DBMS:

Commonly Used Words	Meanings
Attributes	Columns
Tuples	Rows
Relation	Table

1) **Data Storage:**
    1) Data-Processing(retrieval of data, processing the data) is an integral part of the software industry.
2) **Data & Information**

Data	Information
Known facts, figures, objects and events which can be stored	Data that is processed to be useful, i.e. The data we can understand and consume.
Types: 1) Structured: numbers, text, date etc. 2) Un-structured: images, videos, documents etc.	Eg: Roll Number is 12

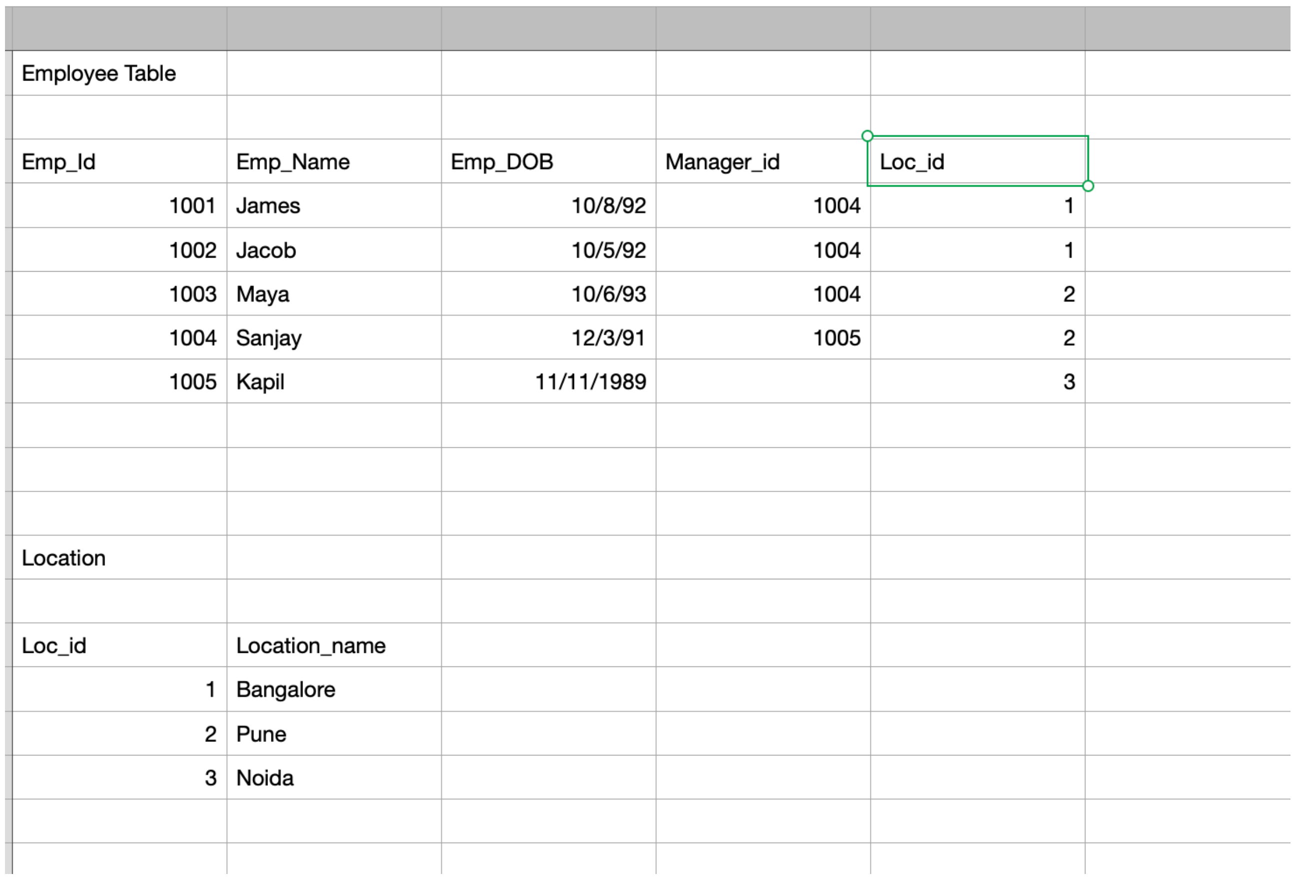
3) **Traditional Approach:**
    1) Storing data in file-based systems
    2) File Based System:
        1) Properties:
            1) Data was stored as collection of records in Flat-files(Data-Files)
            2) We had to generate reports for the end-user to access the data in the files
            3) Each application defined and managed its own data
        2) Limitations:
            1) Separation and isolation of data
            2) Duplication of data
            3) No concurrent access to data
            4) No simultaneous application access to data
            5) No data independence
4) **Database Approach:**
    1) Data is stored in DB as a collection of data-files
    2) DB:
        1) A collection of related data
    3) DBMS(Database Management System):
        1) A s/w system to facilitate the creation and maintenance of computerised database.
    4) Database System:
        1) DBMS+DB

