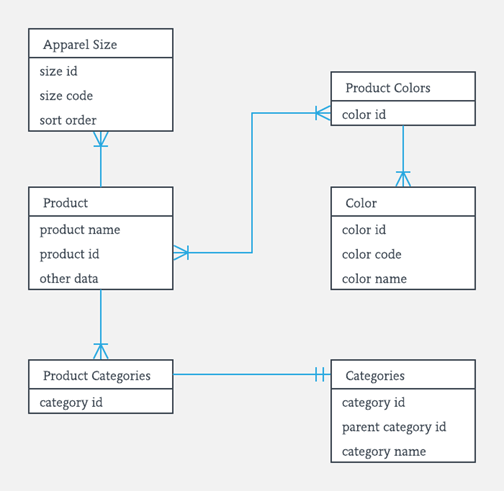
**What is ER Diagram?**

**ER Diagram** stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: **entities, attributes and relationships.**

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.



**What is ER Model?**

**ER Model** stands for Entity Relationship Model is a high-level conceptual data model diagram. ER model helps to systematically analyze data requirements to produce a well-designed database. The ER Model represents real-world entities and the relationships between them. Creating an ER Model in DBMS is considered as a best practice before implementing your database.

ER Modeling helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.

**History of ER models**

ER diagrams are visual tools that are helpful to represent the ER model. Peter Chen proposed ER Diagram in 1971 to create a uniform convention that can be used for relational databases and networks. He aimed to use an ER model as a conceptual modeling approach.

**Why use ER Diagrams?**

Here, are prime reasons for using the ER Diagram

* Helps you to define terms related to entity relationship modeling
* Provide a preview of how all your tables should connect, what fields are going to be on each table
* Helps to describe entities, attributes, relationships
* ER diagrams are translatable into relational tables which allows you to build databases quickly
* ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
* The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
* ERD Diagram allows you to communicate with the logical structure of the database to users

**Facts about ER Diagram Model**

**Now in this ERD Diagram Tutorial, let’s check out some interesting facts about ER**

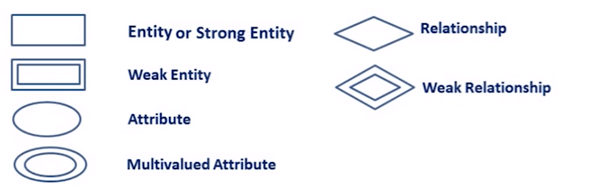
**Diagram Model:**

* ER model allows you to draw Database Design
* It is an easy to use graphical tool for modeling data
* Widely used in Database Design
* It is a GUI representation of the logical structure of a Database
* It helps you to identifies the entities which exist in a system and the relationships between those entities

**ER Diagrams Symbols & Notations**

**Entity Relationship Diagram Symbols & Notations** mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes. There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.

**Following are the main components and its symbols in ER Diagrams:**

* **Rectangles:**This Entity Relationship Diagram symbol represents entity types
* **Ellipses:**Symbol represent attributes
* **Diamonds:**This symbol represents relationship types
* **Lines:**It links attributes to entity types and entity types with other relationship types
* **Primary key:**attributes are underlined
* **Double Ellipses:**Represent multi-valued attributes
* 

