user access

Design to fulfill the need of small and large businesses	It is only limited to smaller DBMS systems.
Remove redundancy and Integrity.	Redundancy and Integrity issues
Expensive. But in the long term Total Cost of Ownership is cheap	It’s cheaper
Easy to implement complicated transactions	No support for complicated transactions

Users of DBMS

Following are the various category of users of DBMS

Component Name	**Task**
Application Programmers	The Application programmers write programs in various programming languages to interact with databases.
Database Administrators	Database Admin is responsible for managing the entire DBMS system. He/She is called Database admin or DBA.
End-Users	The end users are the people who interact with the database management system. They conduct various operations on databases like retrieving, updating, deleting, etc.

Popular DBMS Software

Here is the list of some popular DBMS systems:

- MySQL
- Microsoft Access
- Oracle
- PostgreSQL
- dBASE
- FoxPro
- SQLite
- IBM DB2
- LibreOffice Base

- MariaDB
- Microsoft SQL Server

Application of DBMS

Below are the popular database system applications:

Sector	Use of DBMS
Banking	For customer information, account activities, payments, deposits, loans, etc.
Airlines	For reservations and schedule information.
Universities	For student information, course registrations, colleges, and grades.
Telecommunication	It helps to keep call records, monthly bills, maintain balances, etc.
Finance	For storing information about stock, sales, and purchases of financial instruments like stocks and bonds.
Sales	Use for storing customer, product & sales information.
Manufacturing	It is used to manage the supply chain and track the production of items. Inventories status in warehouses.
HR Management	For information about employees, salaries, payroll, deduction, generation of paychecks, etc.

Types of DBMS

Hierarchical
Network
Relational
Object Oriented

Types of DBMS
The main Four Types of Database Management Systems are:

- Hierarchical database
- Network database
- Relational database
- Object-Oriented database

Hierarchical DBMS

In a Hierarchical database, model data is organized in a tree-like structure. Data is Stored Hierarchically (top-down or bottom-up) format. Data is represented using a parent-child relationship. In Hierarchical DBMS, parents may have many children, but children have only one parent.

Network Model

The network database model allows each child to have multiple parents. It helps you to address the need to model more complex relationships like the orders/parts many-to-many relationship. In this model, entities are organized in a graph which can be accessed through several paths.

Relational Model

Relational DBMS is the most widely used DBMS model because it is one of the easiest. This model is based on normalizing data in the rows and columns of the tables. Relational model stored in fixed structures and manipulated using SQL.

Object-Oriented Model

In the Object-oriented Model data is stored in the form of objects. The structure is called classes which display data within it. It is one of the components of DBMS that defines a database as a collection of objects that stores both data members' values and operations.

Advantages of DBMS

- DBMS offers a variety of techniques to store & retrieve data
- DBMS serves as an efficient handler to balance the needs of multiple applications using the same data
- Uniform administration procedures for data
- Application programmers are never exposed to details of data representation and storage.
- A DBMS uses various powerful functions to store and retrieve data efficiently.
- Offers Data Integrity and Security
- The DBMS implies integrity constraints to get a high level of protection against prohibited access to data.
- A DBMS schedules concurrent access to the data in such a manner that only one user can access the same data at a time
- Reduced Application Development Time

Disadvantage of DBMS

DBMS may offer plenty of advantages, but it has certain flaws-

- The cost of Hardware and Software of a DBMS is quite high, which increases the budget of your organization.
- Most database management systems are often complex, so training users to use the DBMS is required.
- In some organizations, all data is integrated into a single database that can be damaged because of electric failure or corruption in the storage media.
- Using the same program at a time by multiple users sometimes leads to data loss.
- DBMS can't perform sophisticated calculations

When not to use a DBMS system?

Although DBMS system is useful, it is still not suited for the specific task mentioned below:

Not recommended when you do not have the budget or the expertise to operate a DBMS. In such cases, Excel/CSV/Flat Files could do just fine.

For Web 2.0 applications, it's better to use NoSQL DBMS

Characteristics of Database Management System

1. Real World Entity

DBMS these days is very realistic and real-world entities are used to design its architecture. Also, behavior and attributes are used by DBMS. To simplify it we can take an example of an organization database where employee is an entity and his employee id is an attribute.

2. Self-Describing Nature

Before DBMS, a traditional file management system was used for storing information and data. There was no concept of definition in traditional file management system like we have in DBMS. A DBMS should be of Self- Describing nature as it not only contains the database itself but also the metadata. A metadata (data about data) defines and describes not only the extent, type, structure, and format of all data but also relationship between data. This data represents itself that what actions should be taken on it.

3. Support ACID Properties

Any DBMS is able to support ACID (Accuracy, Completeness, Isolation, and Durability) properties. It is made sure in every DBMS that the real purpose of data should not be lost while performing transactions like delete, insert, and update. Let us take an example; if an employee's name is updated then it should make sure that there is no duplicate data and no mismatch of employee information.

4. Concurrent Use of Database

There are many chances that many users will be accessing the data at the same time. They may require altering the database system concurrently. At that time, DBMS supports them to concurrently use database without any problem. With the help of concurrency, economy of the system can be increased. For Example, employees of the railway reservation system can book and access tickets for passengers concurrently. Every employee can see on his own interface that how many seats are available or bogie is fully booked.

5. Insulation Between Data and Program

Program-data independence provides a big relief to database users. In traditional file management system, structure of data files was defined in the application programs so the user had to change all the programs that are using that particular data file.
But in DBMS, structure of data files is not stored in the program but it is stored in system catalogue. With the help of this, internal improvement of data efficiency or any changes in the data do not have any effect on application software.

6. Transactions

Transactions are bunch of actions that are done to bring database from one consistent state to new consistent state. Traditional file-based system did not have this feature. Transaction is always atomic which means it can never be further divided. It can only be completed or uncompleted.
For example, A person wants to credit money from his account to another person's account. Then transaction will be complete if he sends money and another guy receives his money. Anything other than this can lead to an inconsistent transaction.

7. Data Persistence

Persistence means if the data is not removed explicitly then all the data will be maintained in DBMS. If any system failure happens then life span of data stored in the DBMS will be decided by the users directly or indirectly. Any data stored in the DBMS can never be lost. If system failure happens in between any transaction then it will be rolled back or fully completed, but data will never be at risk.

8. Backup and Recovery

There are many chances of failure of the whole database. At that time no one will be able to get the database back and for sure company will be in a big loss. The only solution is to take backup of database and whenever it is needed, it can be stored back. A database must have this characteristic to enable more effectiveness.

9. Data Integrity

This is one of the most important characteristics of database management system. Integrity ensures the quality and reliability of database system. It protects unauthorized access to the database and makes it more secure. It brings only consistent and accurate data into the database.

10. Multiple Views

Users can have multiple views of database depending on their department and interest. DBMS support multiple views of database to the users. For example, a user of the teaching department will have different view and user of hostel department will have different. This feature helps users to have somewhat security because users of other departments cannot access their files.

11. Stores Any Kind of Data

A database management system should be able to store any kind of data. It should not be restricted to employee name, salary, and address. Any kind of data that exists in the real world can be stored in DBMS because we need to work with all kinds of data that is present around us.

12. Security

DBMS provides security to the data stored in it because all users have different rights to access database. Some of the users can access the whole database while other can access a small part of database. For example, a computer network lecturer can only access files that are related to computer subjects but HOD of the department can access files of all subjects that are related to their department.

13. Represents Complex Relationship Between Data

Data stored in a database is connected with each other and a relationship is made in between data. DBMS should be able to represent the complex relationship between data to make efficient and accurate use of data.

14. Query Language

Queries are used to retrieve and manipulate data but DBMS is armed by a strong query language that makes it more effective and efficient. Users have the power to retrieve any kind of data they want from the database by applying different sets of queries. The file-Based system has not this luxury of the query language.