5) Advantages;
    1) Control over Data redundancy
    2) Data Consistency
    3) Program-Data independence
    4) More secure
    5) Concurrent access
    6) Flexible for Application Development

5) **Data Models:**
    1) What?
        1) It is integrated collection of concepts(Tool) for describing data, relationship between data etc.
    2) Why?
        1) To represent data in understandable way
    3) Types:
        1) Object-Based
            1) Entity-Relationship
            2) Semantic
            3) Functional
            4) Object-Oriented
        2) Record-Based
            1) Relational Model:
                1) Definition:
                    1) Proposed in 1970-> E.F Codd
                    2) Several commercial products: DB2, Oracle etc.
                    3) Several free open-source implementations: MySQL, PostgreSQL
                2) Terminologies:
                    1) Table:
                        1) Collection of rows and columns to logically view the data
                        2) Also called a relation
                    2) Attributes: named column in a relation
                    3) Domain: set of allowed values for one or more attributes
                    4) Tuple: a row in a relation
                    5) Degree: Number of attributes(columns) in a relation
                    6) Cardinality: Number of Tuples(rows) in a relation
                    7) It is a collection of normalised relations and distinct relation names
            2) Network Data Model(similar to match-the-following)
            3) Hierarchical Data Model(Tree-Format)
        3) Physical

6) DataBase Keys:

1) One or more attributes(columns), used to identify a record in a relation/table
2) Types:
    1) Candidate Key:
        1) One or more attributes, used to uniquely identify a record in a relation/table
    2) Super Key:
        1) Superset of Candidate Key i.e.
            1) Candidate Key + Non-key attribute
    3) Foreign Key(should always refer to value of candidate key):
        1) Used to related one or more tables
        2) One or more attributes which can be used to refer value of a candidate key: in same relation or a different relation



Employee Table

Emp_Id	Emp_Name	Emp_DOB	Manager_id	Loc_id
1001	James	10/8/92	1004	1
1002	Jacob	10/5/92	1004	1
1003	Maya	10/6/93	1004	2
1004	Sanjay	12/3/91	1005	2
1005	Kapil	11/11/1989		3

Location

Loc_id	Location_name
1	Bangalore
2	Pune
3	Noida

    4) Alternate Key:
        1) Candidate key that is not chosen to be the PK
    5) Primary Key:
        1) Used in table creation

    2) Cannot be null
    3) If a relation has several Candidate Keys, one can be chosen randomly as the PK
  6) Composite Key:
    1) Set of one or more keys, that together can uniquely identify a record.

