What is a Database?

A **Database** is a collection of related data organised in a way that data can be easily accessed, managed and updated. Database can be software based or hardware based, with one sole purpose, storing data.

During early computer days, data was collected and stored on tapes, which were mostly write-only, which means once data is stored on it, it can never be read again. They were slow and bulky, and soon computer scientists realised that they needed a better solution to this problem.

Larry Ellison, the co-founder of **Oracle** was amongst the first few, who realised the need for a software based Database Management System.

What is DBMS?

A **DBMS** is a software that allows creation, definition and manipulation of database, allowing users to store, process and analyse data easily. DBMS provides us with an interface or a tool, to perform various operations like creating database, storing data in it, updating data, creating tables in the database and a lot more.

DBMS also provides protection and security to the databases. It also maintains data consistency in case of multiple users.

Here are some examples of popular DBMS used these days:

- MySql
- Oracle
- SQL Server
- IBM DB2
- PostgreSQL
- Amazon SimpleDB (cloud based) etc.

Characteristics of Database Management System

A database management system has following characteristics:

1. **Data stored into Tables:** Data is never directly stored into the database. Data is stored into tables, created inside the database. DBMS also allows to have relationships between tables which makes the data more meaningful and connected. You can easily understand what type of data is stored where by looking at all the tables created in a database.
2. **Reduced Redundancy:** In the modern world hard drives are very cheap, but earlier when hard drives were too expensive, unnecessary repetition of data in database was a big problem. But DBMS follows **Normalisation** which divides the data in such a way that repetition is minimum.
3. **Data Consistency:** On Live data, i.e. data that is being continuosly updated and added, maintaining the consistency of data can become a challenge. But DBMS handles it all by itself.
4. **Support Multiple user and Concurrent Access:** DBMS allows multiple users to work on it(update, insert, delete data) at the same time and still manages to maintain the data consistency.
5. **Query Language:** DBMS provides users with a simple Query language, using which data can be easily fetched, inserted, deleted and updated in a database.
6. **Security:** The DBMS also takes care of the security of data, protecting the data from un-authorised access. In a typical DBMS, we can create user accounts with different access permissions, using which we can easily secure our data by restricting user access.
7. DBMS supports **transactions**, which allows us to better handle and manage data integrity in real world applications where multi-threading is extensively used.

Advantages of DBMS

- Segregation of applicaion program.
- Minimal data duplicacy or data redundancy.
- Easy retrieval of data using the Query Language.
- Reduced development time and maintainance need.
- With Cloud Datacenters, we now have Database Management Systems capable of storing almost infinite data.
- Seamless integration into the application programming languages which makes it very easier to add a database to almost any application or website.

Disadvantages of DBMS

- It's Complexity
- Except MySQL, which is open source, licensed DBMSs are generally costly.
- They are large in size.

- **A Historical Perspective**

- Database Management Systems (DBMS) have been around for several decades, and their history can be traced back to the early 1960s. In the early days, computer systems were designed to manage data in a hierarchical or navigational manner, where data was stored in a tree-like structure. This method of storing data was inefficient and difficult to use, as it required a lot of manual effort to access and manage the data.

- In the late 1960s, The first general-purpose DBMS, designed by **Charles Bachman**, was called the **Integrated Data Store (IDS)** which was based on network data model for which he was received the **Turing Award** (The most prestigious award which is equivalent to Nobel prize in the field of Computer Science.).
- In the late 1970s, **Mr Edgar Codd** proposed a new data representation framework called the **Relational Database Model**. Mr Edgar Codd won the 1981 Turing Award for his seminal work. This model was based on the concept of a table, with rows representing individual records and columns representing individual fields within those records. The relational model allowed for more efficient storage and retrieval of data and was easier to use than the hierarchical or navigational models.
- In the late 1980s IBM developed the **Structured Query Language (SQL)** for relational databases, as a part of R project. This system was designed to manage large amounts of data and was used primarily in corporate and government applications. SQL was adopted by the American National Standards Institute (ANSI) and International Organization for Standardization (ISO).
- In the 1980s, several new DBMS products were introduced, including Oracle, Sybase, and Microsoft SQL Server. These systems were designed to be more user-friendly and to support more advanced data modeling and query languages.
- In the 1990s, **object-oriented DBMS (OODBMS)** emerged, which were designed to store and manage complex data structures, such as multimedia and other types of non-traditional data. These systems were initially popular in research and academic environments, but their adoption was limited in the commercial sector.
- In the 1991, **Microsoft ships MS access**, a personal DBMS and that displaces all other personal DBMS products.
- In the 1997, **XML** applied to database processing. Many vendors begin to integrate XML into DBMS products.
- In the 2000s, web-based applications and cloud computing became more popular, and DBMS systems began to adapt to these new technologies. New DBMS systems were developed to support

distributed and web-based applications, including NoSQL databases such as MongoDB and Cassandra.

- Today, DBMS systems continue to evolve, with an emphasis on scalability, performance, and support for cloud-based applications. Some of the most popular DBMS systems in use today include Oracle, Microsoft SQL Server, MySQL, PostgreSQL, and MongoDB.

File Systems v/s DBMS

What is a File system?

A file system is a technique of arranging the files in a storage devices like a hard disk, pen drive, DVD, etc. It helps you to organizes the data and allows easy retrieval of files when they are required. A file system enables you to handle the way of reading and writing data to the storage medium. It is directly installed into the computer with the Operating systems such as Windows and Linux.

What is DBMS?

Database Management System (DBMS) is a software for storing and retrieving user's 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 DBMS engine to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data.

Difference between File System and Database systems

File System	Database systems
A file system is a software that manages and organizes the files in a storage medium. It controls how data is stored and retrieved.	DBMS or Database Management System is a software application. It is used for accessing, creating, and managing databases.
The file system provides the details of data representation and storage of data.	DBMS gives an abstract view of data that hides the details.
Storing and retrieving of data can’t be done efficiently in a file system.	DBMS is efficient to use as there are a wide variety of methods to store and retrieve data.
Data redundancy is more.	Data redundancy is less.
Conventional file systems are used where there is less demand for security constraints(Security is very low).	Database systems are used when security constraints are high(Security is high).
File systems define the data in un-structured manner. Data is usually in isolated form.	Database systems define the data in a structured manner. Also there is well defined co-relation among the data.
Data inconsistency is more in file systems.	Data inconsistency is less in database systems.

File System	Database systems
User locates the physical address of file to access the data in conventional file systems.	User is unknown to the physical address of the data used in database systems.
There is no ability to concurrently access the data using conventional file system.	There is ability to access the data concurrently using database systems.
Not provide support for complicated transactions.	Easy to implement complicated transactions.
It doesn't offer backup and recovery of data if it is lost.	DBMS system provides backup and recovery of data even if it is lost.

Disadvantages of File system

- Each application has its data file so, the same data may have to be recorded and stored many times.
- Data dependence in the file processing system are data-dependent, but, the problem is incompatible with file format.
- Limited data sharing.
- The problem with security.
- Time-consuming.
- It allows you to maintain the record of the big firm having a large number of items.
- Required lots of labor work to do.

Advantages of DBMS system

- DBMS offers a variety of techniques to store & retrieve data
- Uniform administration procedures for data
- Application programmers 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.
- Reduced Application Development Time
- Consume lesser space.
- Reduction of redundancy.
- Data independence.