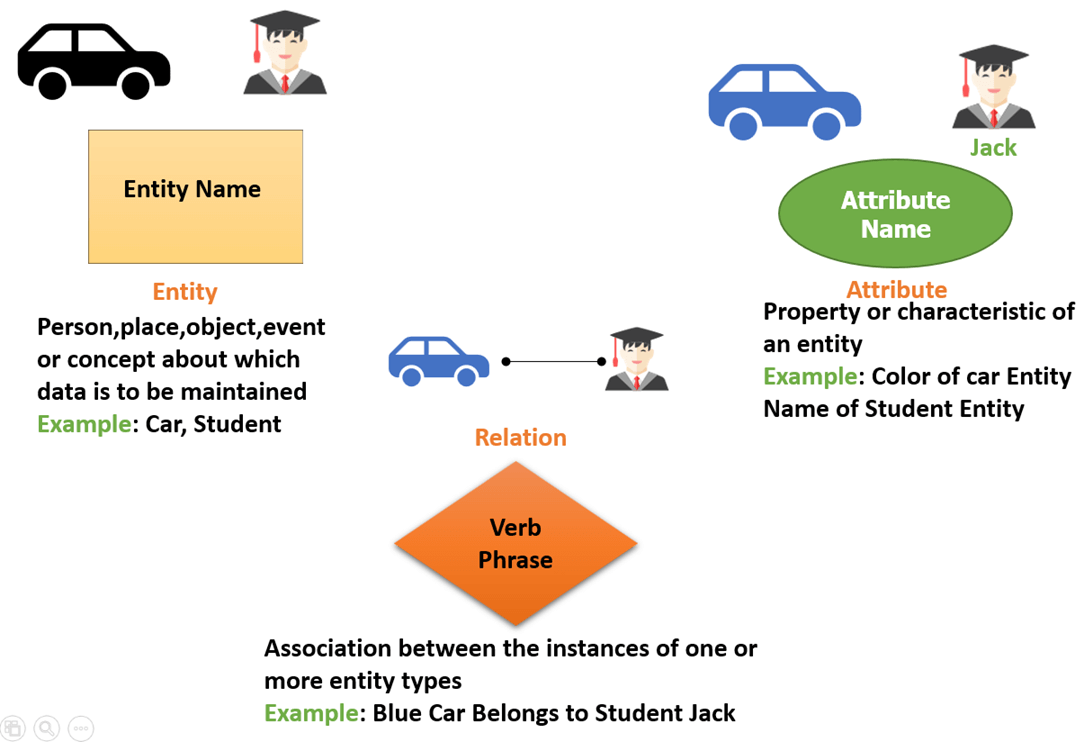
**Components of the ER Diagram**

This model is based on three basic concepts:

* Entities
* Attributes
* Relationships

**ER Diagram Examples**

For example, in a University database, we might have entities for Students, Courses, and Lecturers. Students entity can have attributes like Rollno, Name, and DeptID. They might have relationships with Courses and Lecturers.



**WHAT IS ENTITY?**

A real-world thing either living or non-living that is easily recognizable and nonrecognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.

An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some ‘attributes’ which represent that entity.

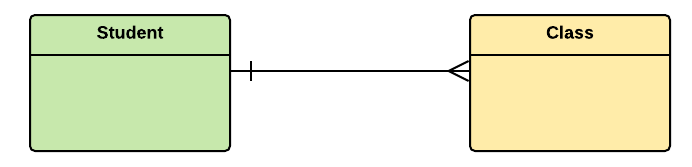
**Examples of entities:**

* **Person:** Employee, Student, Patient
* **Place:** Store, Building
* **Object:** Machine, product, and Car
* **Event:** Sale, Registration, Renewal
* **Concept:** Account, Course

**Entity set**

Student

An entity set is a group of similar kind of entities. It may contain entities with attribute sharing similar values. Entities are represented by their properties, which also called attributes. All attributes have their separate values. For example, a student entity may have a name, age, class, as attributes.



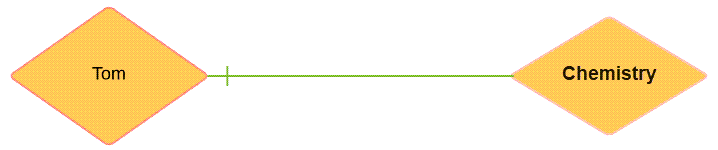
**Example of Entities:**

A university may have some departments. All these departments employ various lecturers and offer several programs.

Some courses make up each program. Students register in a particular program and enroll in various courses. A lecturer from the specific department takes each course, and each lecturer teaches a various group of students.

**Relationship**

Relationship is nothing but an association among two or more entities. E.g., Tom works in the Chemistry department.

