UNIT – II

Logical Database Design: Relational DBMS, Codd's Rule, Entity-Relationship model, Extended ER, Normalization, Functional Dependencies, Anomaly, 1NF to 5NF, Domain Key Normal Form, Denormalization.

COURSE OBJECTIVES:

- To know the fundamentals of E-R model and concepts of Normalization and Functional Dependencies

COURSE OUTCOMES:

✓ Understand the fundamentals of E-R model and concepts of Normalization and Functional Dependencies

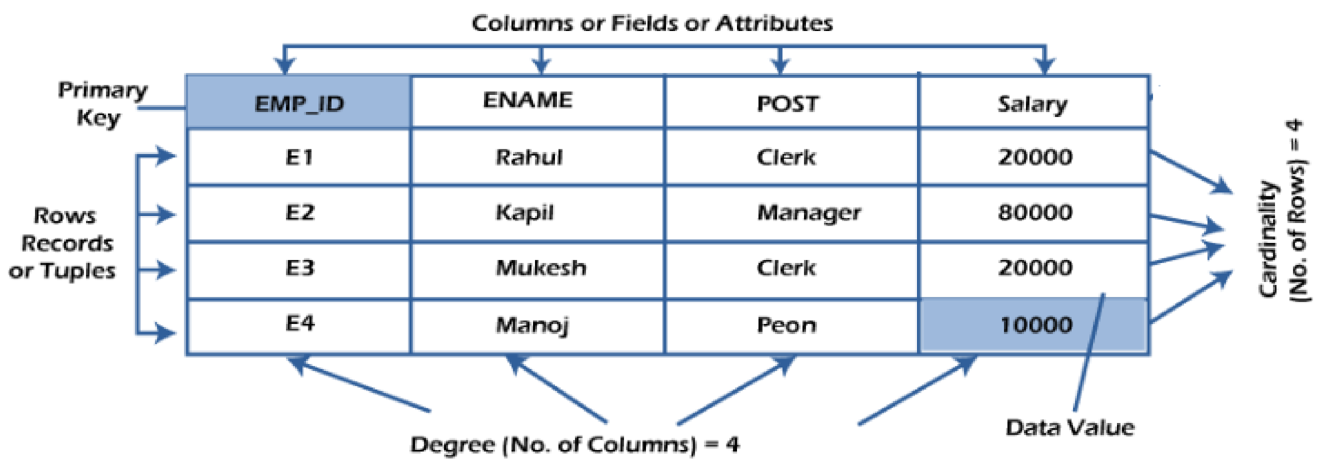
RELATIONAL DBMS

RDBMS stands for *Relational Database Management System.* Modern database management systems like SQL, ORACLE, MySQL are based on RDBMS. It is called Relational Database Management System (RDBMS) because it is based on the relational model introduced by E.F. Codd.

In RDBMS data is stored in the form of tables/relations. A table is a collection of related data entries and contains rows and columns to store data. Each table represents some real-world objects such as person, place, or event about which information is collected.

✓ Data in RDBMS is represented in terms of records/tuples (rows) and attributes (columns). The organized collection of data into a relational table is known as the logical view of the database.

A simple relation in RDBMS is represented as:


Columns or Fields or Attributes
Primary Key
Rows Records or Tuples
Cardinality (No. of Rows) = 4
Degree (No. of Columns) = 4
Data Value

EMP_ID	ENAME	POST	Salary
E1	Rahul	Clerk	20000
E2	Kapil	Manager	80000
E3	Mukesh	Clerk	20000
E4	Manoj	Peon	10000

Properties of a Relation:

- Each relation has a unique name by which it is identified in the database.
- Each attribute contains a distinct name
- Relation does not contain duplicate tuples.
- The tuples of a relation have no specific order.

Domain/Data Value: A set of permitted values of each attribute of a table is called as domain.

Relational schema: A relational schema contains the name of the relation and name of all columns or attributes.

Degree of a relation: The total number of attributes that comprise a relation is known as the degree of the table.

Cardinality of a relation: The total number of records/tuples at any one time in a relation is known as the table's cardinality. The relation whose cardinality is 0 is called an empty table.