15. Cost

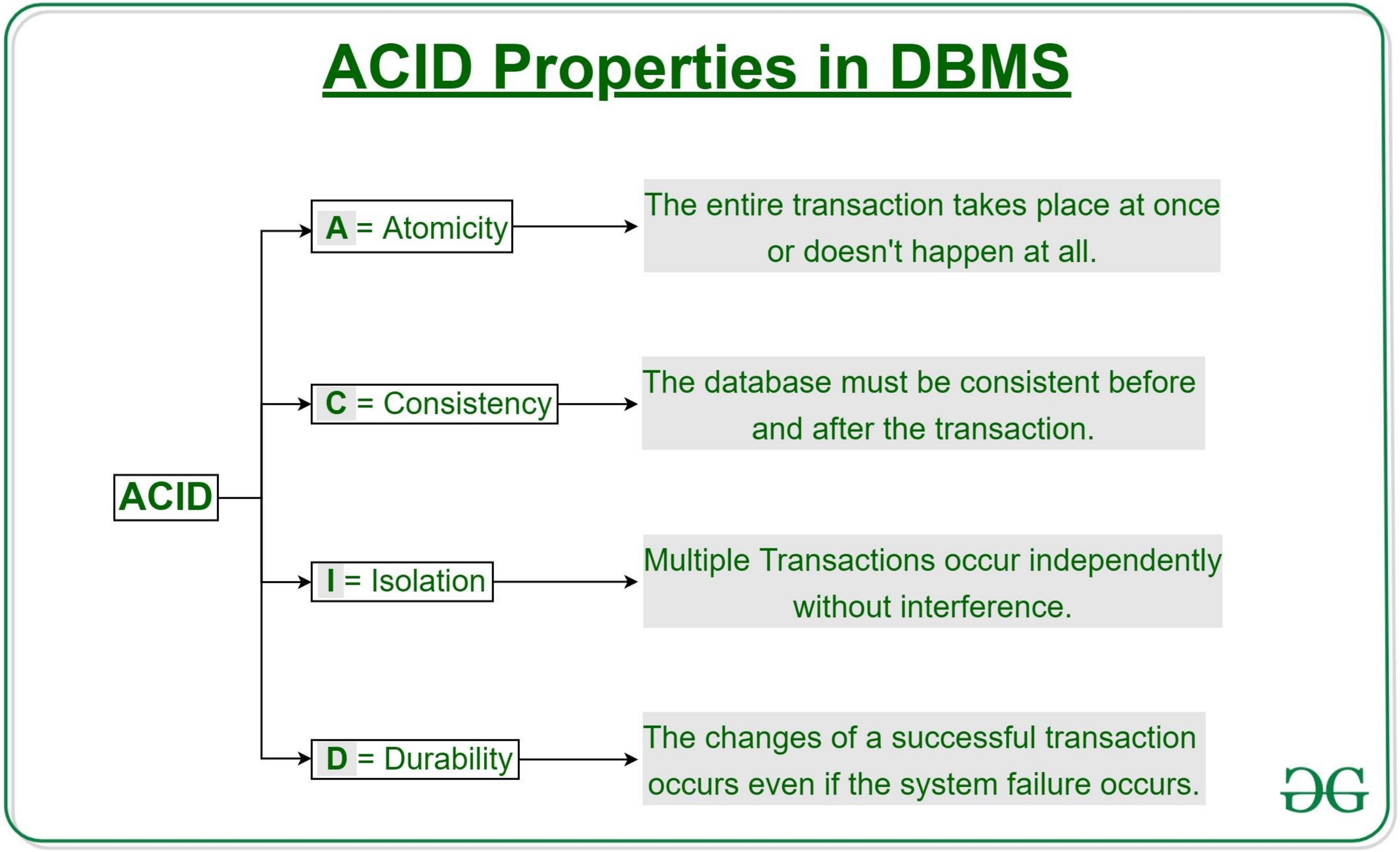
The cost of the DBMS is high as compared to the other software and technology available in the market. But if you consider the long run then DBMS is way far better because its maintenance cost will be almost nothing

So it was all about **Characteristics of Database Management System**. If you have any problem in these characteristics then please comment below.

ACID Properties in DBMS

A **transaction** is a single logical unit of work that accesses and possibly modifies the contents of a database. Transactions access data using read and write operations.

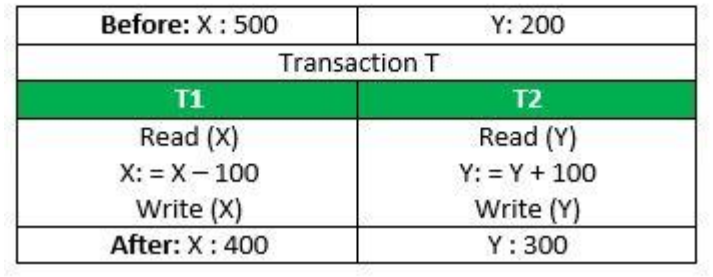
In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called **ACID** properties.


ACID Properties in DBMS
ACID
A = Atomicity
The entire transaction takes place at once or doesn't happen at all.
C = Consistency
The database must be consistent before and after the transaction.
I = Isolation
Multiple Transactions occur independently without interference.
D = Durability
The changes of a successful transaction occurs even if the system failure occurs.

Atomicity:
By this, we mean that either the entire transaction takes place at once or doesn't happen at all. There is no midway i.e. transactions do not occur partially. Each transaction is considered as one unit and either runs to completion or is not executed at all. It involves the following two operations.

—**Abort**: If a transaction aborts, changes made to the database are not visible.
—**Commit**: If a transaction commits, changes made are visible.
Atomicity is also known as the 'All or nothing rule'.

Consider the following transaction **T** consisting of **T1** and **T2**: Transfer of 100 from account **X** to account **Y**.



Before: X : 500	Y: 200
Transaction T	
T1	**T2**
Read (X) X: = X − 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

If the transaction fails after completion of **T1** but before completion of **T2**.(say, after **write(X)** but before **write(Y)**), then the amount has been deducted from **X** but not added to **Y**. This results in an inconsistent database state. Therefore, the transaction must be executed in its entirety in order to ensure the correctness of the database state.

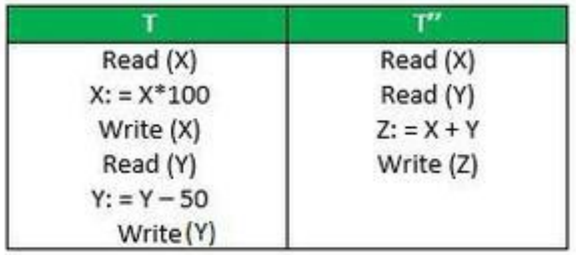
Consistency:

This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to the correctness of a database. Referring to the example above,
The total amount before and after the transaction must be maintained.
Total **before T** occurs = **500 + 200 = 700**.
Total **after T occurs** = **400 + 300 = 700**.
Therefore, the database is **consistent**. Inconsistency occurs in case **T1** completes but **T2** fails. As a result, T is incomplete.

Isolation:

This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of the database state. Transactions occur independently without interference. Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed. This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order.

Let **X**= 500, **Y** = 500.
Consider two transactions **T** and **T".**



T	T"
Read (X)	Read (X)
X: = X*100	Read (Y)
Write (X)	Z: = X + Y
Read (Y)	Write (Z)
Y: = Y – 50	
Write (Y)	

Suppose **T** has been executed till **Read (Y)** and then **T''** starts. As a result, interleaving of operations takes place due to which **T''** reads the correct value of **X** but the incorrect value of **Y** and sum computed by
T'': (X+Y = 50, 000+500=50, 500)
is thus not consistent with the sum at end of the transaction:
T: (X+Y = 50, 000 + 450 = 50, 450).
This results in database inconsistency, due to a loss of 50 units. Hence, transactions must take place in isolation and changes should be visible only after they have been made to the main memory.

Durability:

This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs. These updates now become permanent and are stored in non-volatile memory. The effects of the transaction, thus, are never lost.

Some important points:

Property	Responsibility for maintaining properties
Atomicity	Transaction Manager
Consistency	Application programmer

Property	Responsibility for maintaining properties
Isolation	Concurrency Control Manager
Durability	Recovery Manager

The **ACID** properties, in totality, provide a mechanism to ensure the correctness and consistency of a database in a way such that each transaction is a group of operations that acts as a single unit, produces consistent results, acts in isolation from other operations, and updates that it makes are durably stored.

Difference between DBMS and RDBMS

Although DBMS and RDBMS both are used to store information in physical database but there are some remarkable differences between them.

The main differences between DBMS and RDBMS are given below:

No.	DBMS	RDBMS
1)	DBMS applications store **data as file**.	RDBMS applications store **data in a tabular form**.
2)	In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3)	**Normalization is not** present in DBMS.	**Normalization is** present in RDBMS.

4)	DBMS does **not apply any security** with regards to data manipulation.	RDBMS **defines the integrity constraint** for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property.
5)	DBMS uses file system to store data, so there will be **no relation between the tables**.	in RDBMS, data values are stored in the form of tables, so a **relationship** between these data values will be stored in the form of a table as well.
6)	DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
7)	DBMS **does not support distributed database**.	RDBMS **supports distributed database**.
8)	DBMS is meant to be for small organization and **deal with small data**. it supports **single user**.	RDBMS is designed to **handle large amount of data**. it supports **multiple users**.
9)	Examples of DBMS are file systems, **xml** etc.	Example of RDBMS are **mysql**, **postgre**, **sql server**, **oracle** etc.

After observing the differences between DBMS and RDBMS, you can say that RDBMS is an extension of DBMS. There are many software products in the market today who are compatible for both DBMS and RDBMS. Means today a RDBMS application is DBMS application and vice-versa.

WHAT IS DATA MODELING?

Data modeling is the process of creating a visual representation of either a whole information system or parts of it to communicate connections between data points and structures. The goal is to illustrate the types of data used and stored within the system, the relationships among these data types, the ways the data can be grouped and organized and its formats and attributes.

Data models are built around business needs. Rules and requirements are defined upfront through feedback from business stakeholders so they can be incorporated into the design of a new system or adapted in the iteration of an existing one.