[](https://www.guru99.com/images/1/100518_0621_ERDiagramTu4.png)

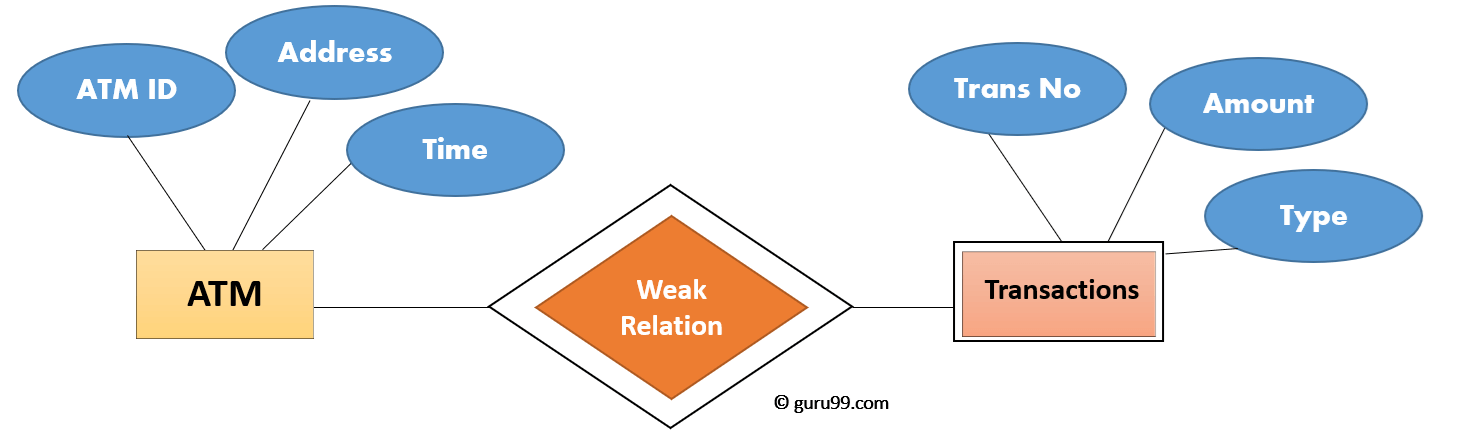
Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

**For example:**

* You are attending this lecture
* I am giving the lecture
* Just loke entities, we can classify relationships according to relationship-types:
* A student attends a lecture
* A lecturer is giving a lecture.

**Weak Entities**

A weak entity is a type of entity which doesn’t have its key attribute. It can be identified uniquely by considering the primary key of another entity. For that, weak entity sets need to have participation.

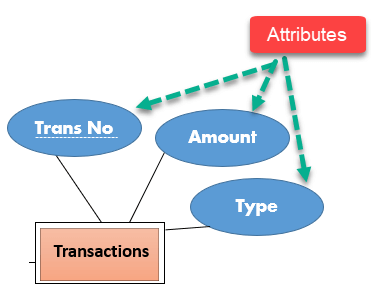


In above ER Diagram examples, “Trans No” is a discriminator within a group of transactions in an ATM.

| **Strong Entity Set** | **Weak Entity Set** |
| --- | --- |
| Strong entity set always has a primary key. | It does not have enough attributes to build a primary key. |
| It is represented by a rectangle symbol. | It is represented by a double rectangle symbol. |
| It contains a Primary key represented by the underline symbol. | It contains a Partial Key which is represented by a dashed underline symbol. |
| The member of a strong entity set is called as dominant entity set. | The member of a weak entity set called as a subordinate entity set. |
| Primary Key is one of its attributes which helps to identify its member. | In a weak entity set, it is a combination of primary key and partial key of the strong entity set. |
| In the ER diagram the relationship between two strong entity set shown by using a diamond symbol. | The relationship between one strong and a weak entity set shown by using the double diamond symbol. |
| The connecting line of the strong entity set with the relationship is single. | The line connecting the weak entity set for identifying relationship is double. |