ATTRIBUTES IN RELATIONAL DATABASE:

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

- **Simple Attribute:** Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
- **Composite Attribute:** Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.
- **Single valued Attribute:** Single-valued attributes contain single value. For example, Social_Security_Number.
- **Multi-valued Attribute:** Multi-valued attributes may contain more than one value. For example, a person can have more than one phone number, email_address, etc.
- **Derived Attribute:** Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.

These attribute types can come together in a way like −

- simple single-valued attributes
- simple multi-valued attributes
- composite single-valued attributes
- composite multi-valued attributes

KEYS IN RELATIONAL DATABASE:

Key: Key is defined as an attribute that contains unique values and is used to identify records in a relation.

Types of Keys in Relational Database

1) **PRIMARY KEY:** The primary key is defined as an attribute that uniquely identifies records in a relation. Primary key does not accept NULL values.

SNo	SName	Age	Second-Language	Division
95	Swetha	19	Hindi	First
96	Dhatri	21	Sanskrit	Second
97	Shivani	20	Telugu	First
98	Kavya	18	Hindi	Second
99	Jagruti	22	Sanskrit	First
100	Shravani	20	Telugu	First

Student Relation

In the above relation **student**, we can choose roll number attribute as primary key because for each row, there is a unique value of Roll number attribute i.e., same roll number is not repeated in another rows. Therefore *Roll Number* is primary key for given relation.

2) **CANDIDATE KEY:** A candidate key is an attribute or set of attributes that can uniquely identify a tuple. Except for the primary key, the remaining keys are considered a candidate key. The candidate keys are as strong as the primary key.
For example: In the EMPLOYEE table, id is best suited for the primary key. The rest of the keys, like Aadhar_Number, Passport_Number, License_Number, etc., are considered as candidate keys.

3) **SECONDARY/ALTERNATE KEY:** A candidate key that is not the primary key, called as alternate key.
For example: In the EMPLOYEE table, id is the primary key, and keys like Aadhar_Number, License_Number can be considered as secondary/alternate keys.

4) **UNIQUE KEY:** The unique key is defined as an attribute that uniquely identifies records in a relation. It is same as Primary key except that Unique key accepts NULL values.

5) **SUPER KEY:** More than one attribute combined together for unique identification of the record is defined as a Super Key As shown in figure below, neither supplier no. (S#), nor product no. (P#) are enough to identify the each row. To get unique information for each row, we need combined attributes s#, p#. i.e. {s# + p#} is a Super key (or) concatenate key.

S#	**P#**	**Qty**
S1	P1	500
S1	P2	700
S2	P3	450
S3	P1	700
S3	P2	500

6) **FOREIGN KEY:** When Primary key of one relation acts as key for another relation, it is known as Foreign key for the second relation.
   - For example, EMPLOYEE and DEPARTMENT are two different relations. Every employee works in a specific department in a company. As we can't store the department's information in the employee table, we add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.
   - In the EMPLOYEE table, Department_Id is the Foreign key, and both the tables are related.

7) **SURROGATE KEY:** The artificial key created for using in data analysis is known as a Surrogate Key. These keys are created when a primary key is large and complex.

CODD'S RULES:

Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.

Rule 1: Information Rule: The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule: Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value).

Rule 3: Systematic Treatment of NULL Values: The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

Rule 4: Active Online Catalog: The structure description of the entire database must be stored in an online catalog, known as data dictionary, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule: A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application.

Rule 6: View Updating Rule: All the views of a database, which can theoretically be updated, must also be updatable by the system.
Rule 7: High-Level Insert, Update, and Delete Rule: A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.
Rule 8: Physical Data Independence: The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.
Rule 9: Logical Data Independence: The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.
Rule 10: Integrity Independence: A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.
Rule 11: Distribution Independence: The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.
Rule 12: Non-Subversion Rule: If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

ENTITY-RELATIONSHIP MODEL

The entity-relationship (E-R) data model uses a collection of basic objects, called ***entities***, and ***relationships*** among these objects. It develops a conceptual design for the database.