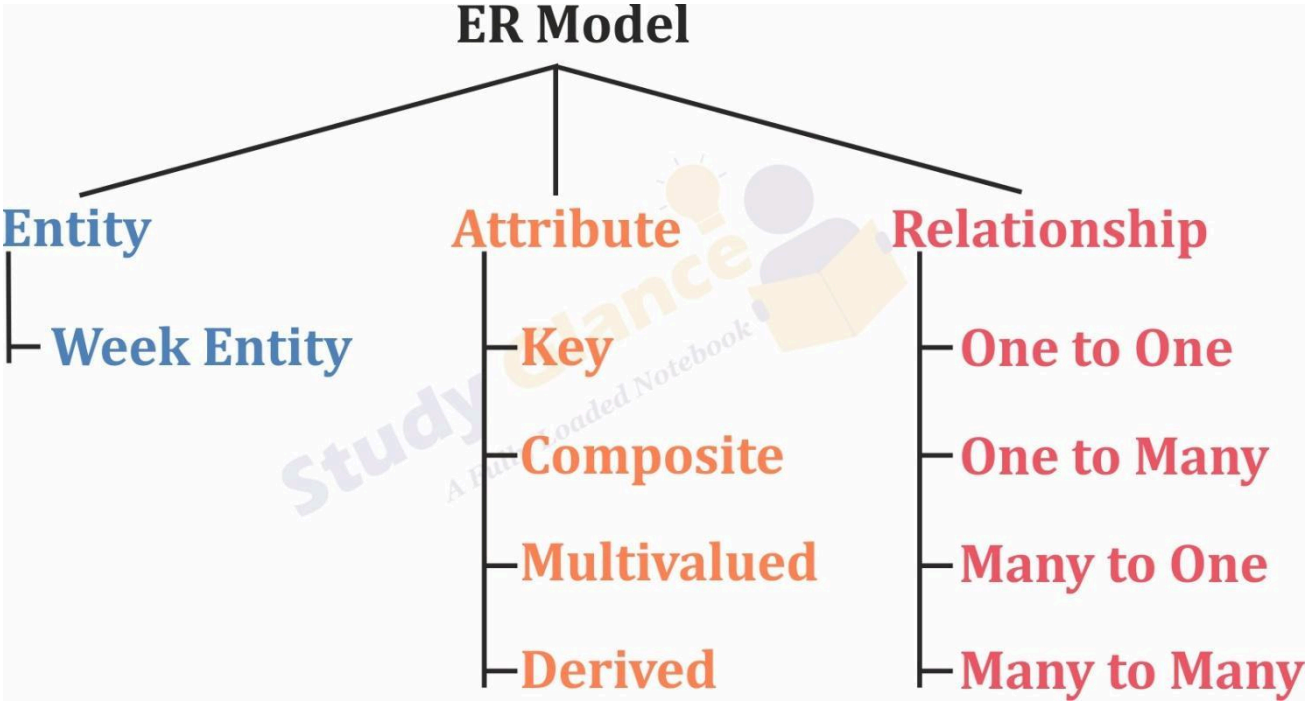
ER model in DBMS is the high-level data model. It stands for the Entity-relationship model and is used to represent a logical view of the system from a data perspective. In simple words, the entity relationship diagram is a blueprint that can used to create a database. E-R diagrams are used to model real-world objects like a person, a car, a company and the relation between these real-world objects.

Features of ER model

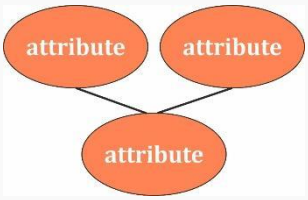
- E-R diagrams are used to represent E-R model in a database, which makes them easy to be converted into relations (tables).
- E-R diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
- E-R diagrams require no technical knowledge and no hardware support.
- These diagrams are very easy to understand and easy to create even by a naive user.
- It gives a standard solution of visualizing the data logically.

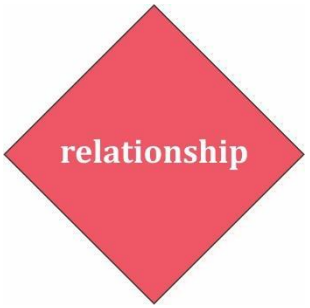
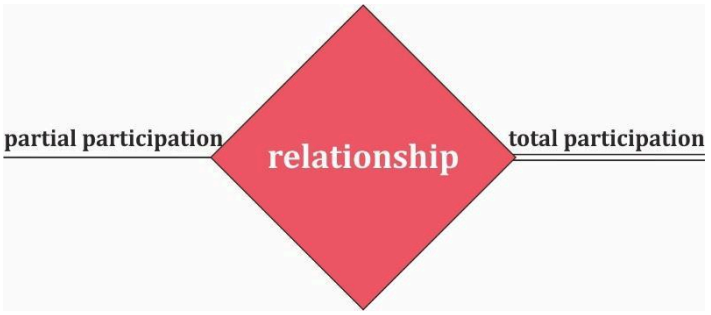
ER diagram basically having three components:

1. **Entities** − It is a real-world thing which can be a person, place, or even a concept. For Example: Department, Admin, Courses, Teachers, Students, Building, etc are some of the entities of a School Management System.
2. **Attributes** − An entity which contains a real-world property called an attribute. For Example: The entity employee has the property like employee id, salary, age, etc.
3. **Relationship** − Relationship tells how two attributes are related. For Example: Employee works for a department.



Component	Symbol
Entity	Entity

Component	Symbol
Weak Entity	
Attribute	
Key Attribute	
Composite Attribute	
Multivalued Attribute	
Derived Attribute	

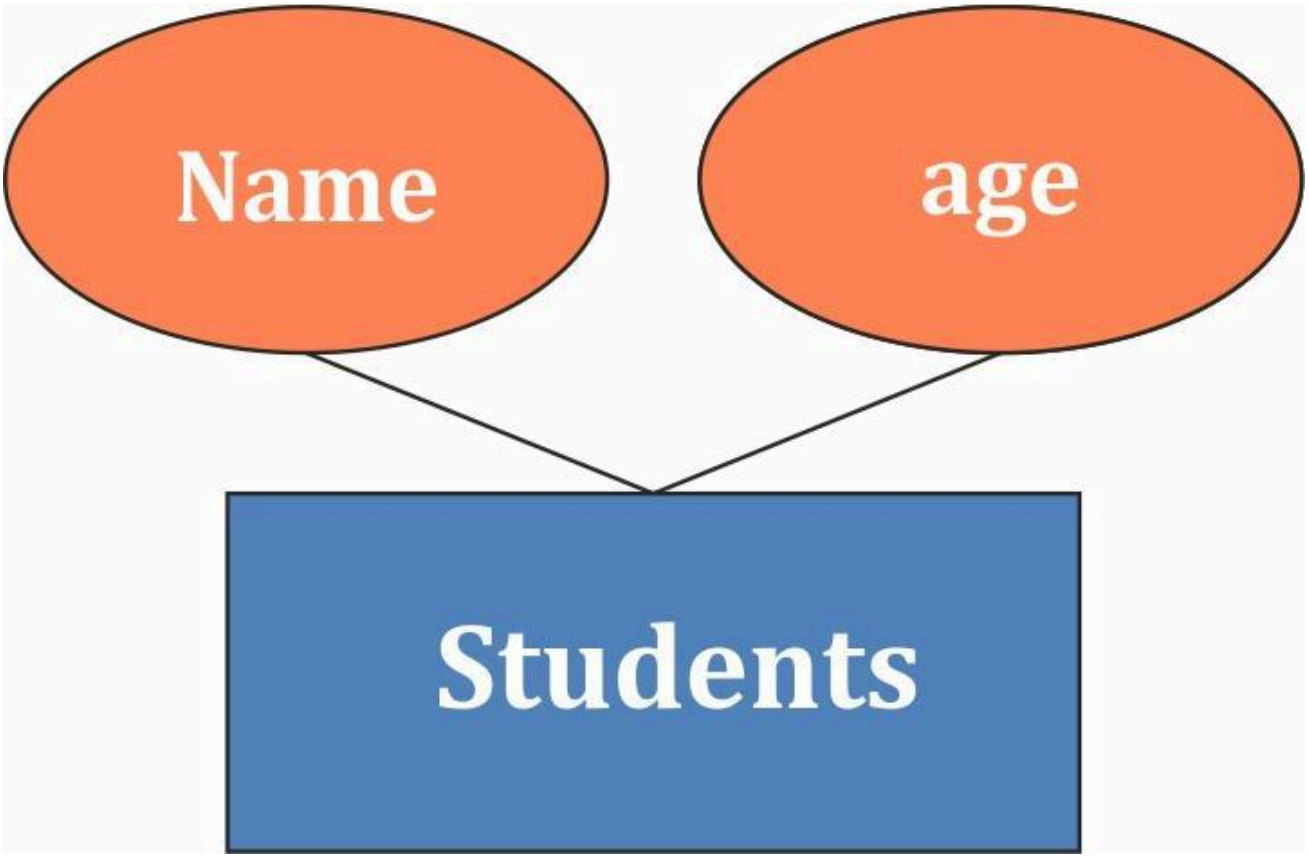
Component	Symbol
Relationship	
Weak Relationship	
Participation Constraints	