Entity-Relationship(ER) Data Model:

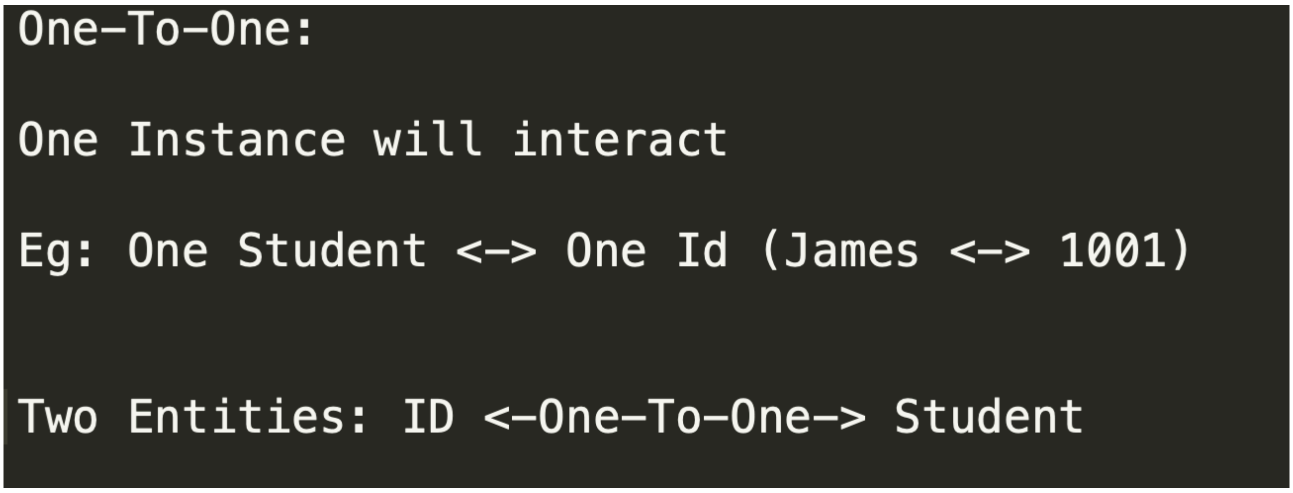
Terminologies:

ER Terminologies	Simplified
Entity	Table
Attributes	Columns
Regular Entity	String Entity
Identifier	Key-Attribute
Identifying Relationship	Relationship Connecting Strong & weak Entity
Degree of Relationship	Number of entities(tables) that can participate in a relationship
Cardinality of Relationship	Number of instances of entities participating

1. Basic Definition:
    1. Helps to capture conceptual DB design
    2. Adopts Top-Down Approach
    3. Describes functional data-requirements in real-world problems in the form of ER
    4. Consists of Entities, Relationships, Identifiers, attributes etc.
    5. Apart from ERD, we can use UML class diagrams
2. Terminologies;
    1. Entity(Table):
        1. Definition:

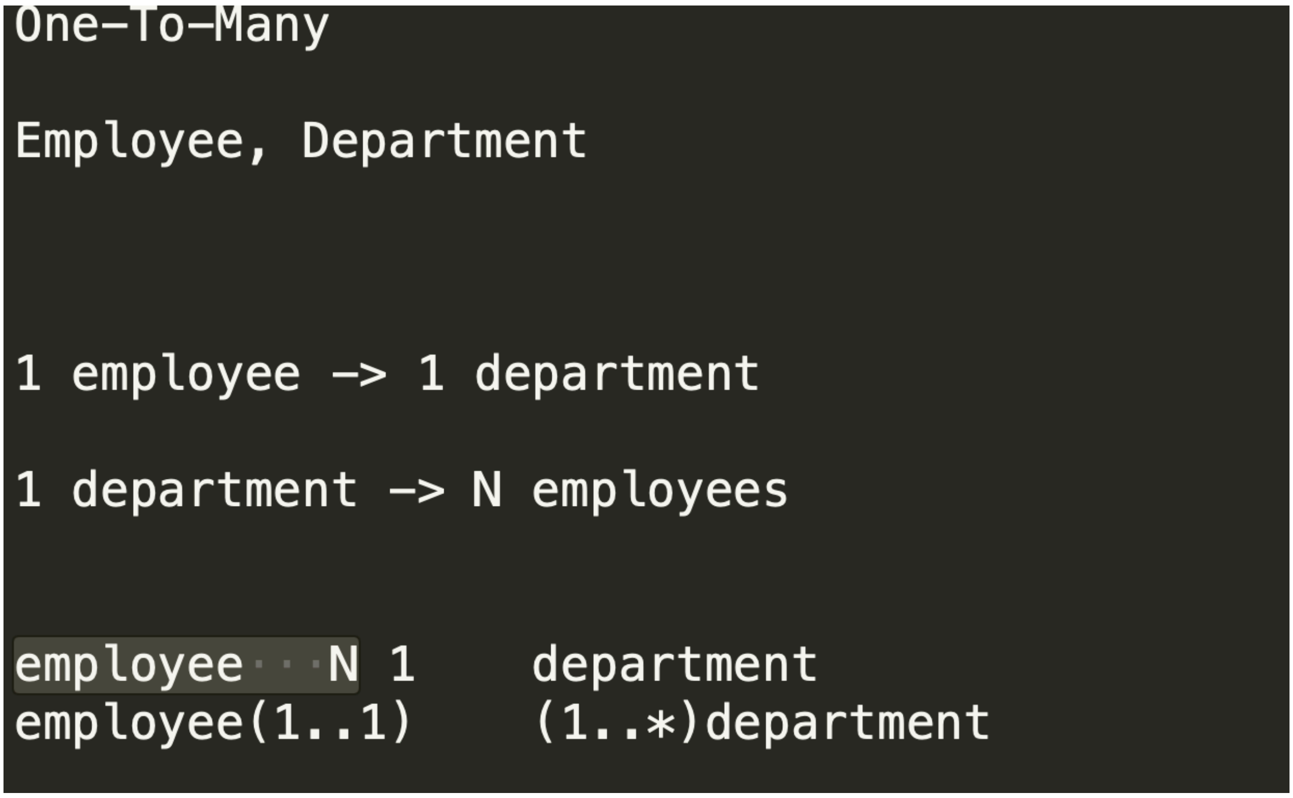
      1. Specific Objects or things that are represented in the database
   2. Types:
      1. Strong/Regular Entity:
         1. Can exist independently
         2. Have its own unique key
      2. Weak Entity:
         1. Cannot exist on their own and are dependent on another strong entity for their existence
         2. Does not have a unique key(only have Partial Key)
2. Attributes(column-names):
   1. Definition:
      1. Properties to describe the entity
      2. Each attribute has a set of values associated with it
   2. Types:
      1. Simple:
         1. Cannot be sub-divided into further sub-components
         2. Eg: Age, gender etc.
      2. Composite:
         1. May be composed of several components.
         2. Eg: Address, Name -> FirstName & LastName
      3. Single-Valued:
         1. Only atomic values
         2. Eg: DOB -> you can have only single DOB
      4. Multi-Valued:
         1. They can have multiple-values
         2. Eg: Degree -> one person can have multiple degrees, Hobbies -> one person can have multiple hobbies
      5. Key-Attribute:
         1. Used to uniquely identify a record
         2. Eg: PK
3. Relationships:
   1. Definition:
      1. A relationship relates two or more distinct entities with a specific meaning.
      2. Eg: Employee -> **<u>WORKS ON</u>** -> Project
      3. The Relationship(WORKS ON) can have their own set of attributes, to describe it
   2. Degrees:
      1. Number of entities that can participate in a relationship
      2. Type:
         1. Unary(Degree-1):