ENTITY: An entity is a real time "thing" or "object". For example, each person is an entity, and bank accounts can be considered as entities. Entities are described in a database by a set of attributes. For example, the attributes ID, name, and salary may describe an EMPLOYEE entity. The attribute ID is used to identify EMPLOYEE uniquely.

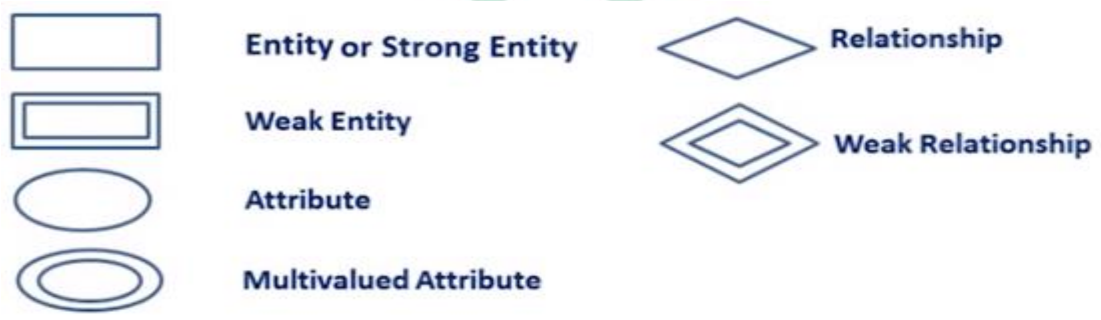
RELATIONSHIP: A relationship is an association among several entities. For example, a member relationship associates an instructor with his/her department.

The set of all entities of the same type and the set of all relationships of the same type are termed an ***entity se***t and ***relationship set***, *respectively*.

The overall structure (schema) of a database can be expressed graphically by an entity- relationship (E-R) diagram. An E-R diagram is represented with:

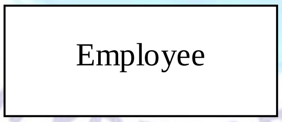
- ***Entity sets*** are represented by a partitioned rectangular box with the entity set name in the header and the attributes listed below it.
- ***Attributes*** are represented by Eclipse. These are used to describe the properties of an entity.
- ***Relationship sets*** are represented by a diamond connecting a pair of related entity sets. The name of the relationship is placed inside the diamond.

Symbols used in E-R Diagrams:

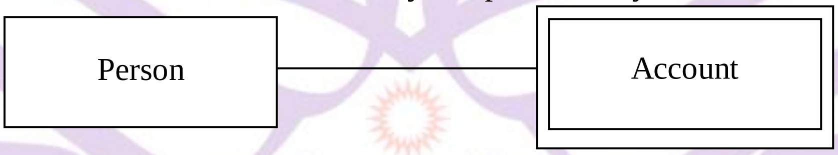


ENTITY SETS: An ***entity*** is a "thing" or "object" in the real world that is distinguishable from all other objects. For example, each person is an entity. An entity has a set of properties, called attributes.

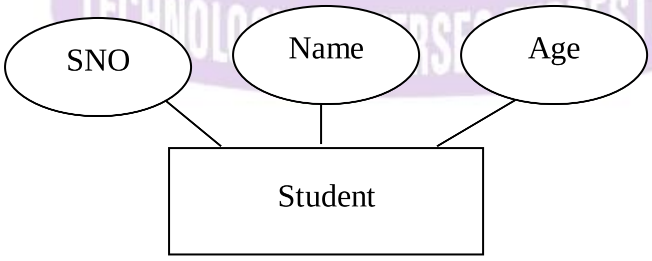
An ***entity set*** is a set of entities of the same type that share the same properties, or attributes. The set of all people who are employees at a company can be defined as the entity set Employee. It is represented as:



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.

