1. Concept design with E-R Model

ER diagram or **Entity Relationship diagram** is a conceptual model that gives the graphical representation of data or it is a visual representation of data that describes how data is related to each other.

An ER diagram is mainly composed of following three components

1. Entity sets
2. Attributes
3. Relationship sets

Entity: An entity is a thing or an object in the real world that is distinguishable from other based on the value of the attribute is posses. (Any noun can be called as Entity)

Types of Entities:

Tangible / concrete: Entities which physically exist in real world.
Example: car, pen, book etc.

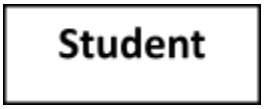
Intangible / abstract: Entities which exists logically.
Example: Account.

Entity set: Entity set is a group of similar entities that share the same properties i.e it represents schema / structure.

PERSON (name, age, address) ---------- Entity set
(Raju, 26, knr) ---------- Entity

- ❖ Entity cannot be represented in an ER diagram as it is instance / data.
- ❖ Entity set is represented by rectangle in ER diagram.

Symbol:



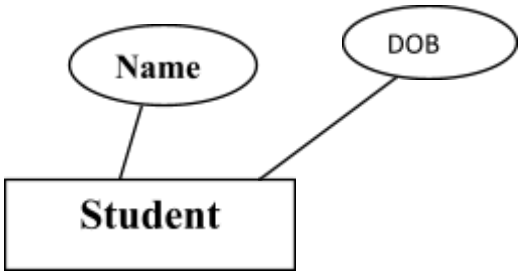
Attributes: Attributes are the units that describe the characteristics / properties of entities.

- ❖ For each attribute there is a set of permitted values called domains.
- ❖ In ER Diagram, represented by ellipse or oval.

Types of Attributes:

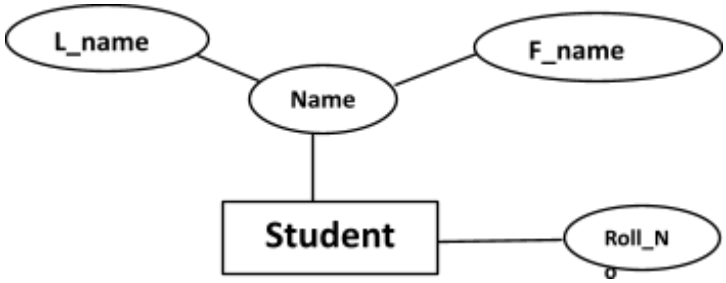
Simple Attributes: Simple attribute cannot be divided further, represented by simple oval.

Example:



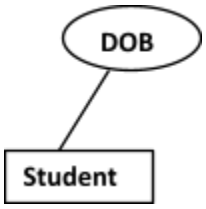
Composite Attribute: Composite attribute can be further divided in simple attribute, represented by oval connected to an oval.

Example:



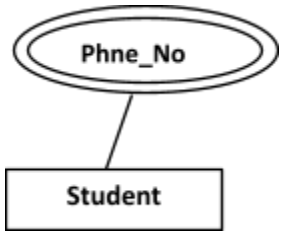
Single attribute: Single attribute can have only one value at an instance of time. Represented by oval

Example:



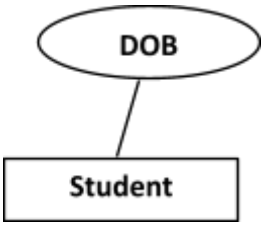
Multi-valued attribute: Multi-valued attribute can have more than one value at an instance of time.
Represented by double oval.

Example:



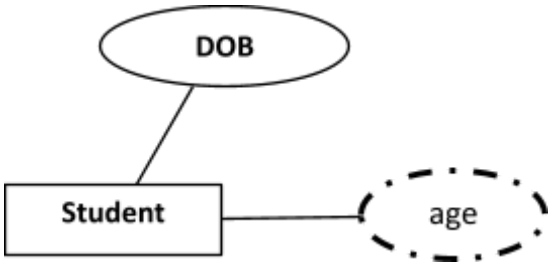
Stored attribute: Stored attribute is an attribute which are physically stored in the database.

