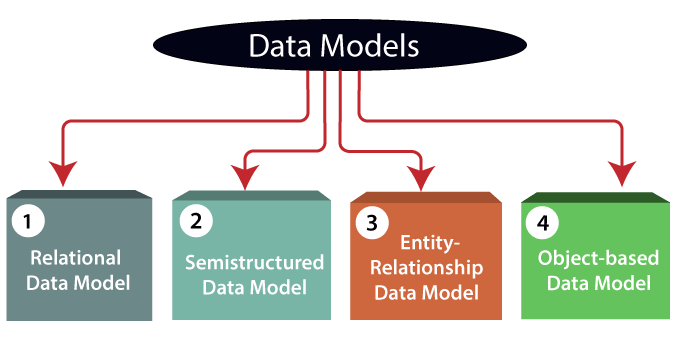
**Data Models**

* Data models define how the logical structure of a database is modelled.
* Data models define how data is connected to each other and how they are processed and stored inside the system.



**Why use Data Model?**

* Ensures that all data objects required by the database are accurately represented.
* Omission of data will lead to creation of faulty reports and produce incorrect results.
* A data model helps to design the database at the conceptual, physical and logical levels.
* Data Model structure helps to define the relational tables, primary and foreign keys and stored procedures.
* It provides a clear picture of the base data and can be used by database developers to create a physical database.
* It is also helpful to identify missing and redundant data.
* Though the initial creation of data model is laborious and time consuming, in the long run, it makes your IT infrastructure upgrade and maintenance cheaper and faster.

**Types of Data Model**

1)Relational Data Model:

* This type of model designs the data in the form of rows and columns within a table.
* Thus, a relational model uses tables for representing data and in-between relationships.
* Tables are also called relations.
* This model was initially described by Edgar F. Codd, in 1969.
* The relational data model is the widely used model which is primarily used by commercial data processing applications.

2) Entity-Relationship Data Model:

* An ER model is the logical representation of data as objects and relationships among them.
* These objects are known as entities, and relationship is an association among these entities.
* This model was designed by Peter Chen and published in 1976.
* It was widely used in database designing.
* A set of attributes describe the entities.
* For example, student\_name, student\_id describes the 'student' entity.
* A set of the same type of entities is known as an 'Entity set', and the set of the same type of relationships is known as 'relationship set'.

3) Object-based Data Model:

* An extension of the ER model with notions of functions, encapsulation, and object identity.
* This model supports a rich type system that includes structured and collection types.
* In 1980, various database systems following the object-oriented approach were developed.
* Here, the objects are nothing but the data carrying its properties.

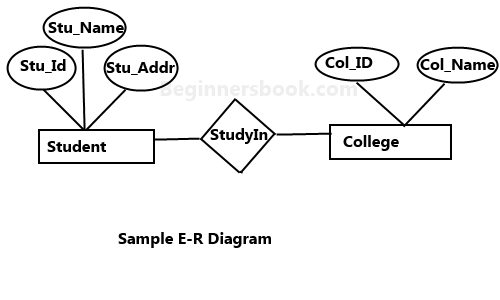
**4) Semi-structured Data Model:**

* This type of data model is different from the other three data models.
* The semi-structured data model allows the data specifications at places where the individual data items of the same type may have different attributes sets.
* The Extensible Markup Language, also known as XML, is widely used for representing the semi-structured data.
* Although XML was initially designed for including the markup information to the text document, it gains importance because of its application in the exchange of data.

**ER Modelling**

* An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram).
* An ER model is a design or blueprint of a database that can later be implemented as a database.
* It develops a conceptual design for the database.
* The main components of E-R model are: entity set and relationship set.
* An ER diagram shows the relationship among entity sets.
* An entity set is a group of similar entities and these entities can have attributes.
* In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

**Example**

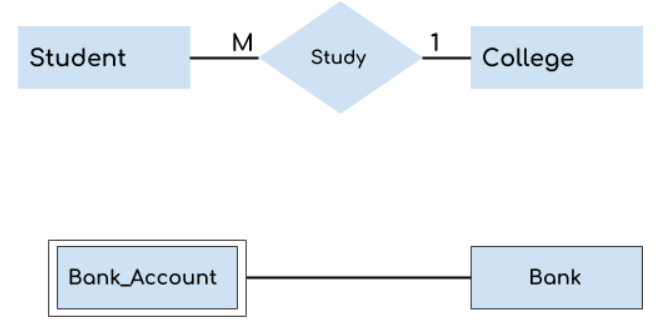
****

**Components of ER Diagram**

* Entity
  + Weak Entity
  + Strong Entity
* Attribute
  + Single valued
  + Multivalued attribute(mobile No.)
  + Composite attribute(name=fname+lname)
  + Derived attribute
* Relation
  + One to one
  + One to many
  + Many to one
  + Many to many

**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.
* Types of entities
  + Weak
  + Strong



**Weak and Strong Entity**

* Weak Entity-
  + 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.



