# Entity-Relationship Diagram

An Entity-Relationship Diagram (ERD) is a visual representation used in database design to depict the structure and relationships between different entities (or tables) within a database system. ERDs are a critical tool for database developers, designers, and stakeholders to understand and communicate how data is organized and connected in a database.

Entities: These represent the main objects or concepts in the database, such as customers, products, employees, or orders. Each entity is typically depicted as a rectangle in the diagram.

Attributes: Attributes describe the properties or characteristics of each entity. For example, a "Customer" entity may have attributes like "CustomerID," "FirstName," "LastName," and "Email.";

Relationships: Relationships illustrate how entities are connected to one another. They show how data is related and can be used to establish links between different entities. Common types of relationships include one-to-one, one-to-many, and many-to-many.

# Entities

Entities can be categorized into different types based on their roles and characteristics within the database model.

**Strong Entities**: These are entities that can exist independently and have their attributes. They represent primary objects in the database. For example, in a university database, the "Student" entity is a strong entity with attributes like student ID, name, and date of birth.

**Weak Entities**: Weak entities depend on a related strong entity for their existence. They do not have a primary key attribute of their own but instead have a partial key that connects them to a strong entity. A common example is the "Address" entity, which may rely on a "Person" entity to exist.

**Associative Entities**: Also known as junction entities or linking entities, associative entities are used to represent many-to-many relationships between two other entities. They often contain attributes that are specific to the relationship itself. For instance, in a library database, an "Authorship" entity could link "Author" and "Book" entities, indicating which author(s) wrote which book(s).

**Derived Entities**: These entities contain attributes that can be calculated or derived from other attributes within the database. They are often used to store information that can be computed when needed, rather than being explicitly stored. An example might be a "Total Price" entity in an e-commerce system, which can be derived from the quantity and unit price of items in a shopping cart.

Subtypes and Supertypes: Subtype entities represent specific categories or subsets of a supertype entity. This is commonly used in situations where entities share common attributes but also have attributes unique to their subtype. For instance, "Employee" could be a supertype with subtypes "Manager" and "Clerk," each having their own set of attributes.

**Singleton Entities:** These entities represent a **single**, unique instance within the database. They are often used for settings or configuration data that should have only one record. For example, a "Settings" entity might contain global settings for an application.

**Historical Entities**: These entities are used to store historical data, such as records of past transactions or changes in status. Historical entities often have attributes like "Effective Date" and "End Date" to track when data was valid.

**External Entities:** In some database models, external entities represent entities from external systems or sources that interact with the database. They are often used in data modeling to illustrate how the database interfaces with external systems or data.