Example:



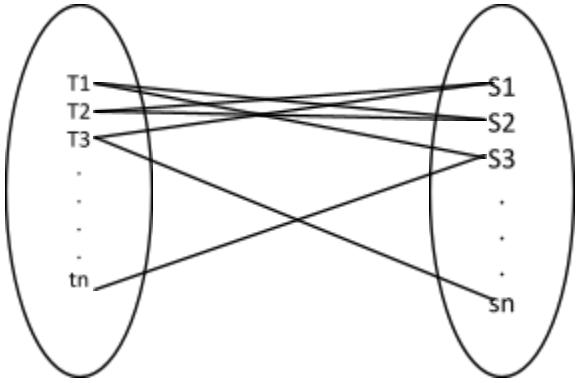
Derived Attribute: Derived attribute are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database.
Represented by dotted oval.

Example:



Complex Attribute: Complex Attribute is a type of attribute in database. It is formed by nesting composite attributes and multi-valued attributes.
Example: A Person can have more than one address (multivalued) and address can have city, pin code, country (composite).
Relationship: Relationship is an association between two or more entities of same or different entity set. (Any verb can be treated as Relationship)

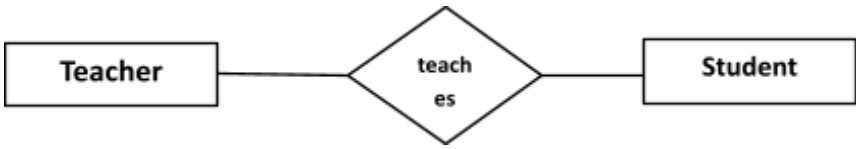
- ❖ No representation in ER diagram as it is an instance or data.



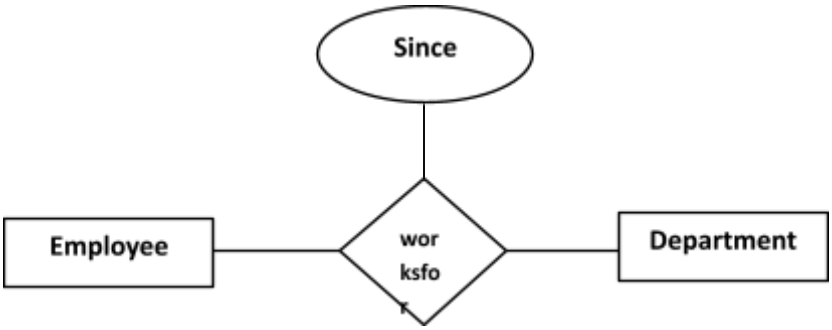
Relationship type / set: Relationship set is a set similar type of relationship.

- ❖ In ER Diagram it is represented using diamond symbol.

Example:

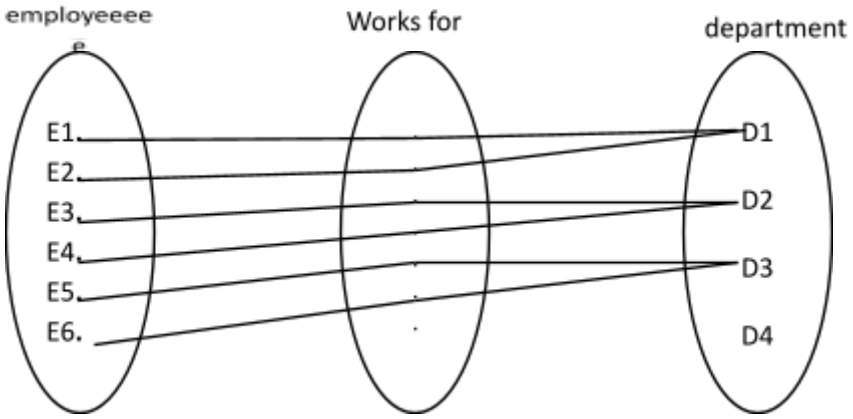


- **A Relationship may also have attributes called descriptive attributes. Example:**



Case Study:

Requirement Analysis: Every employee works for exactly a department and a department can have many employees. New department need not have any employee.



Degree: Number of entities that are participating in a relationship.
In the above example degree is 2.(binary relationship)

Cardinality ratio / Mapping cardinalities: Cardinality ratio is the max number of relationships in which an entity can participate.

In the above diagram e1 can participate only one relationship i.e 1.
Department is participating more than one relationship i.e N.

Participation or existence: Minimum number of relationships in which an entity can participate, sometimes it is also called as min – cardinality.

In the above figure e1 can participate min 1.
Department can participate 0.

Converting above set theory diagram into ER diagram.