* Strong entity- An entity that has a key attribute is called strong entity and represented by rectangle.

**Attribute**

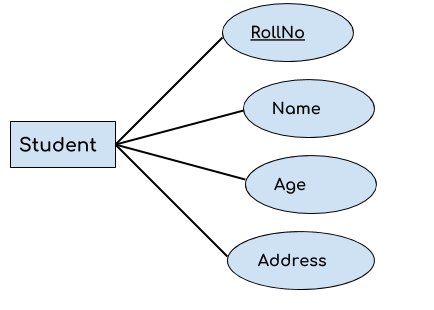
* The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.
* ****For example, id, age, contact number, name, etc. can be attributes of a student.

**Types of attributes**

1. Key attribute  
2. Simple and Composite attribute  
3. Single Valued and Multivalued attribute  
4. Derived attribute

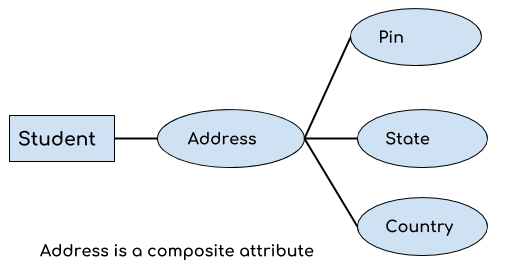
**Key Attribute**

* A key attribute can uniquely identify an entity from an entity set.
* For example, student roll number can uniquely identify a student from a set of students.
* Key attribute is represented by oval same as other attributes however the text of key attribute is underlined.

****

**Simple and Composite attribute:**

* If an attribute cannot be divided into simpler components, it is a **simple attribute**.
* Example for simple attribute : employee\_id of an employee.
* An attribute that is a combination of other attributes is known as **composite attribute**.
* For example, In student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country.

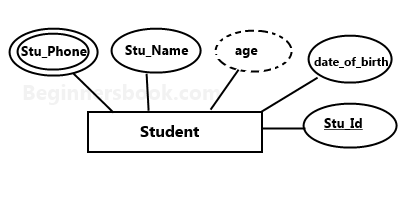


**Single and Multivalued attribute:**

* If an attribute can take only a single value for each entity instance, it is a **single valued attribute**. example for single valued attribute : age of a student. It can take only one value for a particular student.
* An attribute that can hold multiple values is known as **multivalued attribute**.
* It is represented with **double ovals** in an ER Diagram.
* For example – A person can have more than one phone numbers so the phone number attribute is multivalued.

**Derived Attribute :**

* A derived attribute is one whose value is dynamic and derived from another attribute.
* It is represented by dashed oval in an ER Diagram.
* For example – Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).

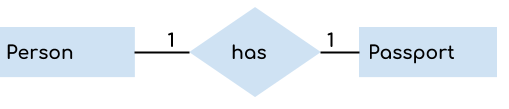


**Relationship**

* A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities.
* There are four types of relationships:  
  1. One to One  
  2. One to Many  
  3. Many to One  
  4. Many to Many

**One to one relationship**

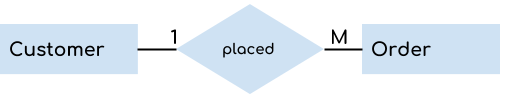
* When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship.



* For example, a person has only one passport and a passport is given to one person.

**One to Many Relationship**

* When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship.
* For example – a customer can place many orders but a order cannot be placed by many customers.



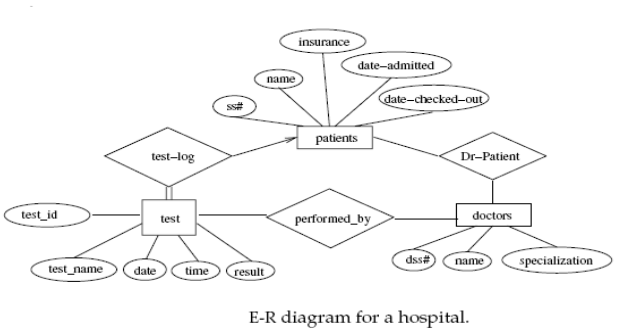
**Many to One Relationship**

* When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship.
* For example – many students can study in a single college but a student cannot study in many colleges at the same time. 

**Many to Many Relationship**

* When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationship.
* For example, a student can be assigned to many projects and a project can be assigned to many students.