**Attributes**

It is a single-valued property of either an entity-type or a relationship-type. For example, a lecture might have attributes: time, date, duration, place, etc. An attribute in ER Diagram examples, is represented by an Ellipse



| **Types of Attributes** | **Description** |
| --- | --- |
| **Simple attribute** | Simple attributes can’t be divided any further. For example, a student’s contact number. It is also called an atomic value. |
| **Composite attribute** | It is possible to break down composite attribute. For example, a student’s full name may be further divided into first name, second name, and last name. |
| **Derived attribute** | This type of attribute does not include in the physical database. However, their values are derived from other attributes present in the database. For example, age should not be stored directly. Instead, it should be derived from the DOB of that employee. |
| **Multivalued attribute** | Multivalued attributes can have more than one values. For example, a student can have more than one mobile number, email address, etc. |

**Cardinality**

Defines the numerical attributes of the relationship between two entities or entity sets.

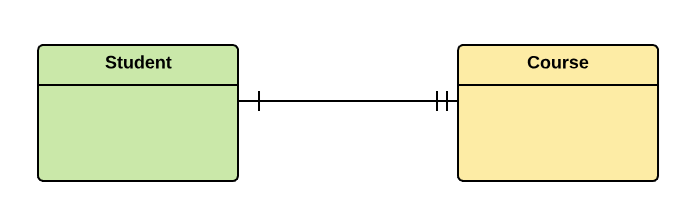
Different types of cardinal relationships are:

* One-to-One Relationships
* One-to-Many Relationships
* May to One Relationships
* Many-to-Many Relationships

**1.One-to-one:**

One entity from entity set X can be associated with at most one entity of entity set Y and vice versa.

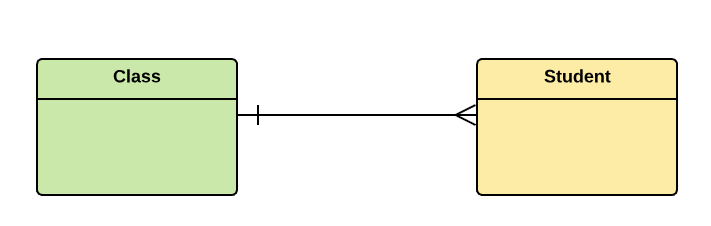
Example: One student can register for numerous courses. However, all those courses have a single line back to that one student.



**2.One-to-many:**

One entity from entity set X can be associated with multiple entities of entity set Y, but an entity from entity set Y can be associated with at least one entity.

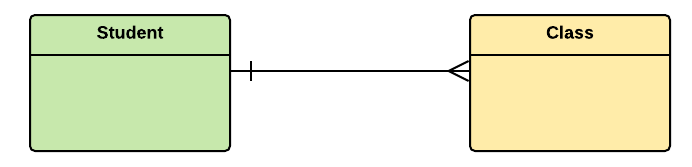
For example, one class is consisting of multiple students.



**3. Many to One**

More than one entity from entity set X can be associated with at most one entity of entity set Y. However, an entity from entity set Y may or may not be associated with more than one entity from entity set X.

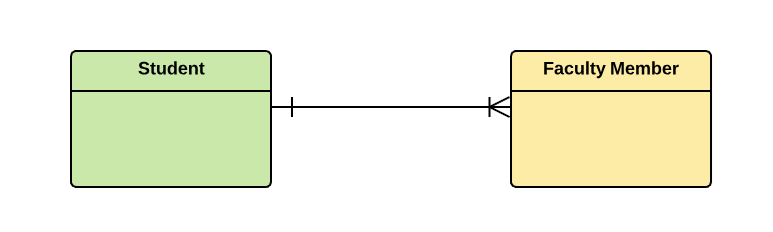
For example, many students belong to the same class.



**4. Many to Many:**

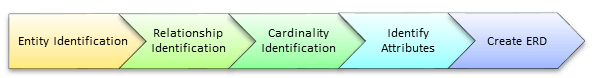
One entity from X can be associated with more than one entity from Y and vice versa.