Data can be modeled at various levels of abstraction. The process begins by collecting information about business requirements from stakeholders and end users. These business rules are then translated into data structures to formulate a concrete database design. A data model can be compared to a roadmap, an architect's blueprint or any formal diagram that facilitates a deeper understanding of what is being designed.

Data modeling employs standardized schemas and formal techniques. This provides a common, consistent, and predictable way of defining and managing data resources across an organization, or even beyond.

Ideally, data models are living documents that evolve along with changing business needs. They play an important role in supporting business processes and planning IT architecture and strategy. Data models can be shared with vendors, partners, and/or industry peers.
Types of data models

Like any design process, database and information system design begins at a high level of abstraction and becomes increasingly more concrete and specific. Data models can generally be divided into three categories, which vary according to their degree of abstraction. The process will start with a conceptual model, progress to a logical model and conclude with a physical model. Each type of data model is discussed in more detail below:

- **Conceptual data models.** They are also referred to as domain models and offer a big-picture view of what the system will contain, how it will be organized, and which business rules are involved. Conceptual models are usually created as part of the process of gathering initial project requirements. Typically, they include entity classes (defining the types of things that are important for the business to represent in the data model), their characteristics and constraints, the relationships between them and relevant security and data integrity requirements. Any notation is typically simple.

- **Logical data models.** They are less abstract and provide greater detail about the concepts and relationships in the domain under consideration. One of several formal data modeling notation systems is followed. These indicate data attributes, such as data types and their corresponding lengths, and show the relationships among entities. Logical data models don't specify any technical system requirements. This stage is frequently omitted in agile or DevOps practices. Logical data models can be useful in highly procedural implementation environments, or for projects that are data-oriented by nature, such as data warehouse design or reporting system development.

- **Physical data models.** They provide a schema for how the data will be physically stored within a database. As such, they're the least abstract of all. They offer a finalized design that can be implemented as a relational database, including associative tables that illustrate the relationships among entities as well as the primary keys and foreign keys that will be used to maintain those relationships. Physical data models can include database management system (DBMS)-specific properties, including performance tuning.

TYPES OF DATA MODELING

Data modeling has evolved alongside database management systems, with model types increasing in complexity as businesses' data storage needs have grown. Here are several model types:

- **Hierarchical data models** represent one-to-many relationships in a treelike format. In this type of model, each record has a single root or parent which maps to one or more child tables. This model was implemented in the IBM Information Management System (IMS), which was introduced in 1966 and rapidly found widespread use, especially in banking. Though this approach is less efficient than more recently developed database models, it's still used in Extensible Markup Language (XML) systems and geographic information systems (GISs).
- **Relational data models** were initially proposed by IBM researcher E.F. Codd in 1970. They are still implemented today in the many different relational databases commonly used in enterprise computing. Relational data modeling doesn't require a detailed understanding of the physical properties of the data storage being used. In it, data segments are explicitly joined through the use of tables, reducing database complexity.

Relational databases frequently employ structured query language (SQL) for data management. These databases work well for maintaining data integrity and minimizing redundancy. They're often used in point-of-sale systems, as well as for other types of transaction processing.

- **Entity-relationship (ER) data models** use formal diagrams to represent the relationships between entities in a database. Several ER modeling tools are used by data architects to create visual maps that convey database design objectives.
- **Object-oriented data models** gained traction as object-oriented programming and it became popular in the mid-1990s. The "objects" involved are abstractions of real-world entities. Objects are grouped in class hierarchies, and have associated features. Object-oriented databases can incorporate tables, but can also support more complex data relationships. This approach is employed in multimedia and hypertext databases as well as other use cases.
- **Dimensional data models** were developed by Ralph Kimball, and they were designed to optimize data retrieval speeds for analytic purposes in a data warehouse. While relational and ER models emphasize efficient storage, dimensional models increase redundancy in order to make it easier to locate information for reporting and retrieval. This modeling is typically used across OLAP systems.

Two popular dimensional data models are the star schema, in which data is organized into facts (measurable items) and dimensions (reference information), where each fact is surrounded by its associated dimensions in a star-like pattern. The other is the snowflake schema, which resembles the star schema but includes additional layers of associated dimensions, making the branching pattern more complex.

BENEFITS OF DATA MODELING

Data modeling makes it easier for developers, data architects, business analysts, and other stakeholders to view and understand relationships among the data in a database or data warehouse. In addition, it can:

- Reduce errors in software and database development.
- Increase consistency in documentation and system design across the enterprise.
- Improve application and database performance.
- Ease data mapping throughout the organization.
- Improve communication between developers and business intelligence teams.
- Ease and speed the process of database design at the conceptual, logical and physical levels.

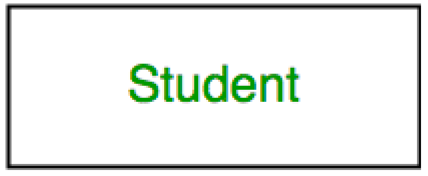
Components of E-R Model

1-Rectangles	▭	Represents entity
2-Ellipse	⬭	Represents attributes
3-Diamonds	◇	Represents relationship among entities
4-Lines	—	link attributes to entities and entity sets to relationship

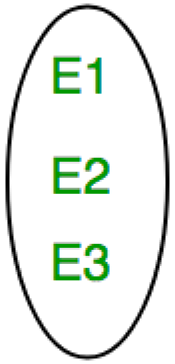
Entity, Entity Type, Entity Set –

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.

An Entity is an object of Entity Type and a set of all entities is called as an 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:

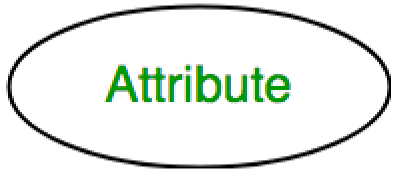


Entity Type

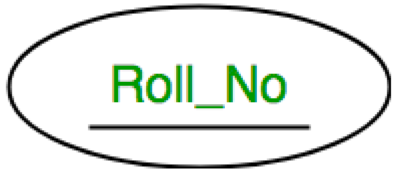


Entity Set

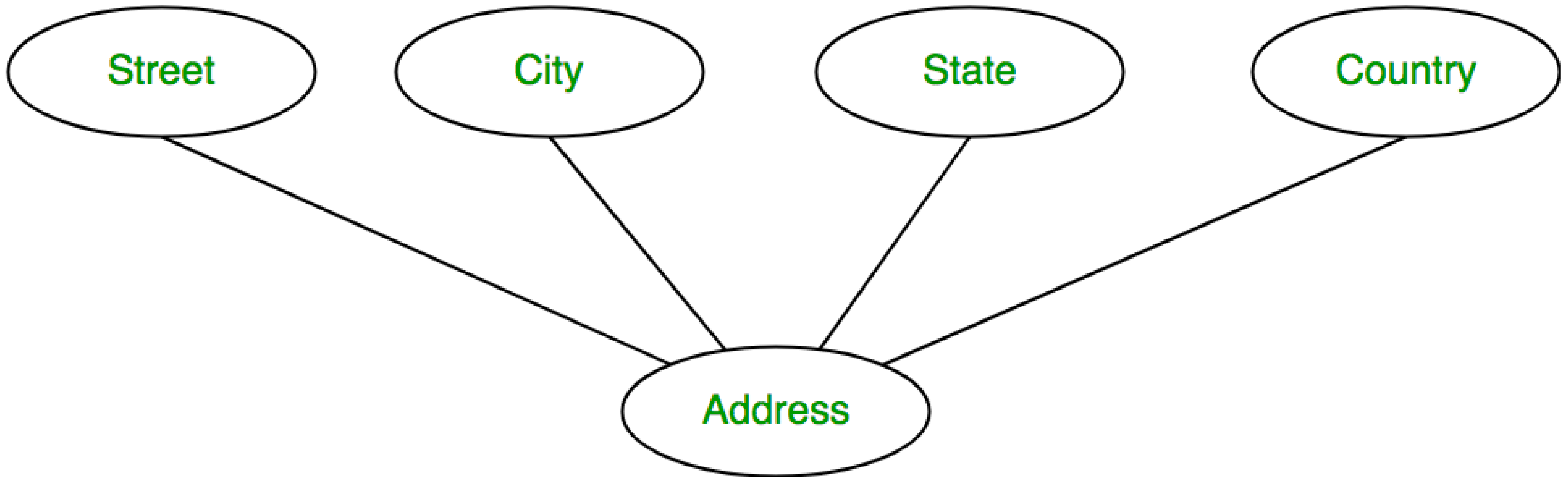
Attribute(s):
Attributes are the **properties that define the entity type**. For example, Roll_No, Name, DOB, Age, Address, Mobile_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.



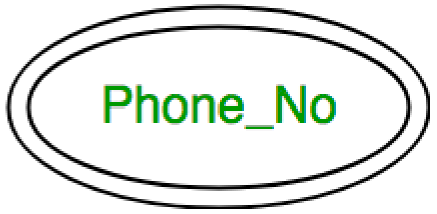
1. Key Attribute –
The attribute which **uniquely identifies each entity** in the entity set is called key attribute.For example, Roll_No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.