        1. Only one participant
        2. One entity related to another entity of same type
        3. Eg: Employee manages another employee
    2. Binary(Degree-2):
        1. Teo different participants
        2. Two different entities related/linked to each other
        3. Eg: Employee & Location
    3. Ternary(Degree-3)
        1. Three different participants
        2. Three different entities that are linked to each other
        3. Eg: Parental relation between mother, father and child
3. Cardinality:
    1. Defintion:
        1. The number of instances of an entity that can participate in a relationship
    2. Types:
        1. One-To-One(1:1):
            1. Each entity in a relationship will have exactly one related entity
            2. Eg: Id <-> Student



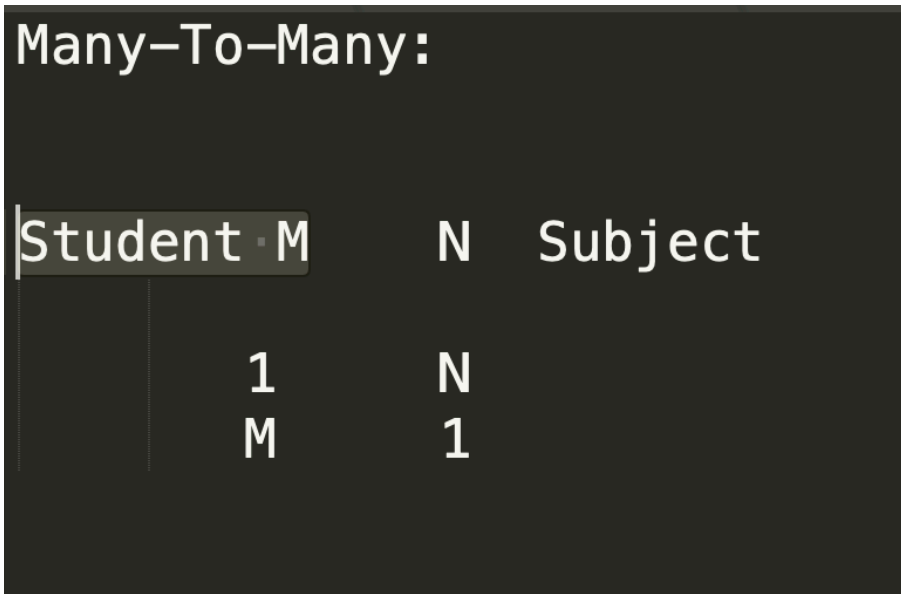
```
One-To-One:

One Instance will interact

Eg: One Student <-> One Id (James <-> 1001)


Two Entities: ID <-One-To-One-> Student
```

        2. One-To-Many(1:N) or Many-To-One(N:1):
            1. An entity on one side can have many related entities, but an entity on other side can have maximum one related entity
            2. Eg: Employee <-> Department