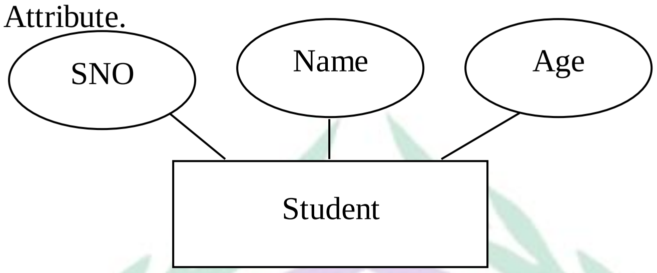


ATTRIBUTES: An attribute is used to describe the property of an entity. Eclipse is used to represent an attribute. For example, SNo, Name, Age, etc. can be attributes of a student.

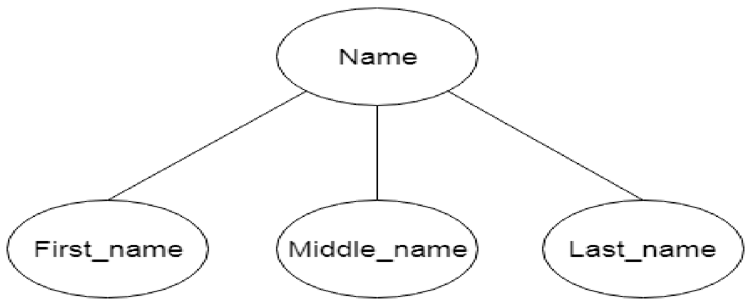


Types of Attributes in E-R Model:

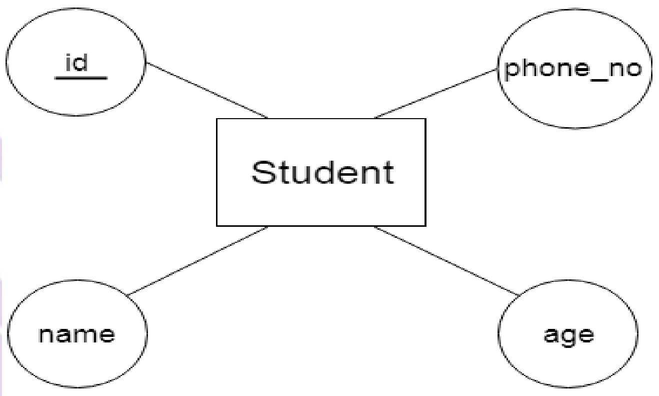
1) **Simple Attribute:** An attribute that contains atomic values, and which cannot be divided further is known as a Simple Attribute.



2) **Composite Attribute:** An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.

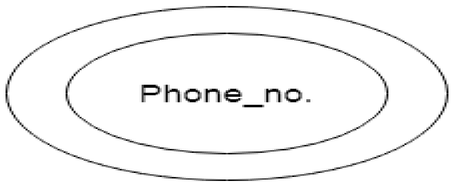


3) **Key Attribute:** The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.

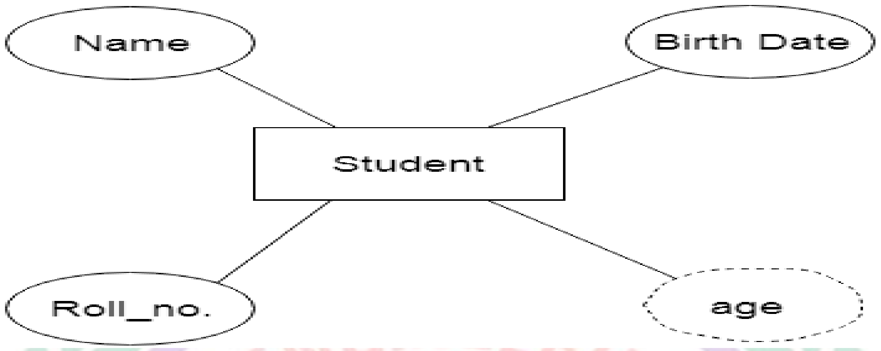


4) **Multivalued Attribute:** An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

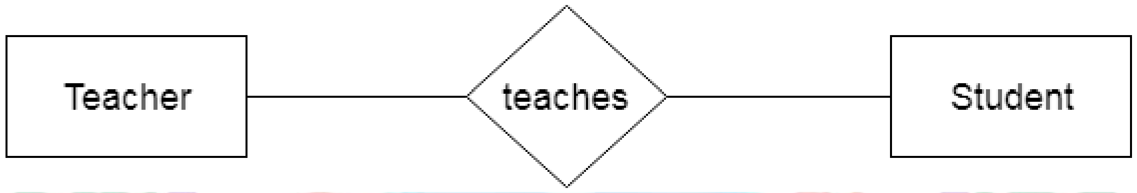
For example, a student can have more than one phone number.