Attributes define the properties of a data object of entity. For example if student is an entity, his ID, name, address, date of birth, class are its attributes.

Types of Attributes in DBMS

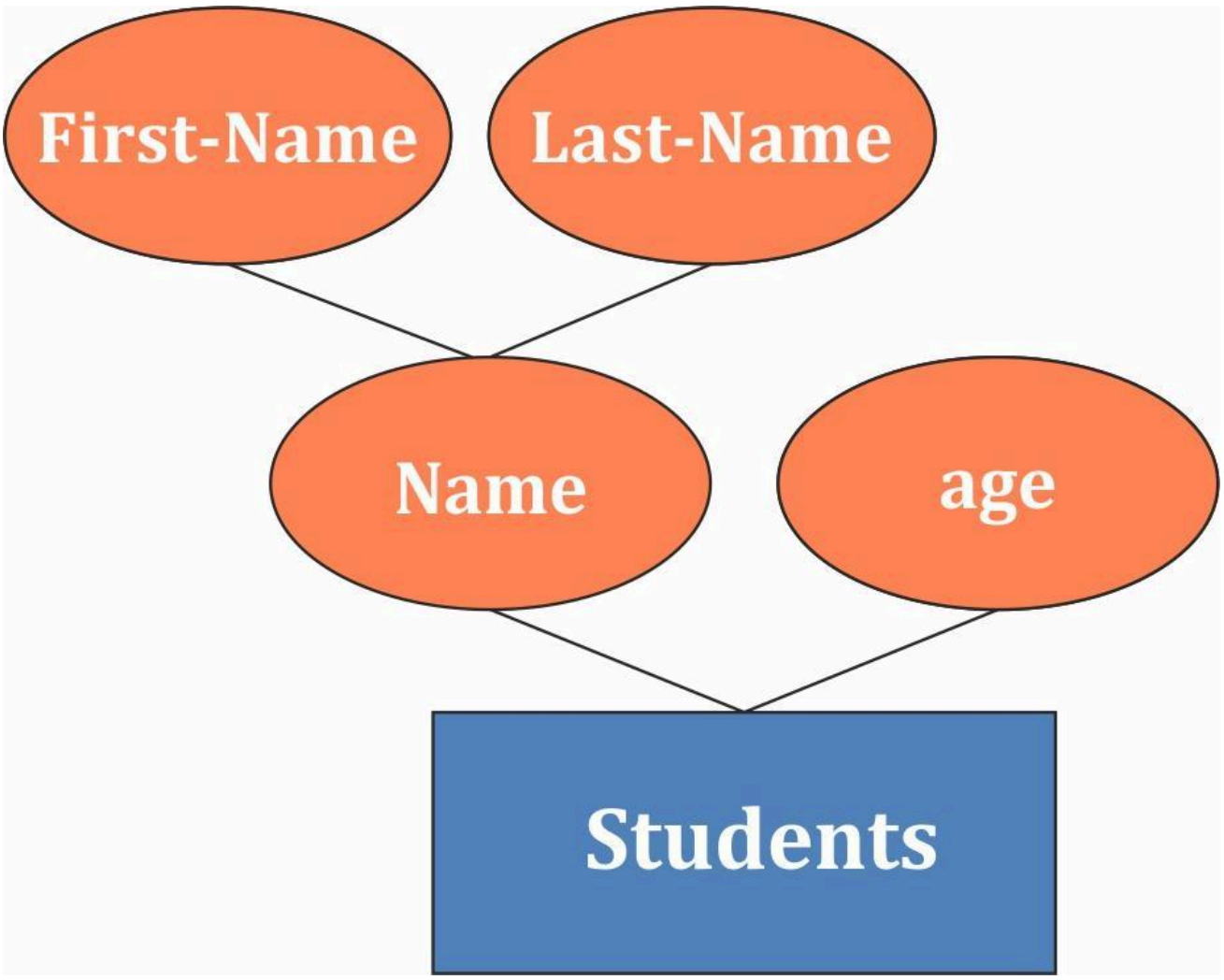
1. Simple Attributes:

Simple attributes are attributes that are drawn from the atomic value domains, which cannot be divided further.



2. Composite attribute:

Composite attributes are those attributes which are composed of many other simple attributes. Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first-name and last-name.