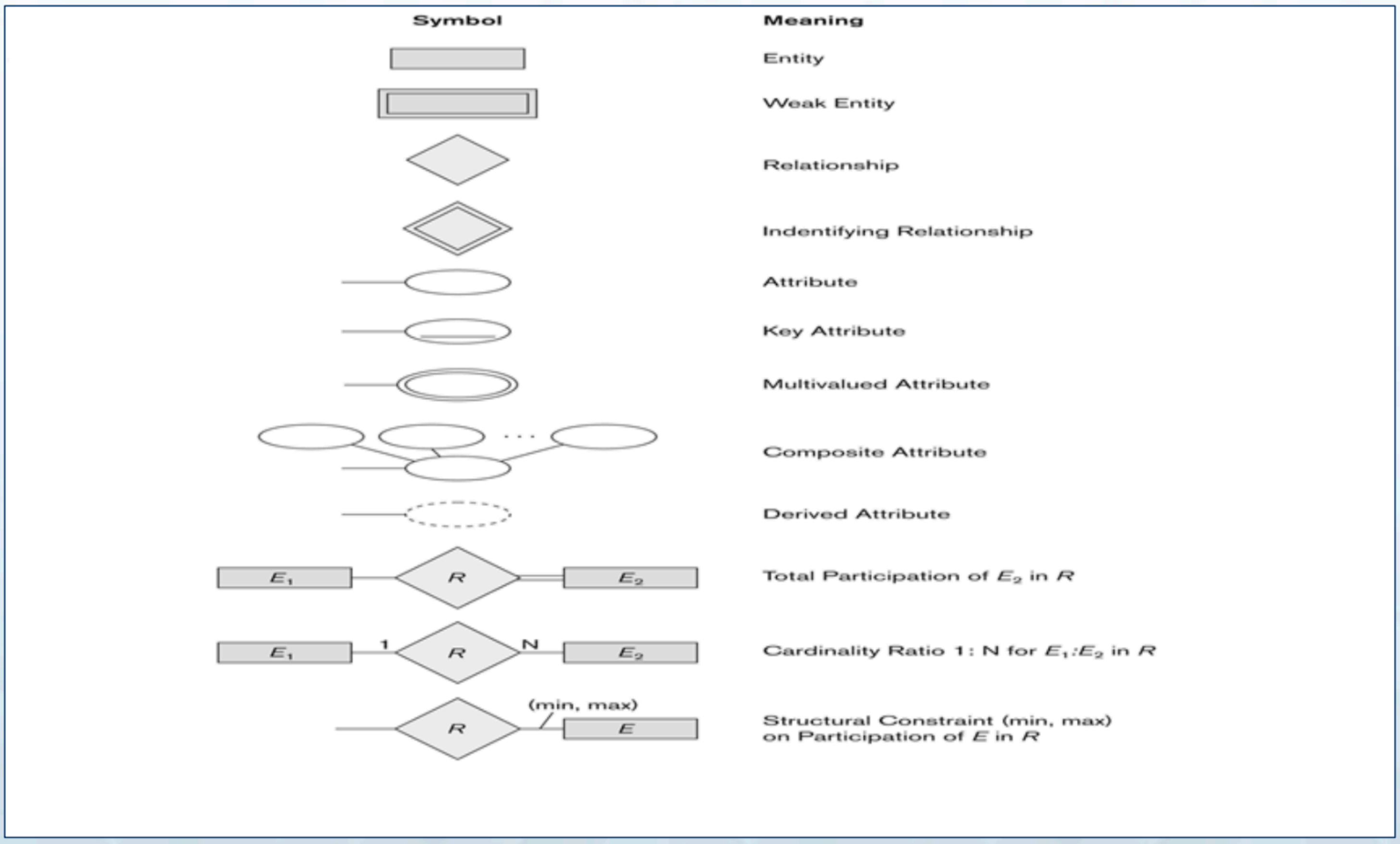
```
One-To-Many

Employee, Department



1 employee -> 1 department

1 department -> N employees


employee   N 1    department
employee(1..1)    (1..*)department
```

3. Many-To-Many(M:N):
    1. Entities on both sides can have many related entities on the other side and vice-versa
    2. Eg: Student <-> Subject



```
Many-To-Many:


Student M     N  Subject

          1   N
          M   1
```