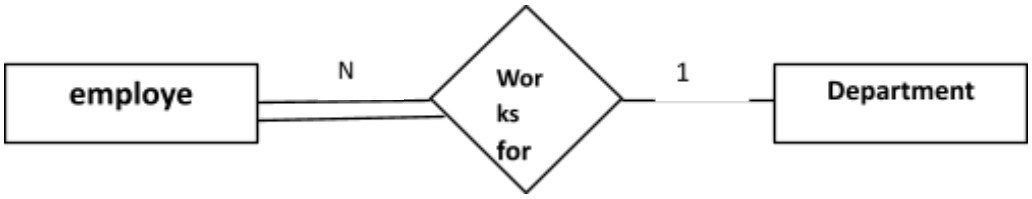
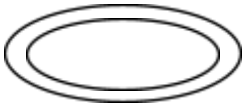


Figure - 1

Rectangles	Represents entity sets	
Ellipses	Represents attributes	
Diamonds	Represents relationships	
Lines	Attributes to entity sets & entity sets to relationships	
Double ellipses	Represents multivalued	
Dashed ellipses	Denotes derived attributes	
Doubles lines	Which indicates total participation of an entity in a relationship sets	
Double rectangles	Represents weak entity sets	

2. Relational Model

Relational Model was proposed by E.F Codd in 1970. At that time, most database systems were based on one of two older data models (hierarchical mode and network model).

Relational model were designed to model data in the form of relations or tables. After designing the conceptual model of database using ER diagram, we need to convert the conceptual model in the relational model which can be implemented using any RDBMS languages like MySQL, Oracle etc.

A database is a collection of one or more relations, where each relation is a table with rows and columns.

A relational model represents how data is stored in relational database. A relational database stores data in the form of relations (tables).

Advantages:

- ✔ The relational model is very simple and elegant.
- ✔ This simple tabular representation enables even novice users to understand the contents of a database and it permits the use of simple, high-level languages to query the data.
- ✔ The major advantages of the relational model over the older data models are its simple data representation and the ease with which even complex queries can be expressed.

Relation: The main construct for representing data in the relational model is a relation. A Relation consists of a *relation schema* and *relation instance*.

Relation schema: Relation schema specifies / represents the name of the relation, name of each field (or column or attribute), and the domain of each field.

The set of permitted values for each attribute is called domain" or "A domain is referred to in a relation schema by the attribute name and has a set of associated values"

Example:

Students (*sid:* string, *name:* string, *login:* string, *age:* integer, *gpa:* real)

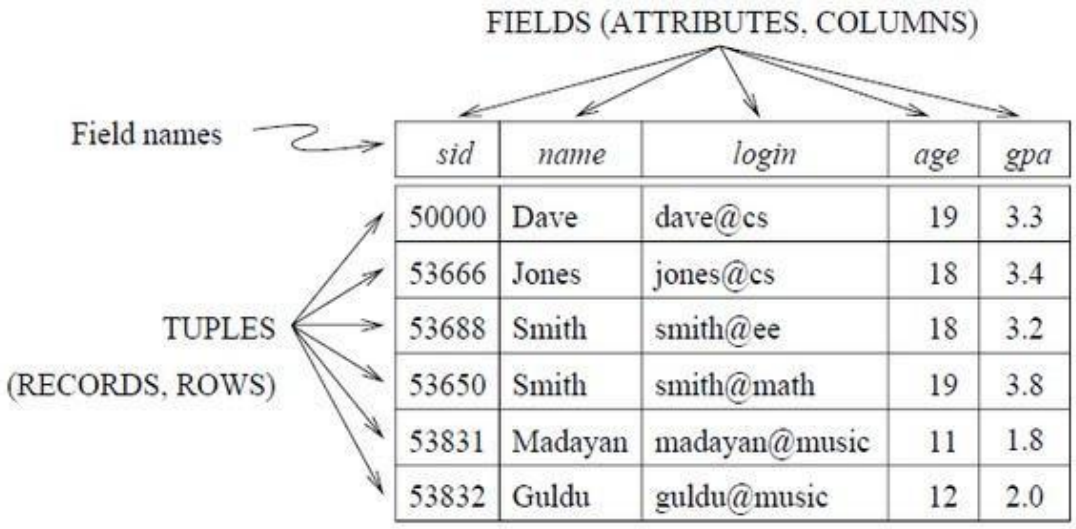
Here field name *sid* has a domain named *string*. The set of values associated with domain string is the set of all character strings.

Relation instance:

An instance of a relation is a set of tuples at a particular instance of time, also called records, in which each tuple has the same number of fields as the relation schema. A relation instance can be thought of as a table in which each tuple is a row, and all rows have the same number of fields. (The term relation instance is often abbreviated to just relation).