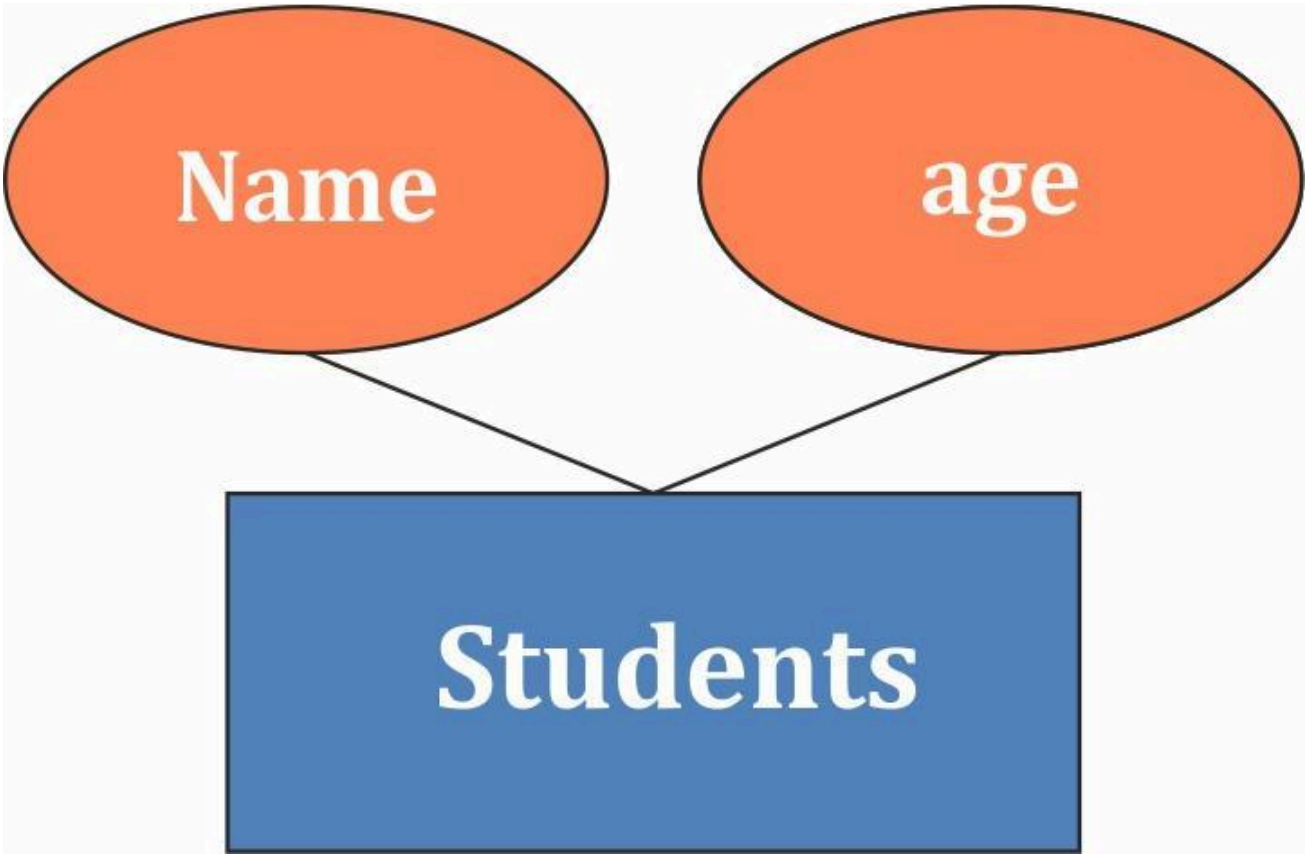


Simple Vs. Composite Attributes

Simple Attribute	Composite Attribute
The attribute which cannot further split into its components is a simple attribute.	The file system provides the details of data representation and storage of data.
Example: The marks of a student, the age of an employee etc.	Example: Name of the student can split into first, middle and last name.

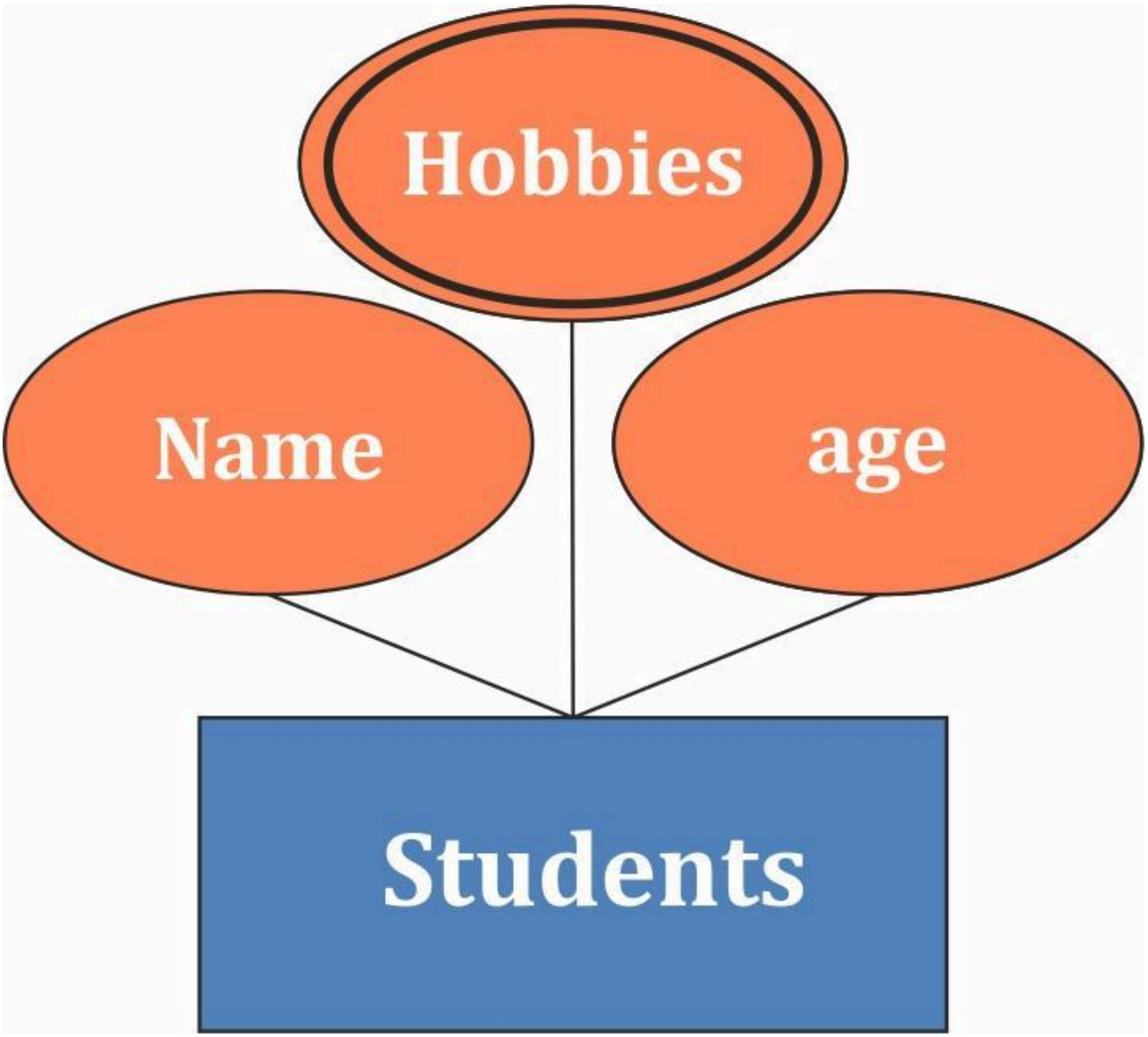
3. Single Valued Attributes:

Single valued attributes are those attributes which can take only one value for a given entity from an entity set. Single-value attributes contain single value. For example − age, gender etc.,.



4. Multi Valued Attributes:

Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set. Multi-value attributes may contain more than one values. For example, a person know more than one Languages, hobbies etc.