4. Notations:

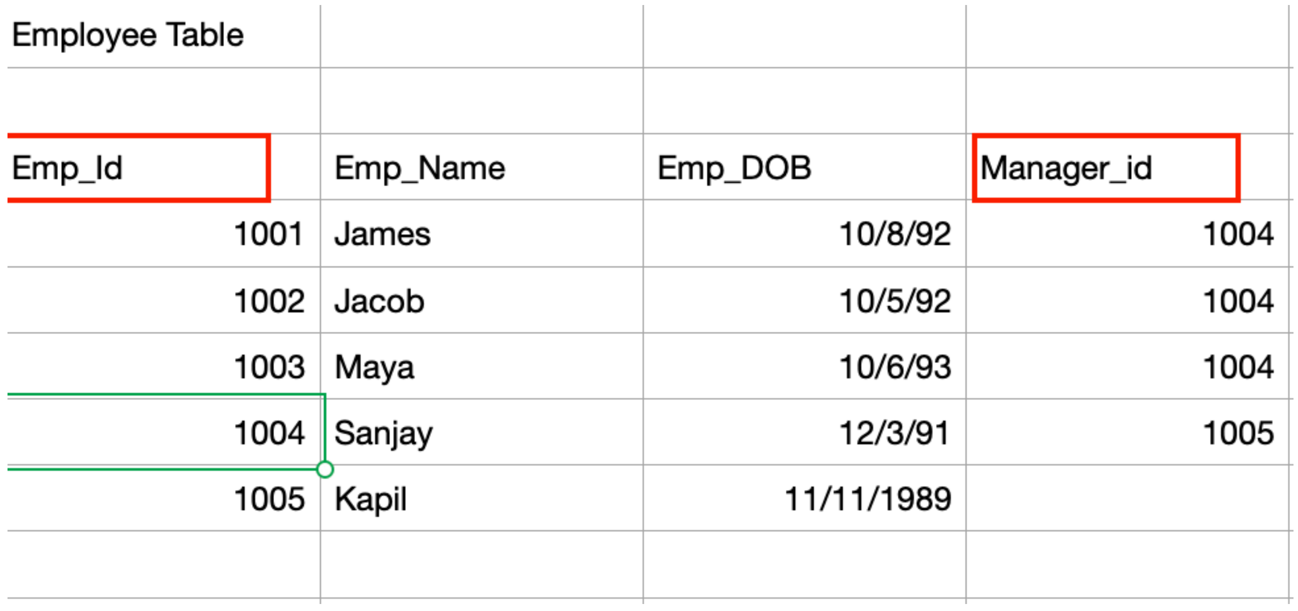


Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
. . .	Composite Attribute
	Derived Attribute
E_1 R E_2	Total Participation of E_2 in R
E_1 1 R N E_2	Cardinality Ratio 1: N for E_1:E_2 in R
R (min, max) E	Structural Constraint (min, max) on Participation of E in R

ERD To Relational Data Model:

1. **Mapping of Regular/Strong Entities:**
    1. Create a table for each strong entity
    2. Create columns for:
        1. Simple
        2. Composite
        3. Single-valued
    3. Create PK (any one of the key attributes)
    4. Ignore Derived attributes (eg: Age that can be calculated using DOB)
2. **Mapping of Weak Entities:**
    1. Create a table for each weak entity
    2. Create columns for
        1. Simple
        2. Composite Attributes
    3. Create a FK column by including PK of the strong entity
    4. Create PK for weak entity -> Partial Key(Weak Entity) + FK
3. **Mapping of multi-valued attributes:**
    1. Create a table for multi-valued attributes
    2. Create PK -> PK of the entity + multi-valued attribute
    3. Eg: for a person with multi valued attribute as hobbies we will create a new table called hobbies and in that the PK for hobbies table will be (person_id+hobby_name)
4. **Mapping of Relationship Types:**
    1. **Binary:**
        1. **One-To-One(1:1)**
            1. Include one attribute as Foreign Key(FK) on either side
        2. **One-To-Many(1:M)**
            1. Includes FK on the ‘M’ side
        3. **Many-To-Many(N:M)**
            1. Create a new table altogether for this relationship
            2. Create PK attribute by combining the PK attributes of both the participating entities
    2. **Unary:**

1. **One-To-Many(1:M):**
    1. Include an attribute as a recursive FK in the same relation
    2. Eg: Manager and Employee



Employee Table			
Emp_Id	Emp_Name	Emp_DOB	Manager_id
1001	James	10/8/92	1004
1002	Jacob	10/5/92	1004
1003	Maya	10/6/93	1004
1004	Sanjay	12/3/91	1005
1005	Kapil	11/11/1989	

2. **Many-To-Many(N:M):**
    1. Create a new table
    2. Create PK attribute by combining the PK of the participating entity and the relation

3. **Ternary:**
    1. Create a new table for relationship-type
    2. Create a PK by combining all the participating entities PK