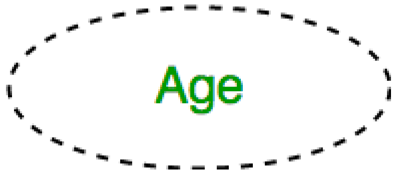
2. Composite Attribute –
An attribute **composed of many other attribute** is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals.



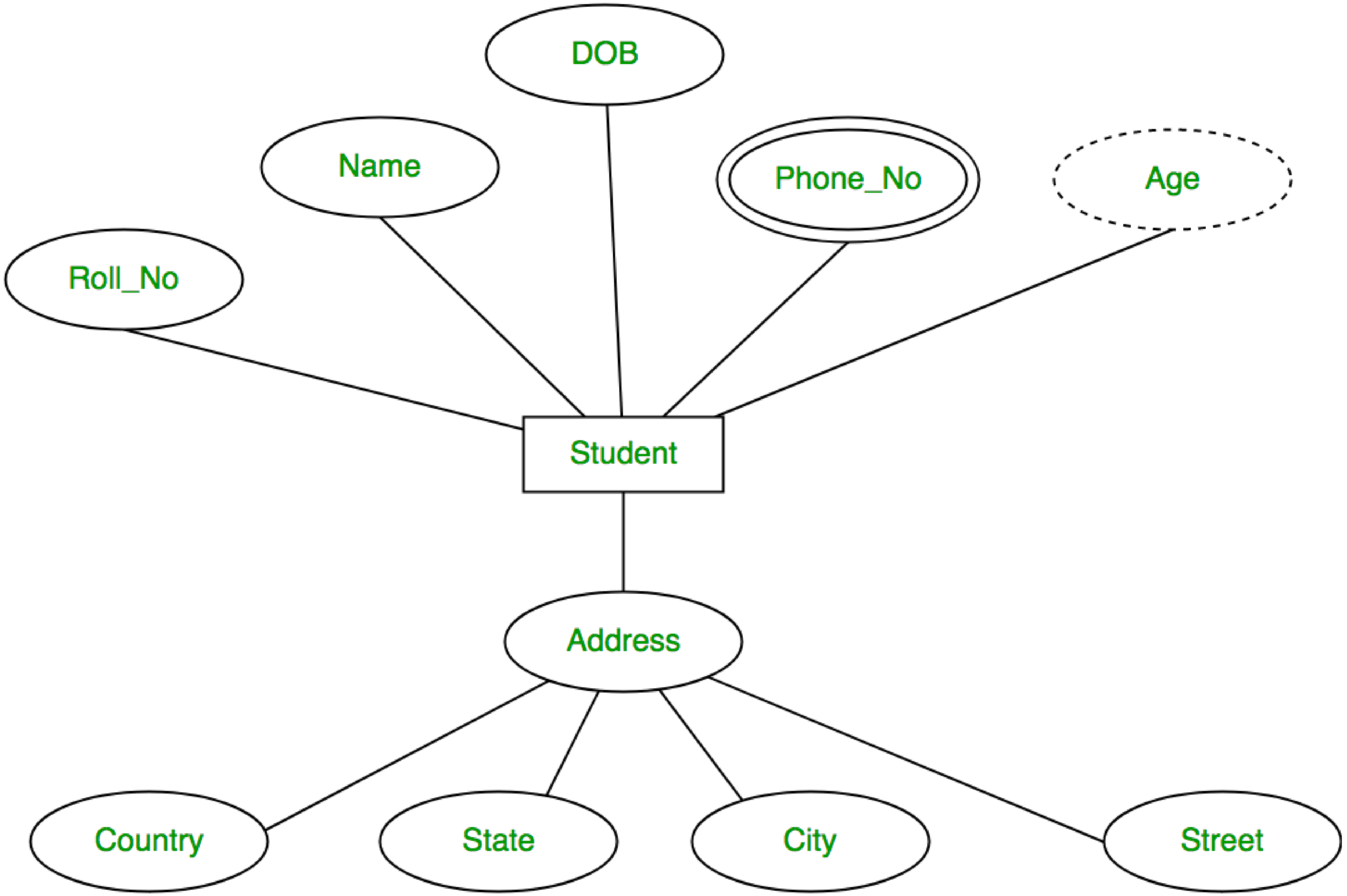
3. Multivalued Attribute –
An attribute consisting **more than one value** for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.



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.



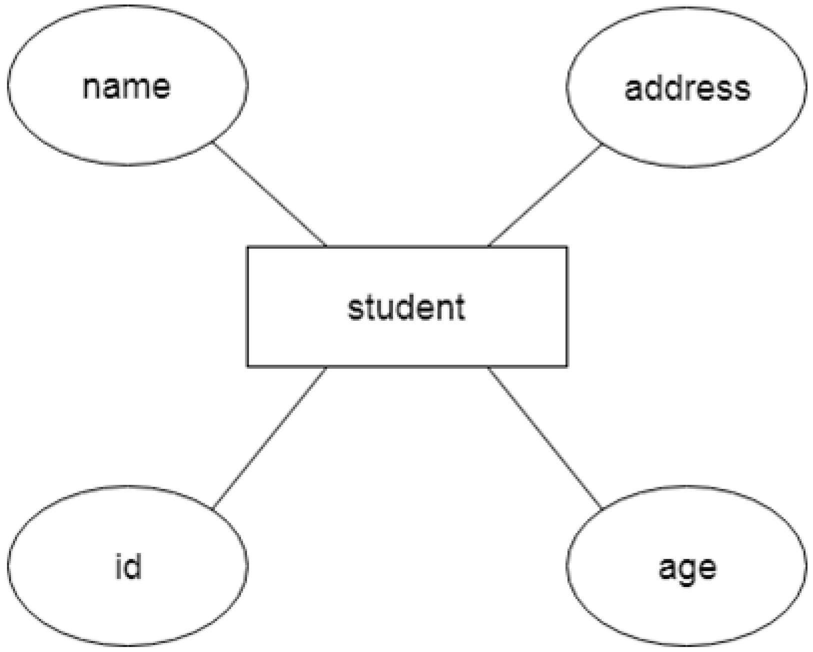
The complete entity type **Student** with its attributes can be represented as:


DOB
Name
Phone_No
Age
Roll_No
Student
Address
Country
State
City
Street

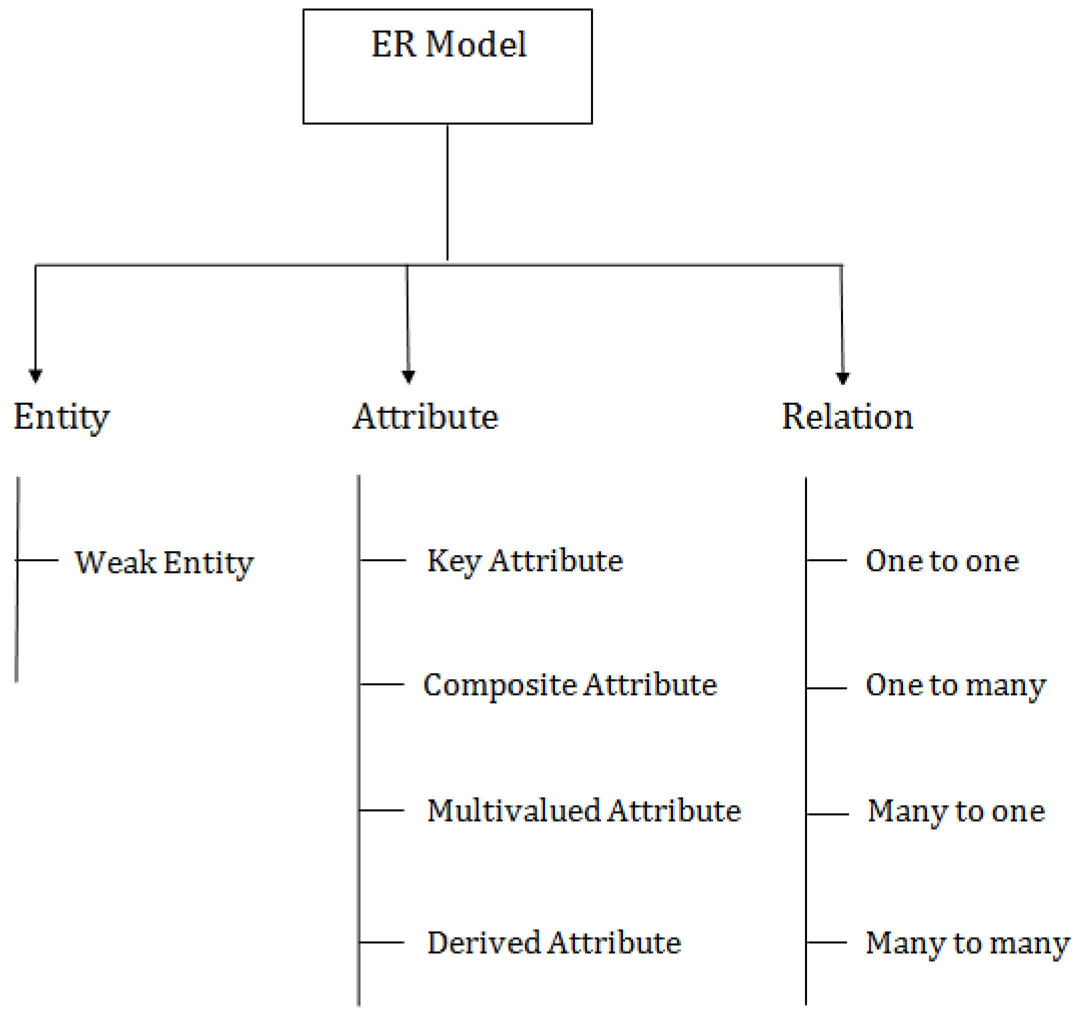
ER (Entity Relationship) Diagram in DBMS