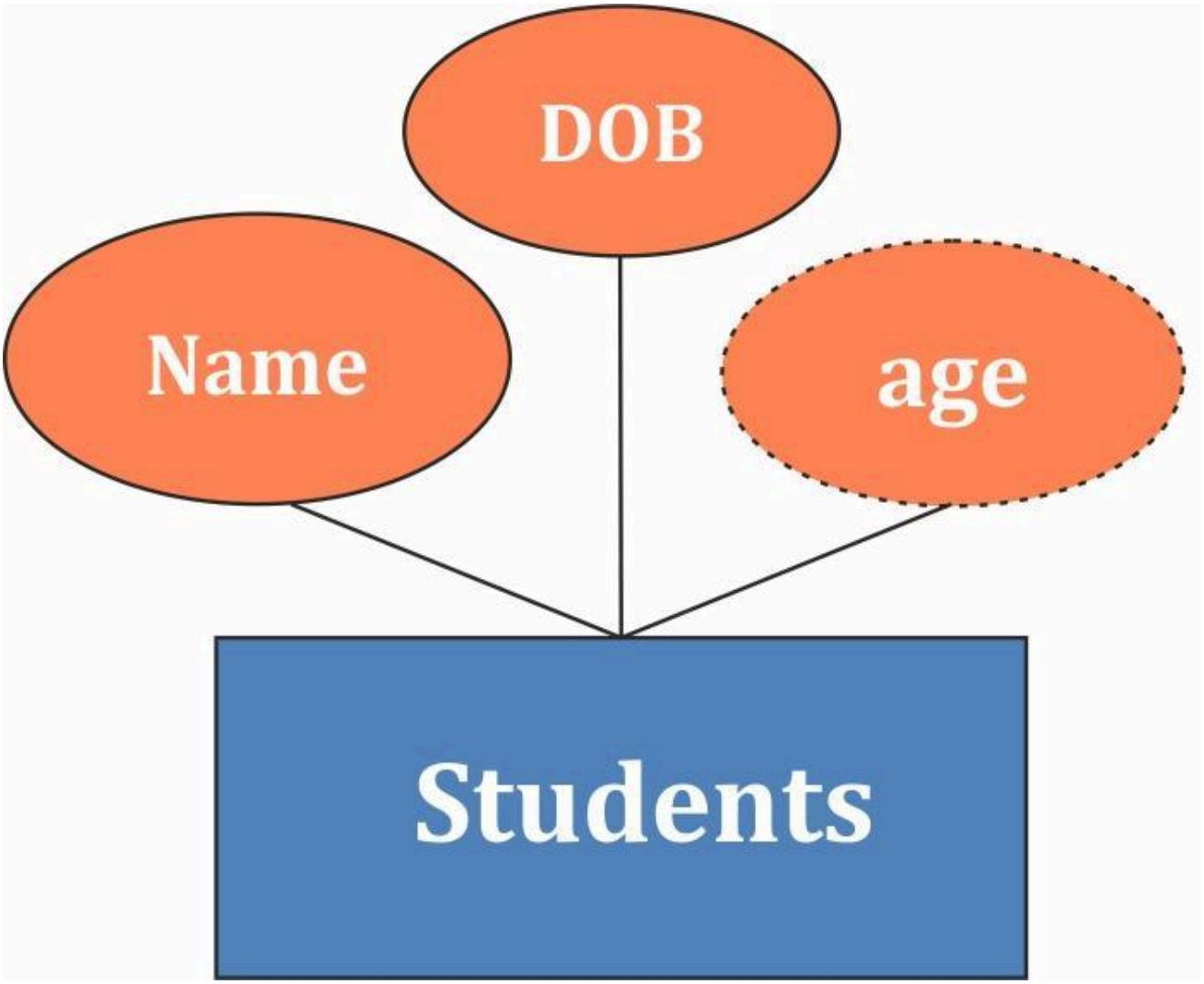
Single-Valued Vs. Multi-Valued Attributes

Single-Valued Attribute	Multi-Valued Attribut
The attribute which has a single value for each entity instance is known as single-valued attribute. There is no alternative of this value.	The attribute which tak known as multi valued
Example: The RollNo, DOB, Gender of a student will always be a single value.	Example: A person kno

5. Derived Attributes:

Derived attributes are those attributes which can be derived from other attribute(s).

Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, age of the student should not be saved directly in the database, instead it can be derived from Date of Birth.



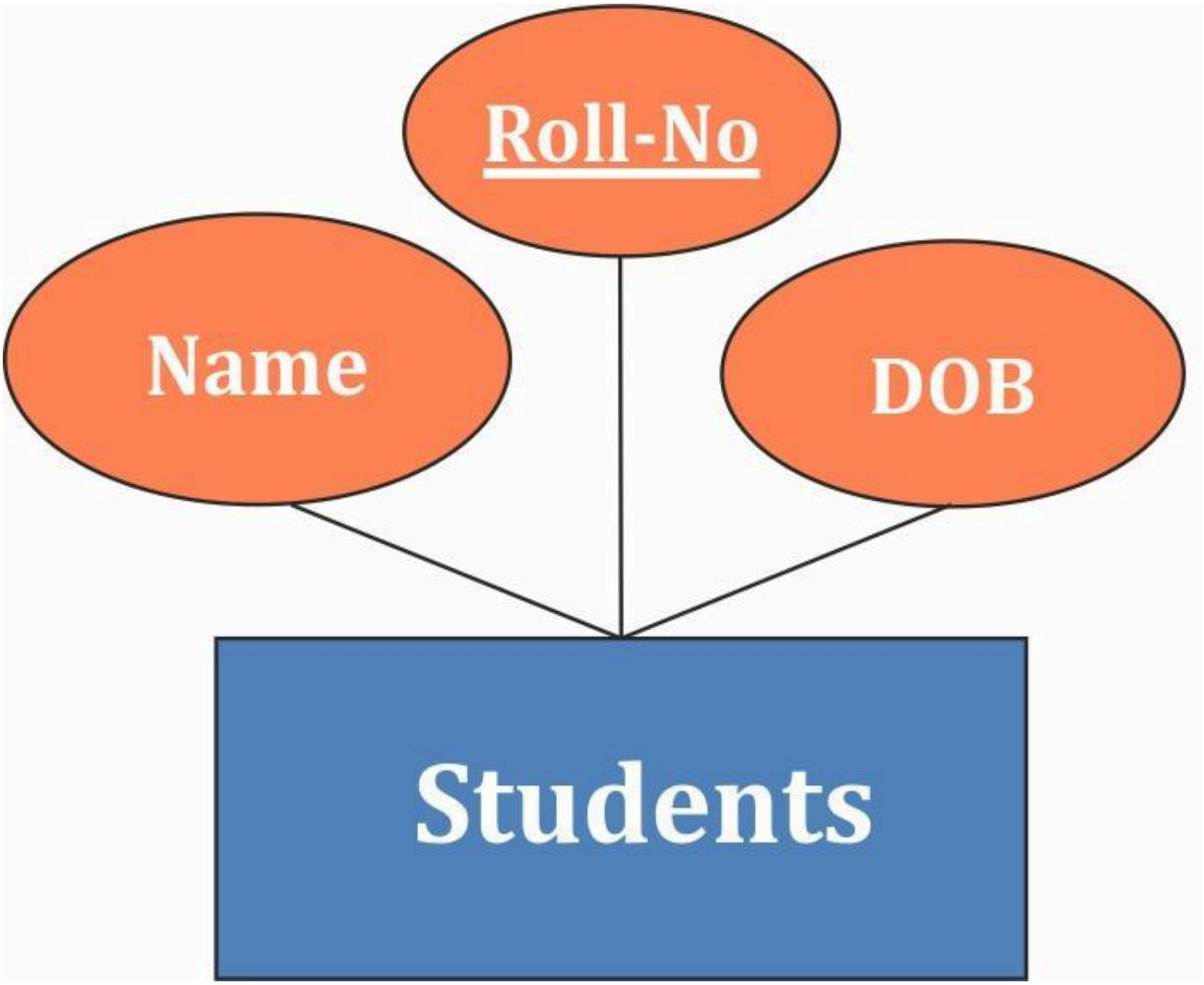
Stored Vs. Derived Attributes

Stored Attribute	Derived Attributes
An attribute which cannot derive from other attributes and it is mandatory to be stored in the database.	An attribute which is determine the value fro mandatory to be stored in the database.
Example: The value which is fixed like Date of Birth.	Example: Age of student can be derived from