It can change whenever there is insertion, deletion, updation in the database.

Example:



sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Figure 3.1 An Instance *S1* of the Students Relation

Degree: Number of attributes in the relation in known as degree of relation. Student relation defined above as degree 5.

Degree is also known as arity.

Cardinality: Cardinality of a relation instance is the number of tuples.

In the above student relation, the cardinality is six.

Integrity constraints over Relations:

An integrity constraint (IC) is a condition on a database schema and restricts the data that can be stored in an instance of the database.

Integrity constraints are a set of rules. It is used to maintain the quality of information.

Integrity constraints ensure that changes (update deletion, insertion) made to the database by authorized users do not result in a loss of data consistency. Thus, integrity constraints guard against accidental damage to the database.

Various types of integrity constraints are –

1. Domain constraints
2. General constraints
3. Key constraints
4. Foreign key constraints.

Domain constraints: Domain integrity means the definition of a valid set of values for an attribute. The values that appear in a column must be drawn from the domain associated with that column. The domain of a field is essentially the type of that filed.
Example:

Roll Number	Name	age
101	Raju	50
102	Ravi	30
104	Ramu	A

Not allowed because age is integer

General Constraints:

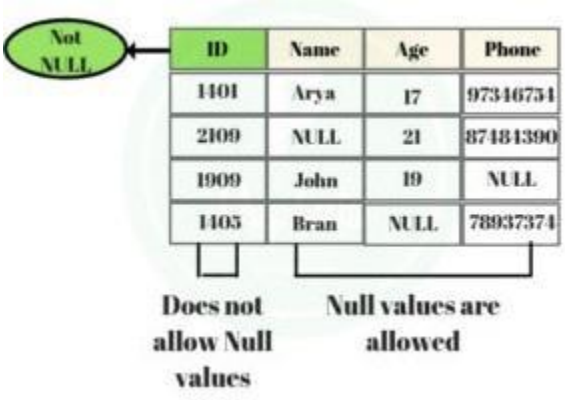
Constraints are the rules that are to be followed while entering data into columns of the database table.
Constraints ensure that data entered by the user into columns must be within the criteria specified by the condition
For example, if you want to maintain only unique IDs in the employee table or if you want to enter only age less than 18 in the student table etc.

Some of the constraints are:

- ❖ NOT NULL
- ❖ DEFAULT
- ❖ CHECK

1. NOT NULL: Once not null is applied to a particular column, you cannot enter null values to that column and restricted to maintain only some proper value other than null.

Example:



Example:

- ❖ CREATE TABLE test (ID int NOT NULL, name char(50),age int, phone varchar (50));
- ❖ DESC test;

- INSERT INTO test values(100,"Deepak",30,"9966554744");
- SELECT * FROM test;
- INSERT INTO test VALUES (NULL,"Raju",40,"9966554744");**//Error**

2. DEFAULT:

- Default clause in SQL is used to add default data to the columns.
- When a column is specified as default with some value then all the rows will use the same value i.e each and every time while entering the data we need not enter that value.
- But ***default column value can be customized*** i.e it can be overridden when inserting a data for that row based on the requirement.

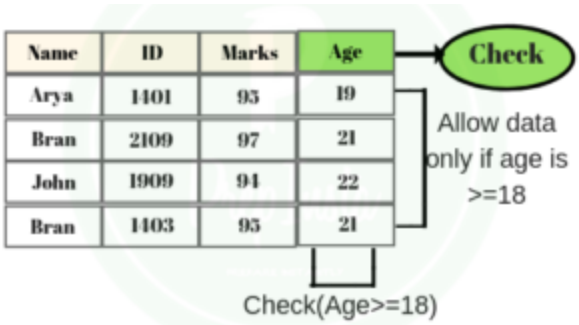
Example:

- DROP TABLE test;
- CREATE TABLE test(rollno int,name varchar(50) DEFAULT "KITS");
- INSERT INTO test values(102,"Rajitha");
- SELECT * FROM test;
- INSERT INTO test(rollno) values(103);
- SELECT * FROM test;

3. CHECK:

- Suppose in real-time if you want to give access to an application only if the age entered by the user is greater than 18 this is done at the back-end by using a check constraint.
- Check constraint ensures that the data entered by the user for that column is within the range of values or possible values specified.

Example:



Example:

Drop table test;

CREATE TABLE test(sno int,age int,CHECK (age>=18));

DESC test;

INSERT INTO test values(101,52);

SELECT * FROM test;

INSERT INTO test values(102,10);

SELECT * FROM test;