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



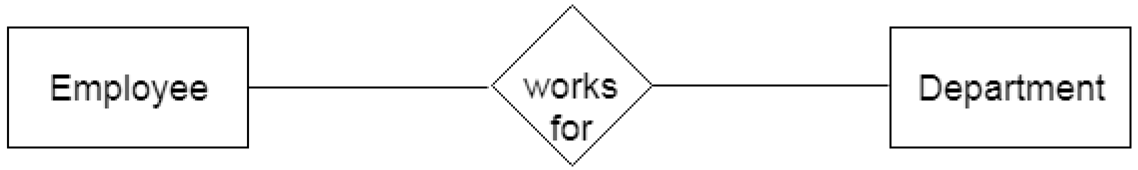
Component of ER Diagram



1. Entity:

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

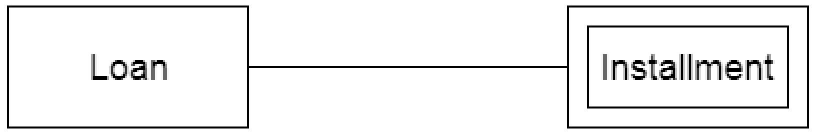
Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



a. Weak Entity

Play Video

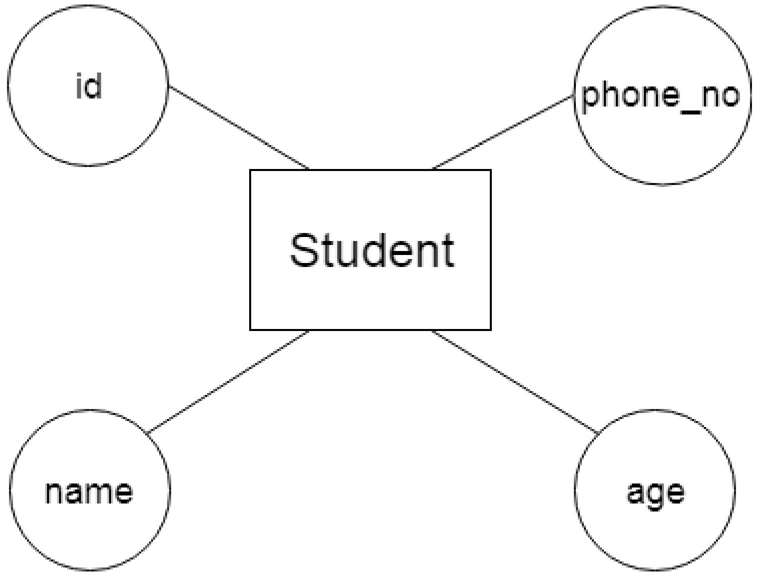
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

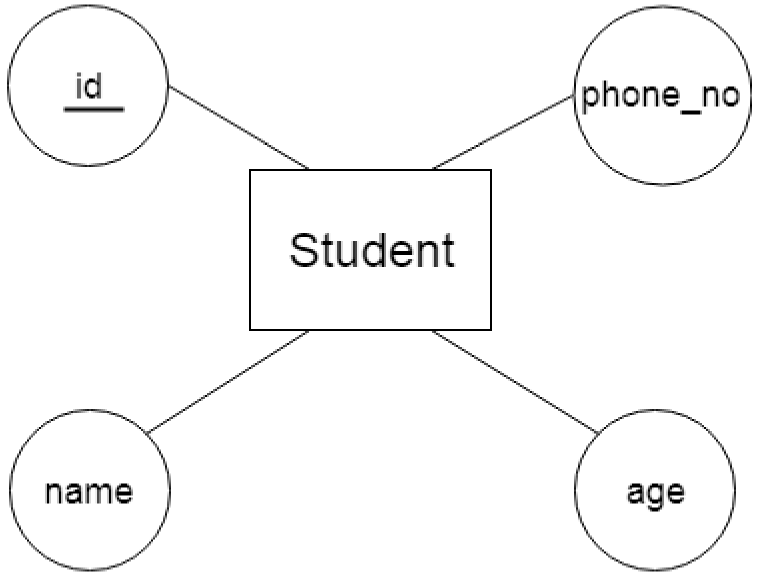
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



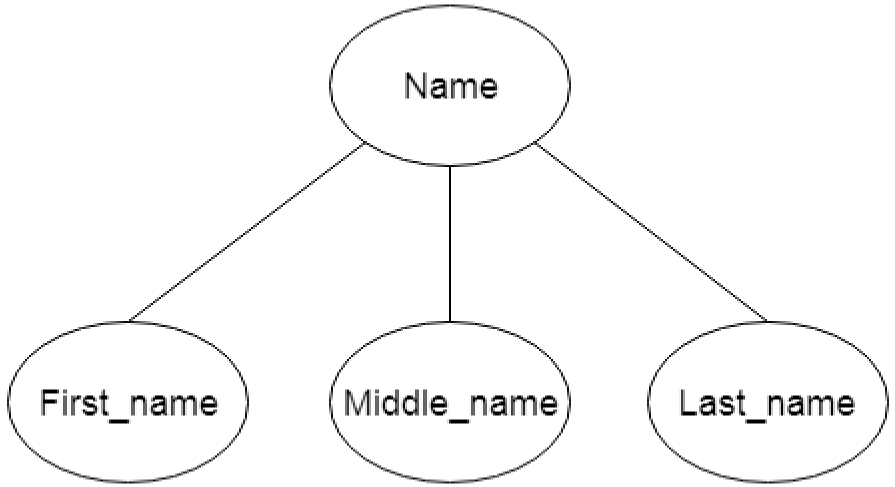
a. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



b. Composite Attribute

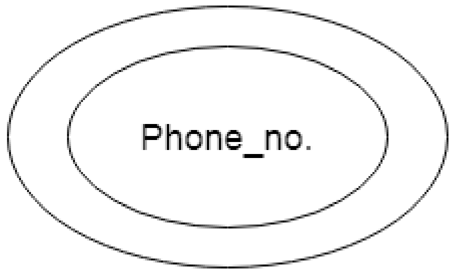
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c. Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

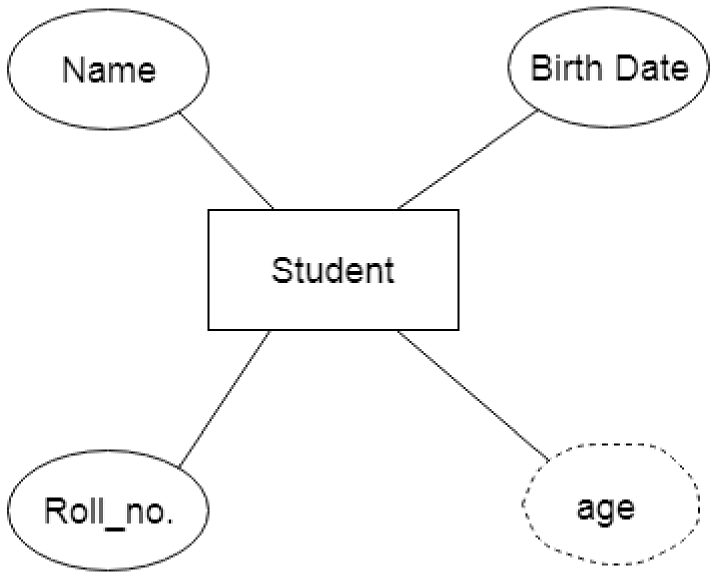
For example, a student can have more than one phone number.



d. Derived Attribute

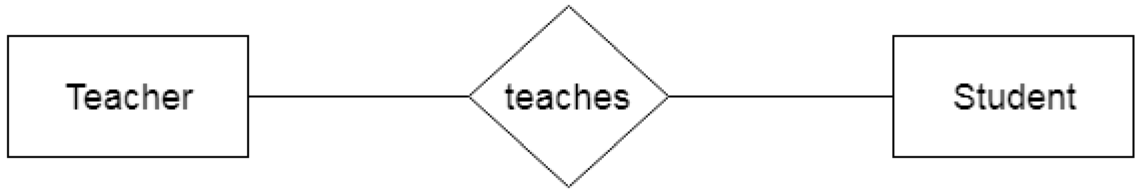
An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