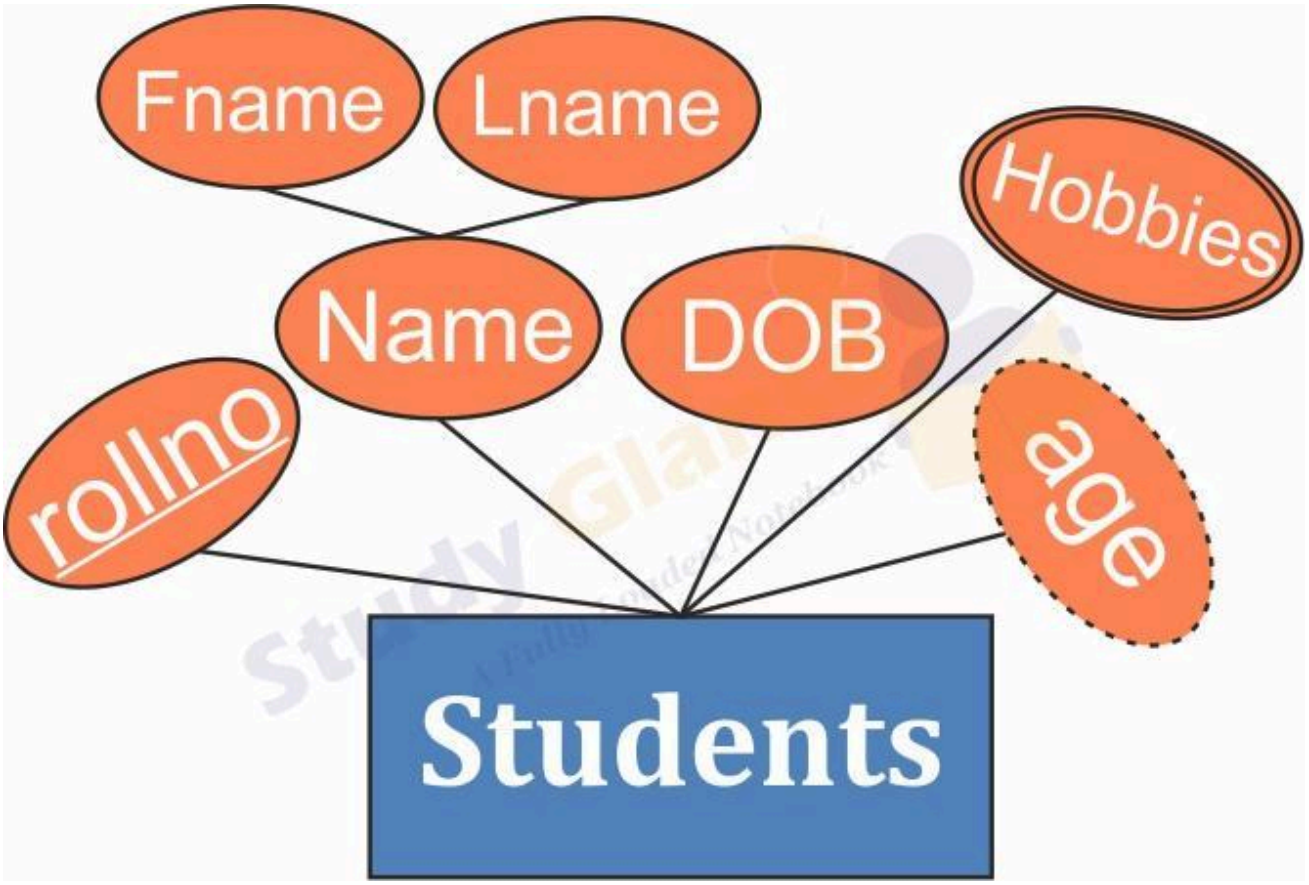
6. Key Attributes:

Key attributes are those attributes which can identify an entity uniquely in an entity set.

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set. For example, the roll_number of a student makes him/her identifiable among students.



Student Entity with all types of Attributes



Entity: An entity is anything in the real world, such as an object, class, person, or place. Objects that physically exist and are logically constructed in the real world are called entities. An entity is distinguishable from other entity.

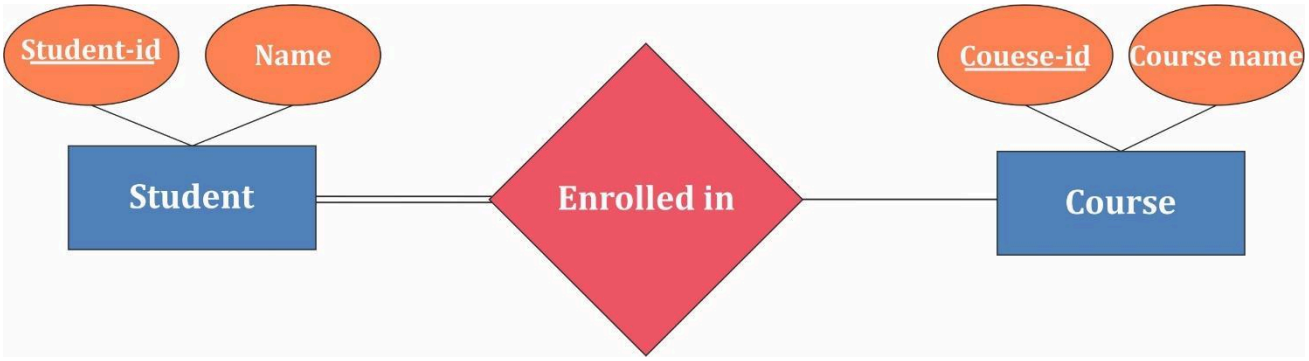
Entity type: The entity type is a collection of the entity having similar attributes.

Entity set: is a group of entities of similar kinds. It can contain entities with attributes that share similar values. It's collectively a group of entities of a similar type. The entity set need not be disjoint.

In summary, an Entity is an object of a Type Entity and the set of all entities is called an entity set.

Entities are of two types:

Strong Entity – A strong entity is an entity type that has a key attribute. It doesn't depend on other entities in the schema. A strong entity always has a primary key, and it is represented by a single rectangle in the ER diagram.



Weak Entity – Weak entity type doesn't have a key attribute and so we cannot uniquely identify them by their attributes alone. Therefore, a foreign key must be used in combination with its attributes to create a primary key. They are called Weak entity types because they can't be identified on their own. It relies on another powerful entity for its unique identity. A weak entity is represented by a double-outlined rectangle in ER diagrams.

Strong Entity v/s Weak Entity

Strong Entity	Weak Entity
Strong entity always has a primary key.	While a weak entity has a partial discriminator k
Strong entity is not dependent on any other entity.	Weak entity depends on strong entity.
Strong entity is represented by a single rectangle.	Weak entity is represented by a double rectangle
Two strong entity's relationship is represented by a single diamond.	While the relation between one strong and one w
Strong entities have either total participation or not.	While weak entity always has total participation

Definition

A relationship is defined as an association among several entities.