5) **Derived Attribute:** An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.
   For example, A person's age can be derived from another attribute like Birth_Date.

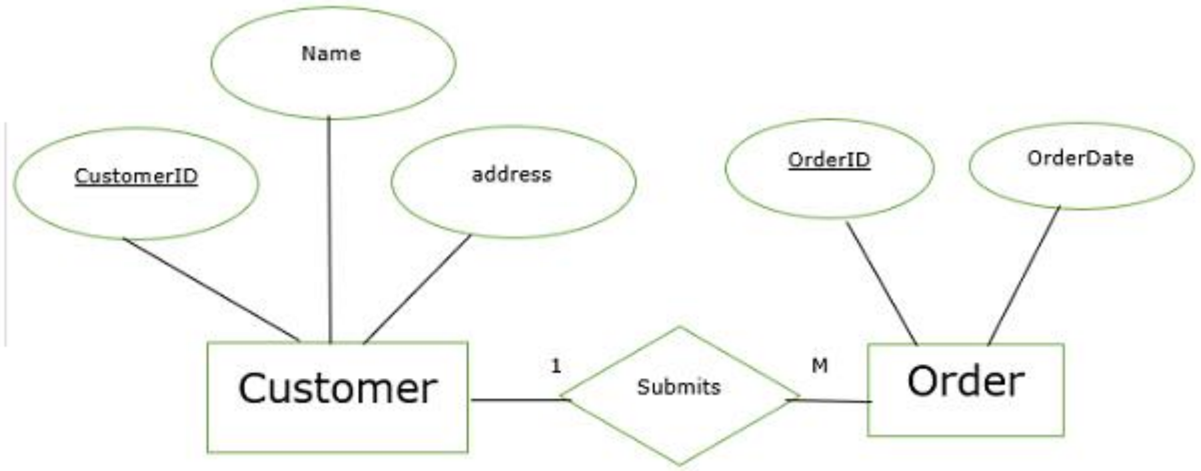


RELATIONSHIP: A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

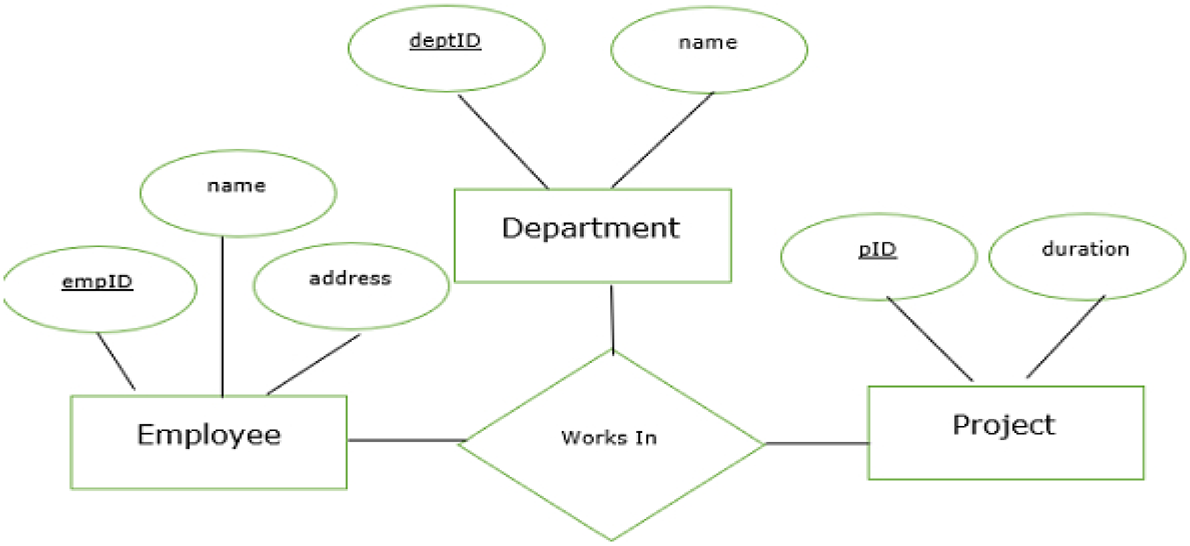


Types of Relationships E-R Diagrams:

1) **Binary Relationship:** E-R diagrams that contain two entity sets depict binary relationship. The degree of binary relationship is 2.



2) **Ternary E-R Diagrams:** E-R diagrams that have three entity sets show ternary relationship. The degree of ternary relationship is 3.

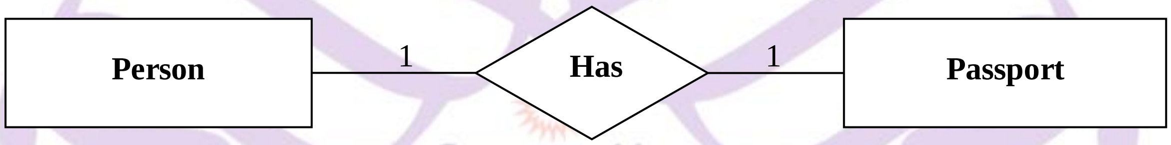


3) **N-ary Relationship:** n entities are involved in the relationship

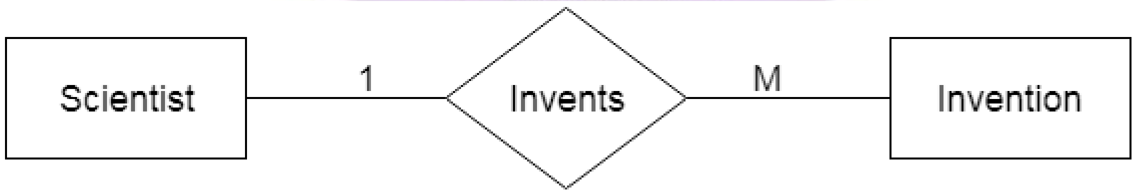
CONSTRAINTS: An E-R schema may define certain constraints to which the contents of a database must conform.