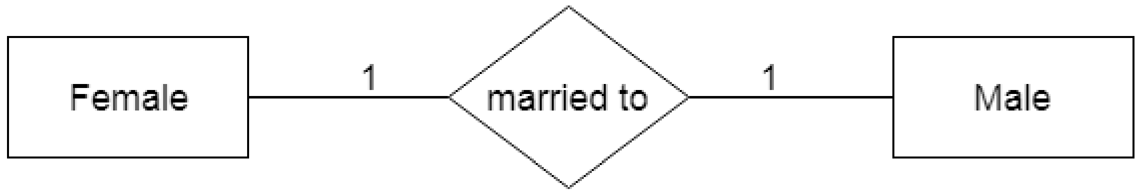


Types of relationship are as follows:

a. One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

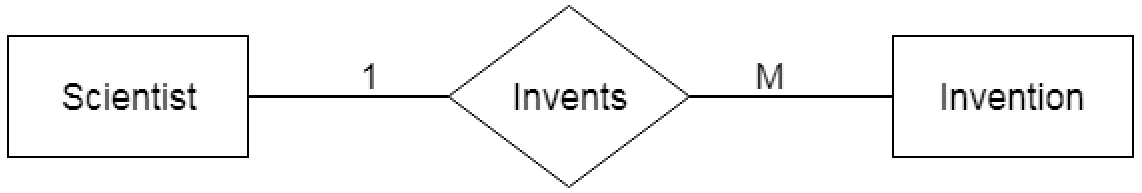
For example, A female can marry to one male, and a male can marry to one female.



b. One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

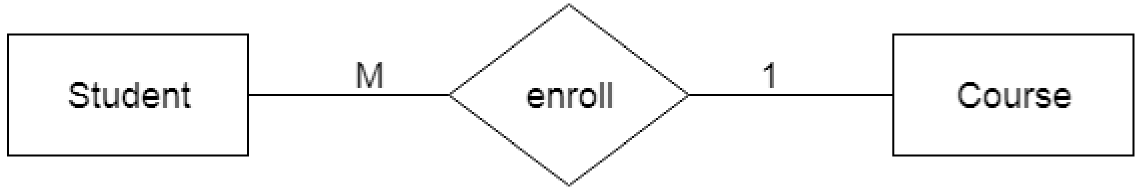
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



c. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

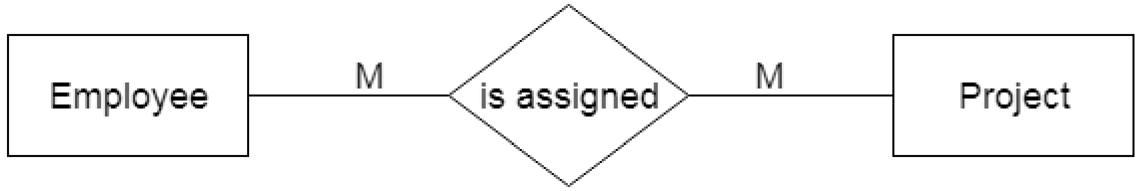
For example, Student enrolls for only one course, but a course can have many students.



d. Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.



Enhanced ER Model

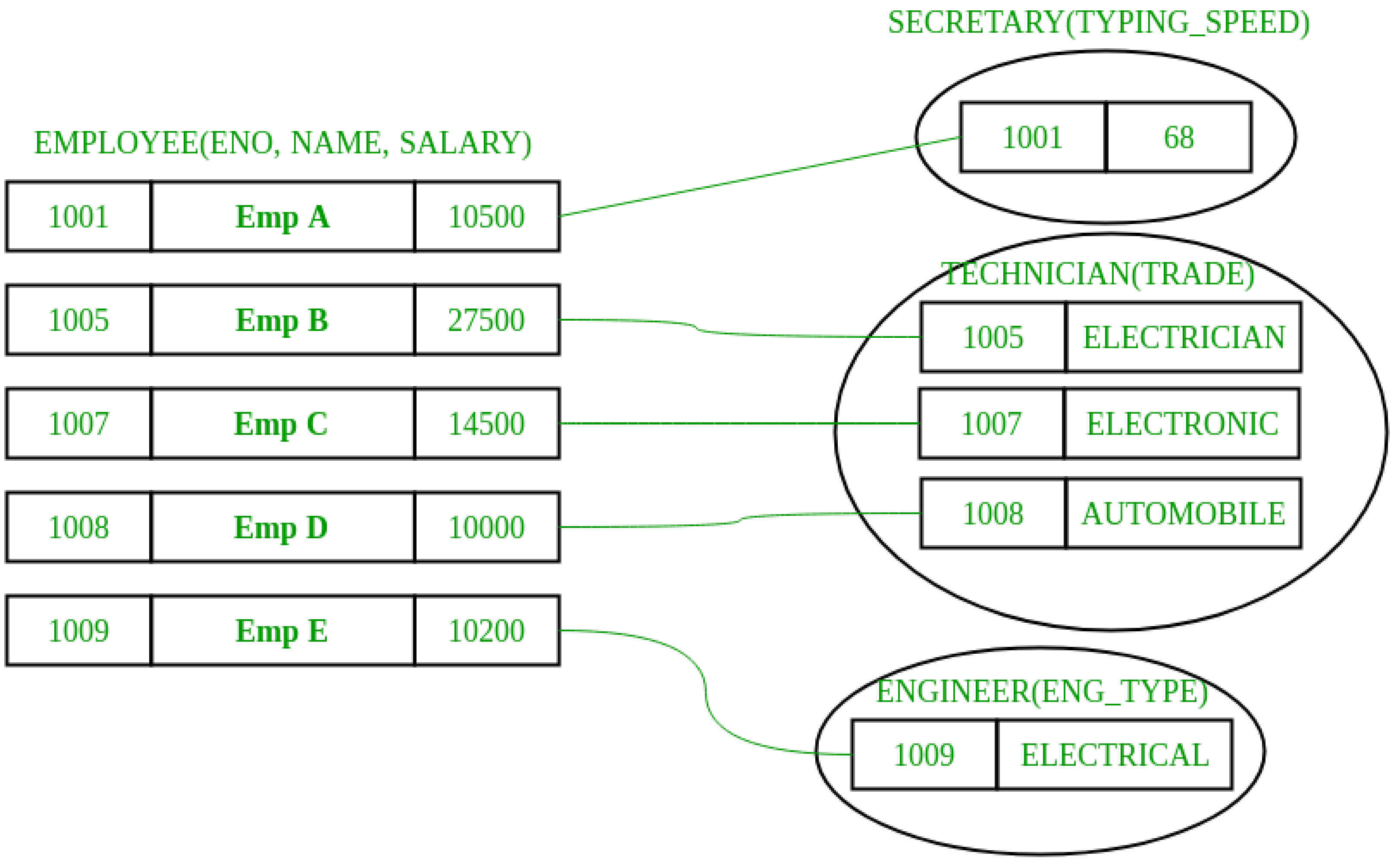
Today the complexity of the data is increasing so it becomes more and more difficult to use the traditional ER model for database modeling. To reduce this complexity of modeling we have to make improvements or enhancements to the existing ER model to make it able to handle the complex application in a better way.

Enhanced entity-relationship diagrams are advanced database diagrams very similar to regular ER diagrams which represent the requirements and complexities of complex databases.

It is a diagrammatic technique for displaying the Sub Class and Super Class; Specialization and Generalization; Union or Category; Aggregation etc.