For example, Students as a group are associated with multiple faculty members, and faculty members can be associated with multiple students.



**How to Create an Entity Relationship Diagram (ERD)**

Following are the steps to create an ER Diagram:



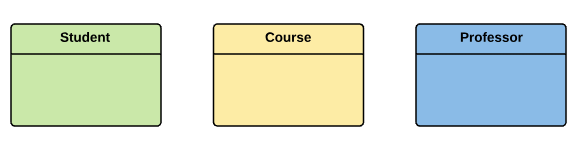
Let’s study them with an Entity Relationship Diagram Example:

**In a university, a Student enrolls in Courses. A student must be assigned to at least one or more Courses. Each course is taught by a single Professor. To maintain instruction quality, a Professor can deliver only one course**

**Step 1) Entity Identification**

We have three entities

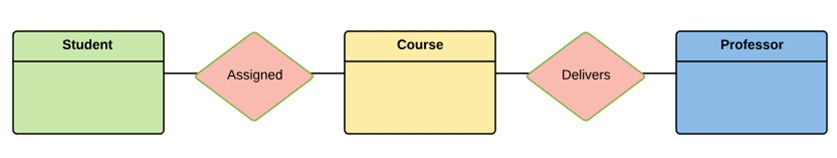
* Student
* Course
* Professor



**Step 2) Relationship Identification**

We have the following two relationships

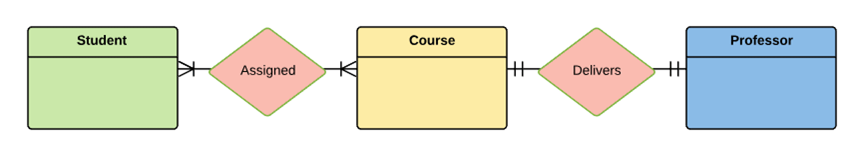
* The student is **assigned** a course
* Professor **delivers** a course



**Step 3) Cardinality Identification**

For them problem statement we know that,

* A student can be assigned **multiple** courses
* A Professor can deliver only **one** course



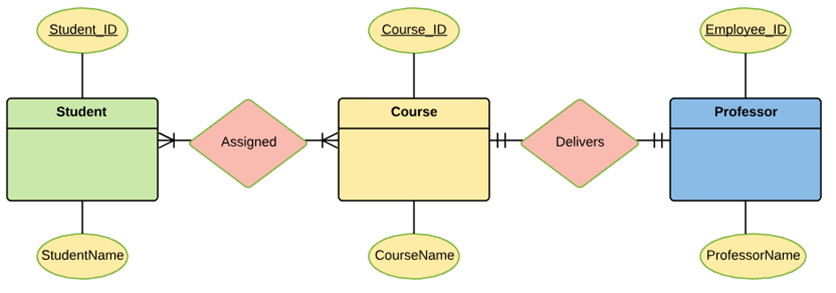
**Step 4) Identify Attributes**

You need to study the files, forms, reports, data currently maintained by the organization to identify attributes. You can also conduct interviews with various stakeholders to identify entities. Initially, it’s important to identify the attributes without mapping them to a particular entity.

Once, you have a list of Attributes, you need to map them to the identified entities. Ensure an attribute is to be paired with exactly one entity. If you think an attribute should belong to more than one entity, use a modifier to make it unique.

Once the mapping is done, identify the primary Keys. If a unique key is not readily available, create one.

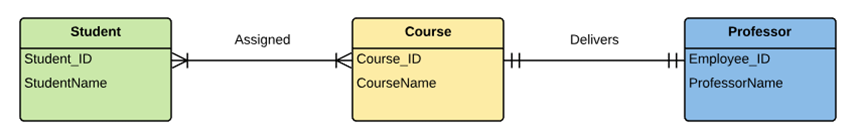
| **Entity** | **Primary Key** | **Attribute** |
| --- | --- | --- |
| Student | Student\_ID | StudentName |
| Professor | Employee\_ID | ProfessorName |
| Course | Course\_ID | CourseName |



For Course Entity, attributes could be Duration, Credits, Assignments, etc. For the sake of ease we have considered just one attribute.