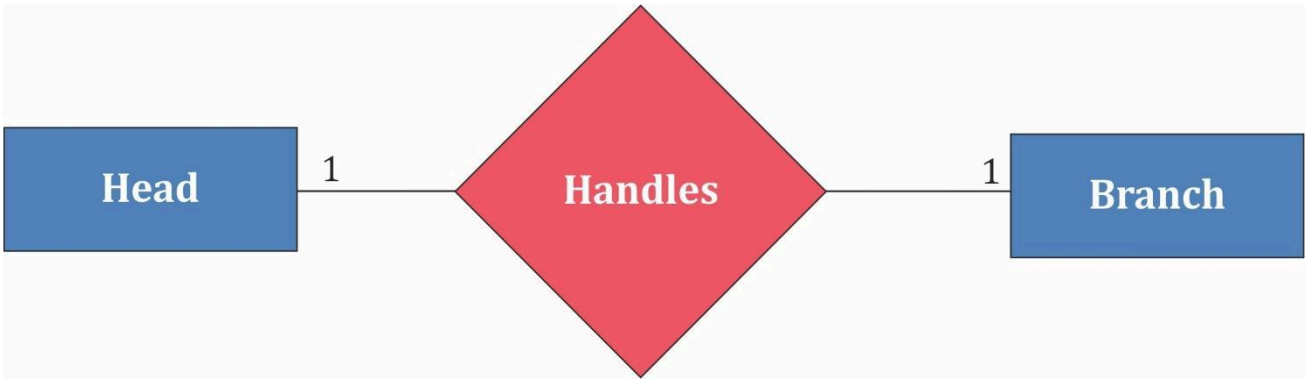
Relationship Set

A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called descriptive attributes.

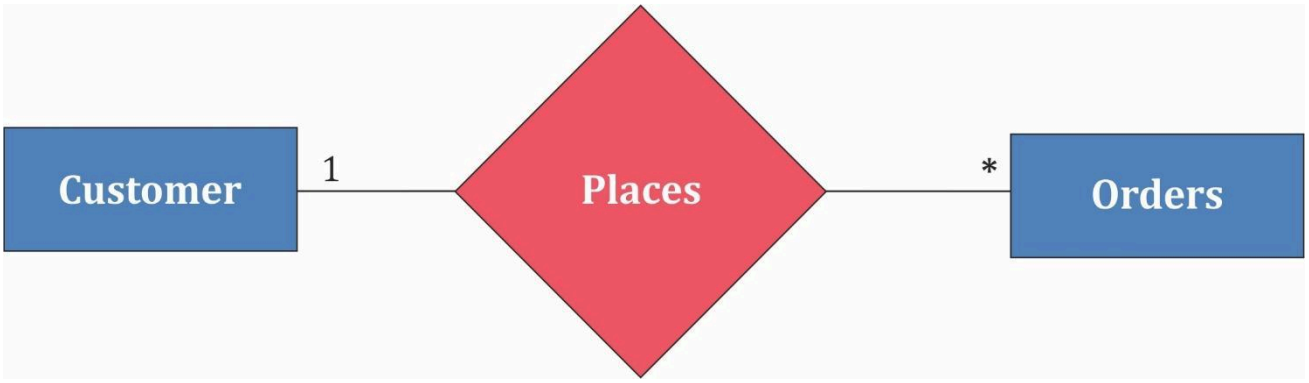
Mapping Cardinalities:

express the number of entities to which another entity can be associated via a relationship. For binary relationship sets between entity sets A and B, the mapping cardinality must be one of:

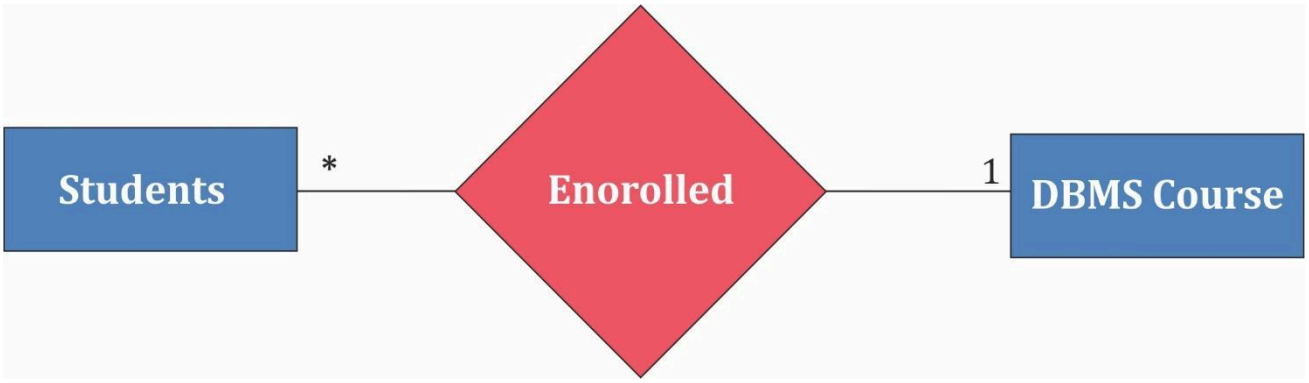
One-to-one: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



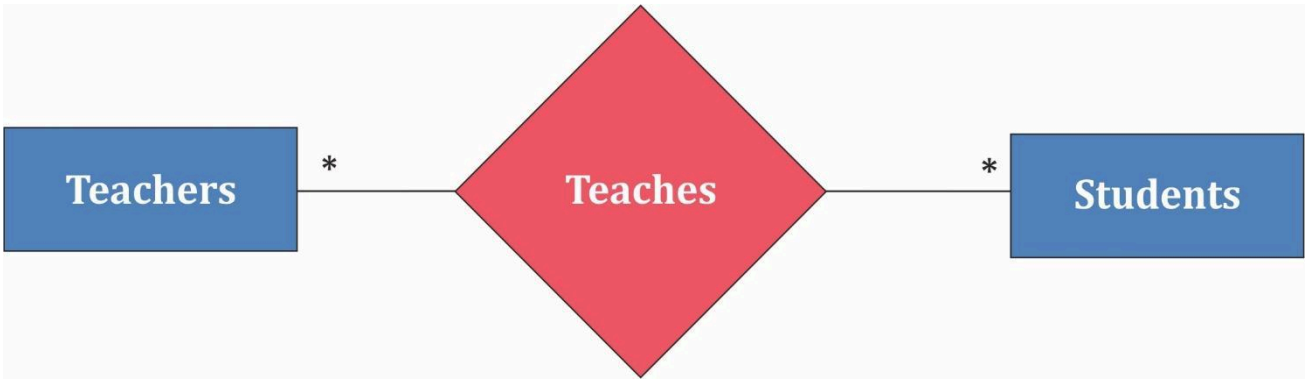
One-to-many: An entity in A is associated with any number in B. An entity in B is associated with at most one entity in A.



Many-to-one: An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.



Many-to-many: Entities in A and B are associated with any number from each other.



The appropriate mapping cardinality for a particular relationship set depends on the real world being modeled.