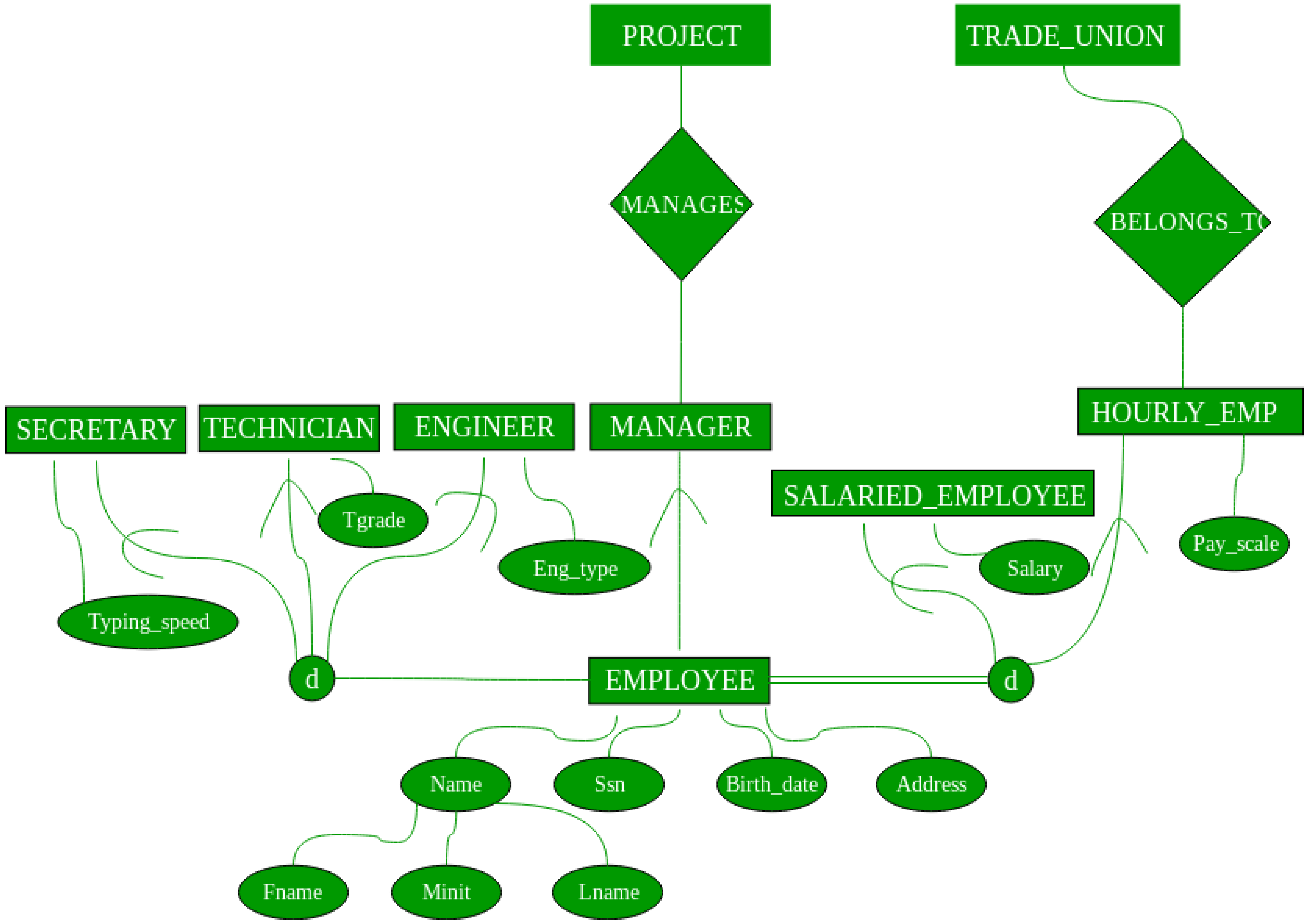
Generalization and Specialization: These are very common relationships found in real entities. However, this kind of relationship was added later as an enhanced extension to the classical ER model. **Specialized** classes are often called **subclass** while a **generalized class** is called a superclass, probably inspired by object-oriented programming. A sub-class is best understood by **"IS-A analysis"**. The following statements hopefully make some sense to your mind "Technician IS-A Employee", and "Laptop IS-A Computer".
An entity is a specialized type/class of another entity. For example, a Technician is a special Employee in a university system Faculty is a special class of Employees. We call this phenomenon generalization/specialization. In the example here Employee is a generalized entity class while the Technician and Faculty are specialized classes of Employee.

Example:
This example instance of **"sub-class"** relationships. Here we have four sets of employees: Secretary, Technician, and Engineer. The employee is a super-class of the rest three sets of individual sub-class is a subset of Employee set.



- An entity belonging to a sub-class is related to some super-class entity. For instance emp, no 1001 is a secretary, and his typing speed is 68. Emp no 1009 is an engineer (sub-class) and her trade is "Electrical", so forth.
- Sub-class entity "inherits" all attributes of super-class; for example, employee 1001 will have attributes eno, name, salary, and typing speed.

Enhanced ER model of above example



Constraints – There are two types of constraints on the "Sub-class" relationship.

1. **Total or Partial –** A sub-classing relationship is total if every super-class entity is to be associated with some sub-class entity, otherwise partial. Sub-class "job type based employee category" is partial sub-classing – not necessary every employee is one of (secretary, engineer, and technician), i.e. union of these three types is a proper subset of all

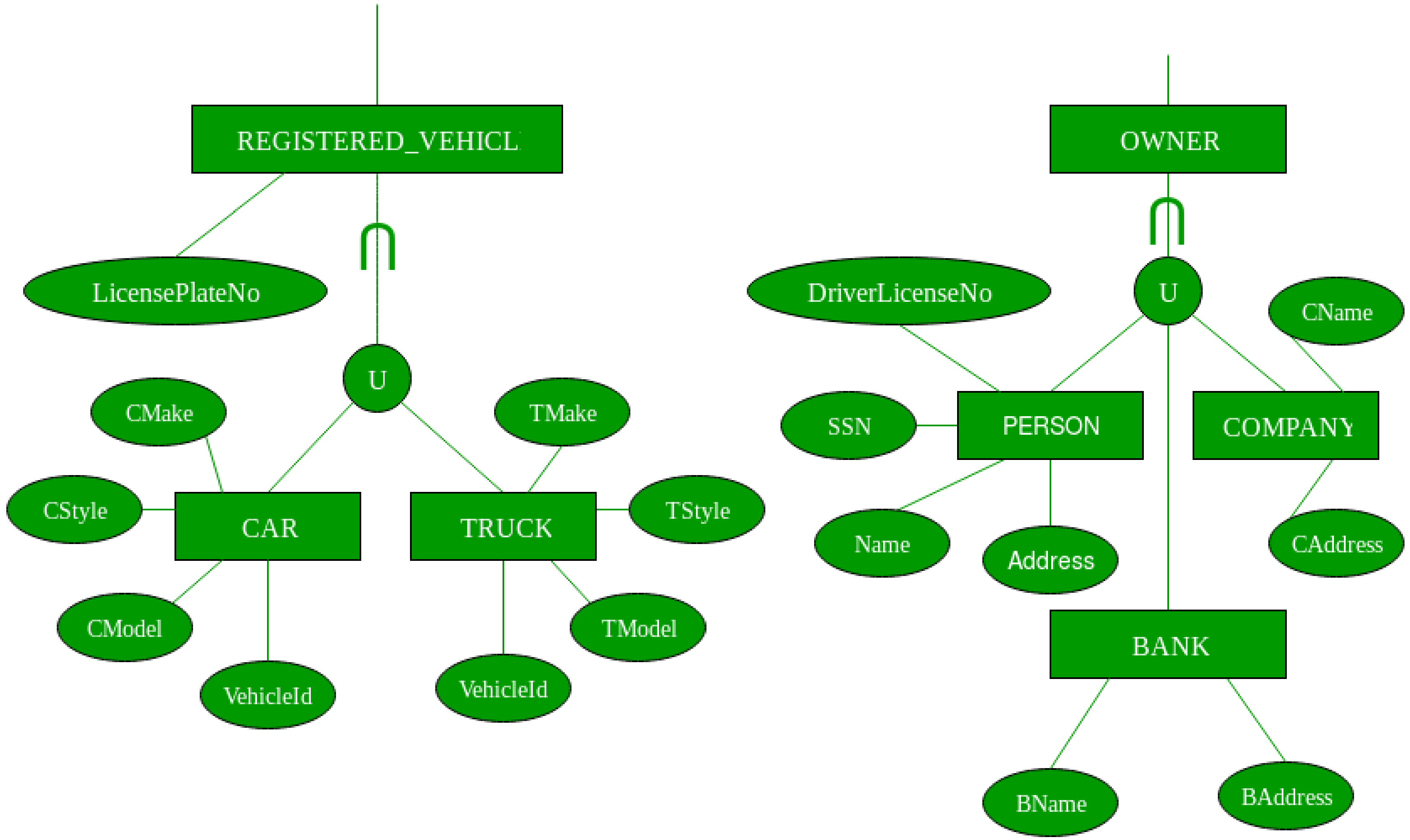
employees. Whereas other sub-classing “Salaried Employee AND Hourly Employee” is total; the union of entities from sub-classes is equal to the total employee set, i.e. every employee necessarily has to be one of them.

2. **Overlapped or Disjoint –** If an entity from a super-set can be related (can occur) in multiple sub-class sets, then it is overlapped sub-classing, otherwise disjoint. Both the examples: job-type based and salaries/hourly employee sub-classing are disjoint.

Note – *These constraints are independent of each other: can be “overlapped and total or partial” or “disjoint and total or partial”. Also, sub-classing has transitive properties.*

Multiple Inheritance (sub-class of multiple superclasses**) –**
An entity can be a sub-class of multiple entity types; such entities are sub-class of multiple entities and have multiple super-classes; Teaching Assistant can subclass of Employee and Student both. A faculty in a university system can be a subclass of Employee and Alumnus. In multiple inheritances, attributes of sub-class are the union of attributes of all super-classes.
Union –

- Set of Library Members is **UNION** of Faculty, Student, and Staff. A union relationship indicates either type; for example, a library member is either Faculty or Staff or Student.
- Below are two examples that show how **UNION** can be depicted in ERD – Vehicle Owner is UNION of PERSON and Company, and RTO Registered Vehicle is UNION of Car and Truck.



You might see some confusion in Sub-class and UNION; consider an example in above figure Vehicle is super-class of CAR and Truck; this is very much the correct example of the subclass as well but here use it differently we are saying RTO Registered vehicle is UNION of Car and Vehicle, they do not inherit any attribute of Vehicle, attributes of car and truck are altogether independent set, where is in sub-classing situation car and truck would be inheriting the attribute of vehicle class.