### Step 5) Create the ERD Diagram

A more modern representation of Entity Relationship Diagram Example



**Entity-Relationship Diagram (ERD) for the scenario you've mentioned.**

Entities:

1. Student:
   * Attributes: StudentID (Primary Key), FirstName, LastName, DateOfBirth, etc.
2. Course:
   * Attributes: CourseID (Primary Key), CourseName, CourseCode, Credits, etc.
3. Professor:
   * Attributes: ProfessorID (Primary Key), FirstName, LastName, ContactInfo, etc.

Relationships:

1. Enrollment:
   * Represents the relationship between Students and Courses.
   * Cardinality: Many-to-Many (M:N), as each student can enroll in multiple courses, and each course can have multiple students.
   * Attributes: EnrollmentID (Primary Key), EnrollmentDate, Grade (or Score), etc.
   * Foreign Keys: StudentID (References Student), CourseID (References Course).
2. Teaching:
   * Represents the relationship between Professors and Courses.
   * Cardinality: One-to-Many (1:N), as each professor can teach multiple courses, but each course is taught by only one professor.
   * Attributes: TeachingID (Primary Key), Semester, Year, etc.
   * Foreign Keys: ProfessorID (References Professor), CourseID (References Course).

In this ERD:

* Students can enroll in multiple courses, and each enrollment is recorded in the Enrollment table.
* Professors can teach multiple courses, and each teaching assignment is recorded in the Teaching table.
* Each course is associated with one or more enrollments, and it is also taught by one professor.

**In this representation:**

* The three main entities are "Student," "Course," and "Professor."
* "Student" and "Course" are connected by the "Enrolls" relationship, representing the enrollment of students in courses.
* "Professor" and "Course" are connected by the "Teaching" relationship, representing the teaching assignment of professors to courses.
* "Enrollment" and "Teaching" tables store additional information about these relationships, such as enrollment dates, grades/scores, semesters, and years.

-- Create the Student table

CREATE TABLE Student (

StudentID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DateOfBirth DATE

);

-- Create the Course table

CREATE TABLE Course (

CourseID INT PRIMARY KEY,

CourseName VARCHAR(100),

CourseCode VARCHAR(20) UNIQUE,

Credits INT

);

-- Create the Professor table

CREATE TABLE Professor (

ProfessorID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

ContactInfo VARCHAR(100)

);

-- Create the Enrolls relationship table

CREATE TABLE Enrolls (

EnrollmentID INT PRIMARY KEY,

StudentID INT,

CourseID INT,

EnrollmentDate DATE,

GradeOrScore DECIMAL(5, 2),

CONSTRAINT FK\_Enrolls\_Student FOREIGN KEY (StudentID) REFERENCES Student(StudentID),

CONSTRAINT FK\_Enrolls\_Course FOREIGN KEY (CourseID) REFERENCES Course(CourseID)

);

-- Create the Teaching relationship table

CREATE TABLE Teaching (

TeachingID INT PRIMARY KEY,

ProfessorID INT,

CourseID INT,

Semester VARCHAR(20),

Year INT,

CONSTRAINT FK\_Teaching\_Professor FOREIGN KEY (ProfessorID) REFERENCES Professor(ProfessorID),

CONSTRAINT FK\_Teaching\_Course FOREIGN KEY (CourseID) REFERENCES Course(CourseID)

);

**In this MySQL code:**

* The Student, Course, and Professor tables represent the main entities.
* The Enrolls table represents the "Enrolls" relationship and includes columns for "EnrollmentDate" and "GradeOrScore" to store additional information about student enrollments.
* The Teaching table represents the "Teaching" relationship and includes columns for "Semester" and "Year" to store additional information about professor teaching assignments.