Degree of a Relationship Set

The number of entity sets that participate in a relationship set is termed as the degree of that relationship set. Thus,

Degree

Degree of a relationship set = Number of entity sets participating in a relationship set

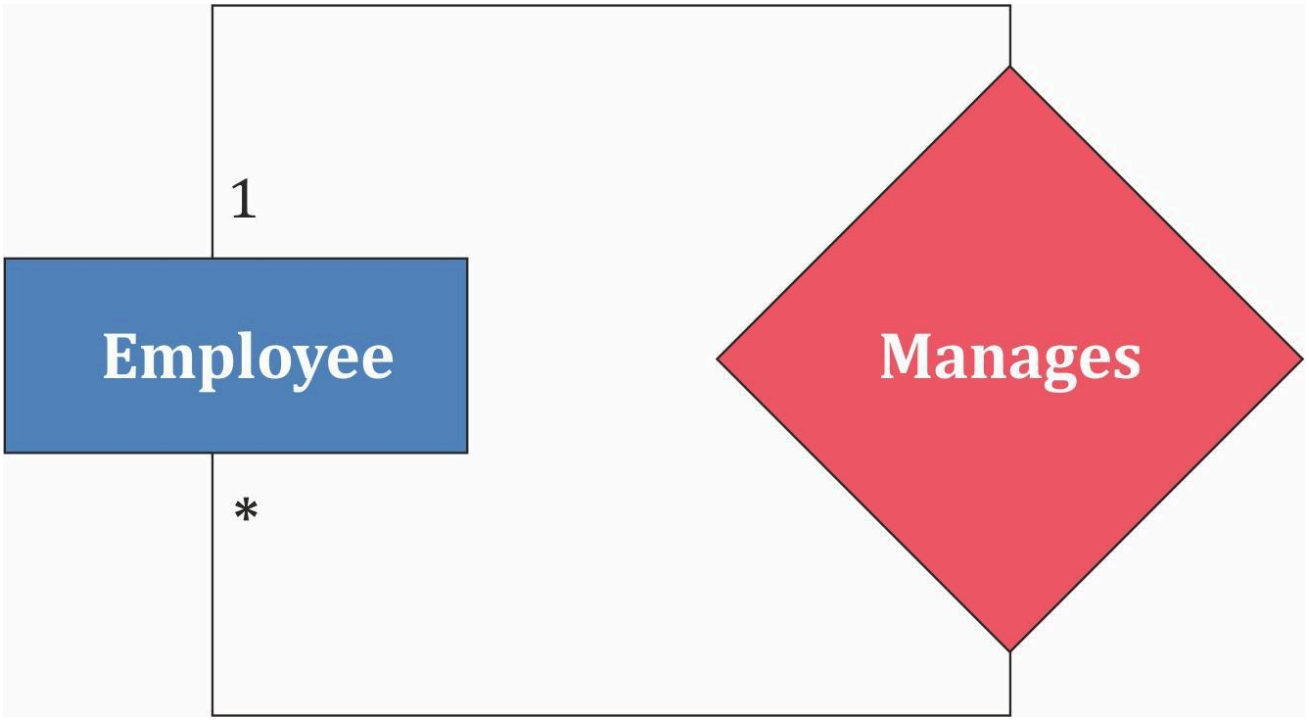
Types of Relationship Sets

On the basis of degree of a relationship set, a relationship set can be classified into the following types-

- **Unary relationship set**
- **Binary relationship set**
- **Ternary relationship set**

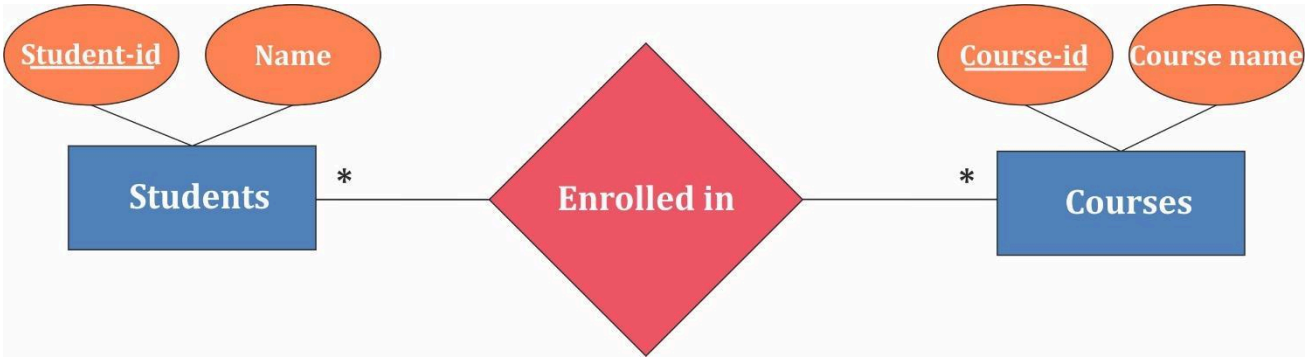
Unary Relationship Set

Unary relationship set is a relationship set where only one entity set participates in a relationship set.



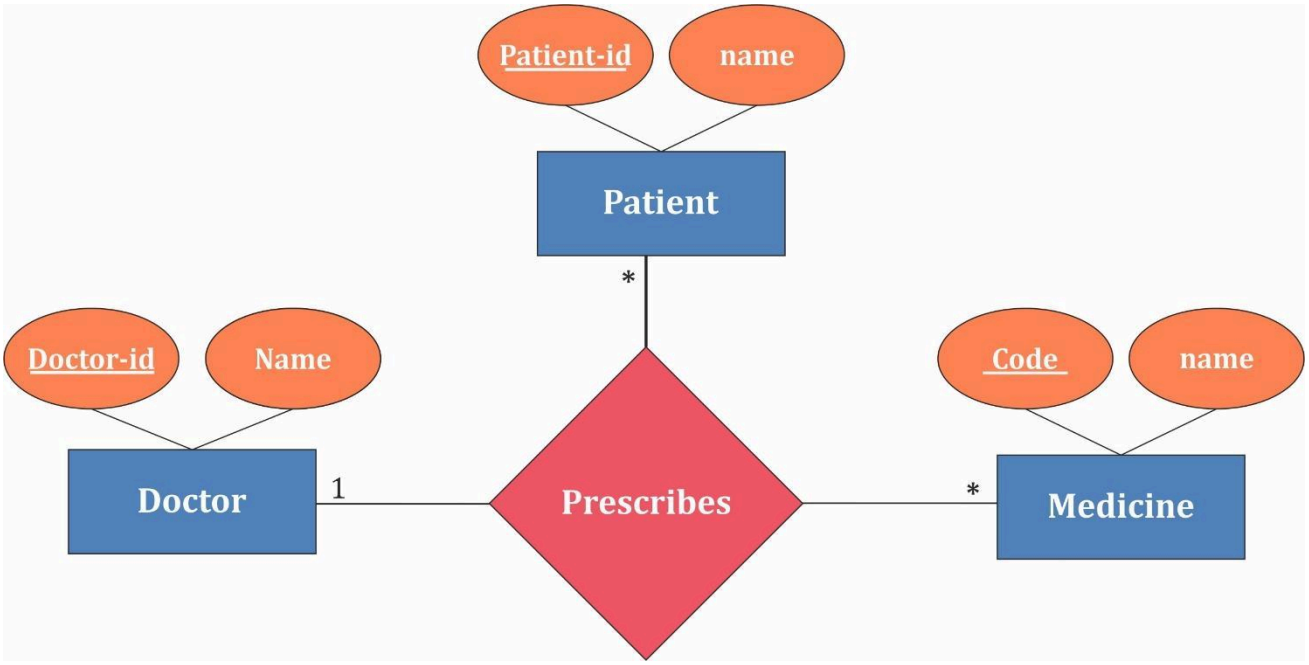
Binary Relationship Set

Binary relationship set is a relationship set where two entity sets participate in a relationship set.



Ternary Relationship Set

Ternary relationship set is a relationship set where three entity sets participate in a relationship set.



Participation Constraints

Total Participation of an Entity set

Total participation of an entity set represents that each entity in entity set must have at least one relationship in a relationship set. It is also called mandatory participation. Total participation is represented using a double line between the entity set and relationship set.

Partial participation of an Entity Set

Partial participation of an entity set represents that each entity in the entity set may or may not participate in the relationship instance in that relationship set. It is also called as optional participation. Partial participation is represented using a single line between the entity set and relationship set.