An Enhanced Entity-Relationship (EER) model is an extension of the original Entity-Relationship (ER) model that includes additional concepts and features to support more complex data modeling requirements. The EER model includes all the elements of the ER model and adds new constructs, such as subtypes and supertypes, generalization and specialization, and inheritance.

Here are some of the key features of the EER model:

- **Subtypes and Supertypes:** The EER model allows for the creation of subtypes and supertypes. A supertype is a generalization of one or more subtypes, while a subtype is a specialization of a supertype. For example, a vehicle could be a supertype, while car, truck, and motorcycle could be subtypes.
- **Generalization and Specialization:** Generalization is the process of identifying common attributes and relationships between entities and creating a supertype based on these common features. Specialization is the process of identifying unique attributes and relationships between entities and creating subtypes based on these unique features.
- **Inheritance:** Inheritance is a mechanism that allows subtypes to inherit attributes and relationships from their supertype. This means that any attribute or relationship defined for a supertype is automatically inherited by all its subtypes.
- **Constraints:** The EER model allows for the specification of constraints that must be satisfied by entities and relationships. Examples of constraints include cardinality constraints, which specify the number of relationships that can exist between entities, and participation constraints, which specify whether an entity is required to participate in a relationship.
- Overall, the EER model provides a powerful and flexible way to model complex data relationships, making it a popular choice for database design. An Enhanced Entity-Relationship (EER) model is an extension of the traditional Entity-Relationship (ER) model that includes additional features to represent complex relationships between entities more accurately. Some of the main features of the EER model are:
- **Subclasses and Superclasses:** EER model allows for the creation of a hierarchical structure of entities where a superclass can have one or more subclasses. Each subclass inherits attributes and relationships from its superclass, and it can also have its unique attributes and relationships.
- **Specialization and Generalization:** EER model uses the concepts of specialization and generalization to create a hierarchy of entities. Specialization is the process of defining subclasses from a superclass, while generalization is the process of defining a superclass from two or more subclasses.
- **Attribute Inheritance:** EER model allows attributes to be inherited from a superclass to its subclasses. This means that attributes defined in the superclass are automatically inherited by all its subclasses.
- **Union Types: E**ER model allows for the creation of a union type, which is a combination of two or more entity types. The union type can have attributes and relationships that are common to all the entity types that make up the union.
- **Aggregation:** EER model allows for the creation of an aggregate entity that represents a group of entities as a single entity. The aggregate entity has its unique attributes and relationships.

- **Multi-valued Attributes:** EER model allows an attribute to have multiple values for a single entity instance. For example, an entity representing a person may have multiple phone numbers.
- **Relationships with Attributes:** EER model allows relationships between entities to have attributes. These attributes can describe the nature of the relationship or provide additional information about the relationship.

Overall, these features make the EER model more expressive and powerful than the traditional ER model, allowing for a more accurate representation of complex relationships between entities.

DBMS Keys:

What are Keys in DBMS?

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. Key is also helpful for finding unique record or row from the table. Database key is also helpful for finding unique record or row from the table.

Example:

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

In the above-given example, employee ID is a primary key because it uniquely identifies an employee record. In this table, no other employee can have the same employee ID.

Why we need a Key?

Here are some reasons for using sql key in the DBMS system.

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

Types of Keys in DBMS (Database Management System)

There are mainly Eight different types of Keys in DBMS and each key has it's different functionality:

1. Super Key
2. Primary Key
3. Candidate Key
4. Alternate Key
5. Foreign Key
6. Compound Key
7. Composite Key
8. Surrogate Key

Let's look at each of the keys in DBMS with example:

- **Super Key –** A super key is a group of single or multiple keys which identifies rows in a table.
- **Primary Key –** is a column or group of columns in a table that uniquely identify every row in that table.
- **Candidate Key –** is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
- **Alternate Key –** is a column or group of columns in a table that uniquely identify every row in that table.
- **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.

- **Compound Key –** has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.
- **Composite Key –** is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individual uniqueness is not guaranteed.
- **Surrogate Key –** An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.

What is the Super key?

A superkey is a group of single or multiple keys which identifies rows in a table. A Super key may have additional attributes that are not needed for unique identification.

Example:

EmpSSN	EmpNum	Empname
9812345098	AB05	Shown
9876512345	AB06	Roslyn
199937890	AB07	James

In the above-given example, EmpSSN and EmpNum name are superkeys.

What is a Primary Key?

PRIMARY KEY in DBMS 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.

Rules for defining Primary key:

- Two rows can't have the same primary key value
- It must for every row to have a primary key value.
- The primary key field cannot be null.

- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

Example:

In the following example, <code>StudID</code> is a Primary 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

What is the Alternate key?

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.
Example:

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

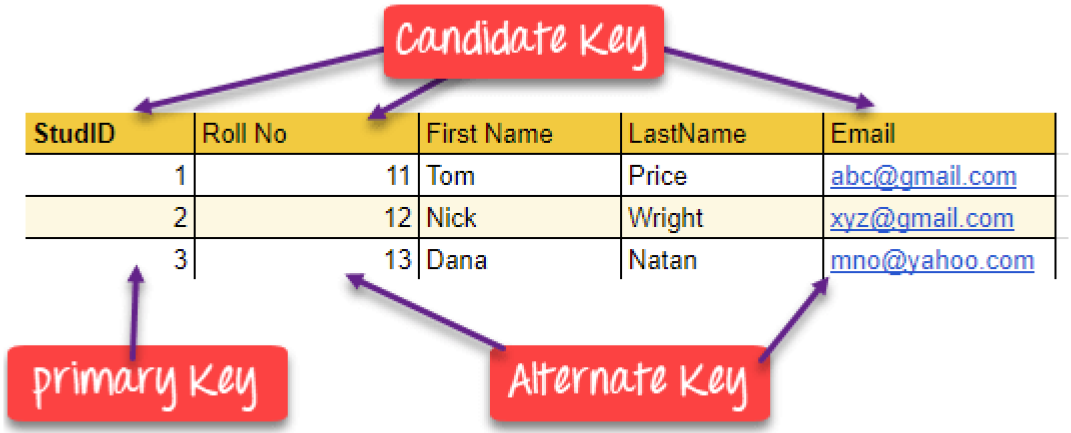
What is a Candidate Key?

CANDIDATE KEY in SQL 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. A table can have multiple candidate keys but only a single primary key.
Properties of Candidate key:

- It must contain unique values
- Candidate key in SQL may have multiple attributes
- Must not contain null values
- It should contain minimum fields to ensure uniqueness
- Uniquely identify each record in a table

Candidate key Example: In the given table Stud ID, Roll No, and email are candidate keys which help us to uniquely identify the student record in the table.

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



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 in DBMS

What is the 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.
Example:

DeptCode	DeptName
001	Science
002	English
005	Computer

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

In this key in dbms example, we have two table, teach and department in a school. However, there is no way to see which search work in which department.

In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

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

This concept is also known as Referential Integrity.

What is the Compound key?

COMPOUND KEY has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database. However, when combined with the other column or columns the combination of composite keys become unique. The purpose of the compound key in database is to uniquely identify each record in the table.
Example:

OrderNo	PorductID	Product Name	Quantity
B005	JAP102459	Mouse	5
B005	DKT321573	USB	10
B005	OMG446789	LCD Monitor	20
B004	DKT321573	USB	15
B002	OMG446789	Laser Printer	3

In this example, OrderNo and ProductID can't be a primary key as it does not uniquely identify a record. However, a compound key of Order ID and Product ID could be used as it uniquely identified each record.

What is the Composite key?

COMPOSITE KEY is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individually uniqueness is not guaranteed. Hence, they are combined to uniquely identify records in a table.
The difference between compound and the composite key is that any part of the compound key can be a foreign key, but the composite key may or maybe not a part of the foreign key.

What is a Surrogate key?

SURROGATE KEYS is An artificial key which aims to uniquely identify each record is called a surrogate key. This kind of partial key in dbms is unique because it is created when you don't have any natural primary key. They do not lend any meaning to the data in the table. Surrogate key in DBMS is usually an integer. A surrogate key is a value generated right before the record is inserted into a table.

Fname	**Lastname**	**Start Time**	**End Time**
Anne	Smith	09:00	18:00
Jack	Francis	08:00	17:00
Anna	McLean	11:00	20:00
Shown	Willam	14:00	23:00

Above, given example, shown shift timings of the different employee. In this example, a surrogate key is needed to uniquely identify each employee.

Surrogate keys in sql are allowed when

- No property has the parameter of the primary key.
- In the table when the primary key is too big or complicated.

Difference Between Primary key & Foreign key

Following is the main difference between primary key and foreign key:

Primary Key	**Foreign Key**
Helps you to uniquely identify a record in the table.	It is a field in the table that is the primary key of another table.
Primary Key never accept null values.	A foreign key may accept multiple null values.
Primary key is a clustered index and data in the DBMS table are physically organized in the sequence of the clustered index.	A foreign key cannot automatically create an index, clustered or non-clustered. However, you can manually create an index on the foreign key.
You can have the single Primary key in a table.	You can have multiple foreign keys in a table.