**Example-E-R diagram for hospital**



**Advantages and Disadvantages of ER Modeling**

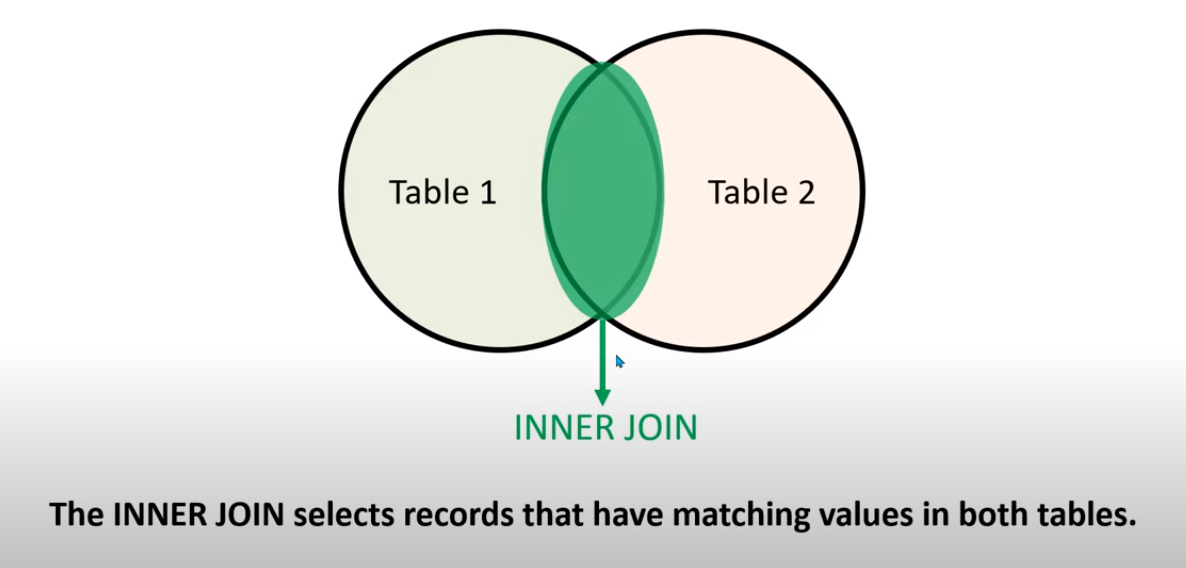
* Advantages
  + ER Modeling is simple and easily understandable.
  + It is represented in business users language and it can be understood by non-technical specialist.
  + Intuitive and helps in Physical Database creation.
  + Can be generalized and specialized based on needs.
  + Can help in database design.
  + Gives a higher level description of the system
* Disadvantages
  + Physical design derived from E-R Model may have some amount of ambiguities or inconsistency.
  + Sometime diagrams may lead to misinterpretations

SQL JOIN

The SQL Join help in retrieving data from two or more database tables. The tables are mutually related, using primary keys and foreign keys.

Type of Join

1. **INNER JOIN –** Returns rows when there is a match in both tables.



1.1 EQUI JOIN in SQL

Equi Join in SQL is a type of INNER Join that displays the output by performing a join operation between two or more tables based on the common column between them. It uses the equality ( = ) symbol to compare the data between two columns

1.2 NON-EQUI JOIN in SQL

Non-Equi Join is also a type of INNER Join in which we need to retrieve data from multiple tables. Non-Equi Join matches the column values from different tables based on an inequality based on the operators like <,>,<=,>=,!=, BETWEEN, etc.

1.3 Natural Join

Natural Join in SQL joins two tables based on the same attribute name and datatypes. The resulting table will contain all the attributes of both tables but keep only one copy of each common column.

|  |  |  |  |
| --- | --- | --- | --- |
| **s.no** | **Natural Join** | **s.no** | **Inner Join** |
| 1 | Natural Join joins two tables based on same attribute name and datatypes. | 1 | Inner Join joins two table on the basis of the column which is explicitly specified in the ON clause. |
| 2 | In Natural Join, The resulting table will contain all the attributes of both the tables but keep only one copy of each common column | 2 | In Inner Join, The resulting table will contain all the attribute of both the tables including duplicate columns also |
| 3 | In Natural Join, If there is no condition specifies then it returns the rows based on the common column | 3 | In Inner Join, only those records will return which exists in both the tables |
| 4 | Syntax-  SELECT \*  FROM table1 NATURAL JOIN table2; | 4 | Syntax-  SELECT \*  FROM table1 INNER JOIN table2 ON table1.Column\_Name= table2.Column\_Name; |

1. **Outer – Join** 
   1. LEFT OUTER JOIN – Returns all rows from the left table, and only matched data from the right table.
   2. RIGHT OUTER JOIN – Returns all rows from the right table, and only matched data from the left table
   3. FULL OUTER JOIN – Returns rows when there is a match in one of the table.
2. **SELF JOIN** - is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
3. **CROSS JOIN –** Produces a result set which is the number of rows in the first table multiplied by the number of rows in the second table. This kind of result is called as cartesian product.