**Best Practices for Developing Effective ER Diagrams**

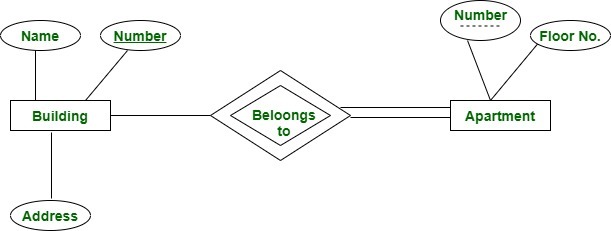
Here are some best practice or example for Developing Effective ER Diagrams.

* Eliminate any redundant entities or relationships
* You need to make sure that all your entities and relationships are properly labeled
* There may be various valid approaches to an ER diagram. You need to make sure that the ER diagram supports all the data you need to store
* You should assure that each entity only appears a single time in the ER diagram
* Name every relationship, entity, and attribute are represented on your diagram
* Never connect relationships to each other
* You should use colors to highlight important portions of the ER diagram
* ER Model in DBMS stands for an Entity-Relationship model
* The ER model is a high-level data model diagram
* ER diagrams are a visual tool which is helpful to represent the ER model
* ER diagrams in DBMS are blueprint of a database
* Entity relationship diagram DBMS displays the relationships of entity set stored in a database
* ER diagrams help you to define terms related to entity relationship modeling
* ER Model in DBMS is based on three basic concepts: Entities, Attributes & Relationships
* An entity can be place, person, object, event or a concept, which stores data in the database (DBMS)
* Relationship is nothing but an association among two or more entities
* A weak entity is a type of entity which doesn’t have its key attribute
* It is a single-valued property of either an entity-type or a relationship-type
* It helps you to defines the numerical attributes of the relationship between two entities or entity sets
* ER- Diagram DBMS is a visual representation of data that describe how data is related to each other
* While Drawing ER diagrams in DBMS, you need to make sure all your entities and relationships are properly labeled.

**Partial, Unique, Secondary, Composite and Surrogate keys in DBMS**

1. **Partial Key :**  
   The set of attributes that are used to uniquely identify a weak entity set is called the Partial key. Only a bunch of the tuples can be identified using the partial keys. The partial Key of the weak entity set is also known as a discriminator.

It is just a part of the key as only a subset of the attributes can be identified using it. It is partially unique and can be combined with other strong entity set to uniquely identify the tuples.



*Partial Key apartment number is shown with a dashed line.*

Here we have an apartment as a weak entity and building as a strong entity type connected via ‘belongs to’ relationship set. Apartment number is not globally unique i.e. more than one apartment may have same number globally but it is unique for a particular building since a building may not have same apartment number. Thus apartment number cannot be primary key of entity Apartment but it is a partial key shown with a dashed line.

1. **Unique key :**  
   It uniquely identifies a tuple in a relation. Unlike the Primary key, There can be more than one unique key in a table. It can accept only one null value. It cannot have duplicate values and it is non-updatable i.e. cannot be updated once it is assigned.

**Example –**  
The best example of unique key is Voter ID.  
Voter Id is unique for all the citizens. If it gets lost and another duplicate copy is issued, then the duplicate copy always has the same number as before. Thus, it is non-updatable. Few citizens may not have got their Voter Id, so for them, its value is NULL.

1. **Secondary Key :**  
   It is a candidate key that is not selected as primary key. For example, we have schema: Student(Id, email, enroll\_no, name) Candidate key of this relation is: ID, email, enroll\_no  
   If we select Id as primary key, then email and enroll\_no becomes secondary key of the relation.
2. **Composite key :**

Up to 16 columns can be combined into a single composite index key initially using sqlplus earlier versions. However, the limit is extended and we can combine any number of columns (practically of no use)

1. **Surrogate key :**  
   The surrogate key is internally generated.  
   Example : System date/time stamp, Counter