Mapping Cardinalities: Mapping cardinalities express the number of entities to which another entity can be associated via a relationship set. Mapping cardinalities are most useful in describing binary relationship sets. For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following:

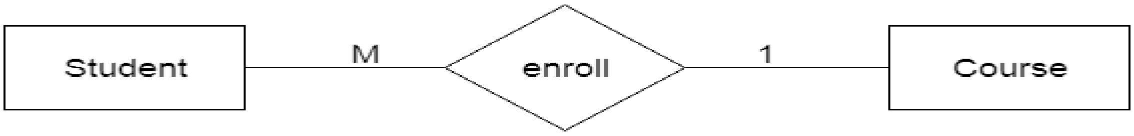
1) **One-to-One:** An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.
   For example, a Person can have only one Passport.



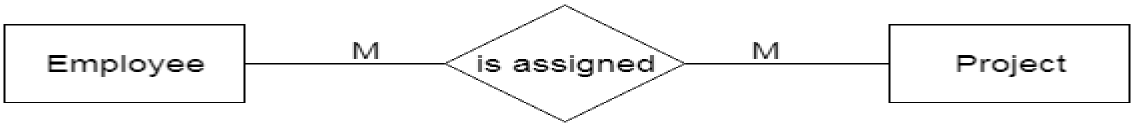
2) **One-to-Many:** An entity in A is associated with any number of entities in B. An entity in B, however, can be associated with at most one entity in A.
   For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



3) **Many-to-One:** An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number (zero or more) of entities in A.
For example, Student enrolls for only one course, but a course can have many students.

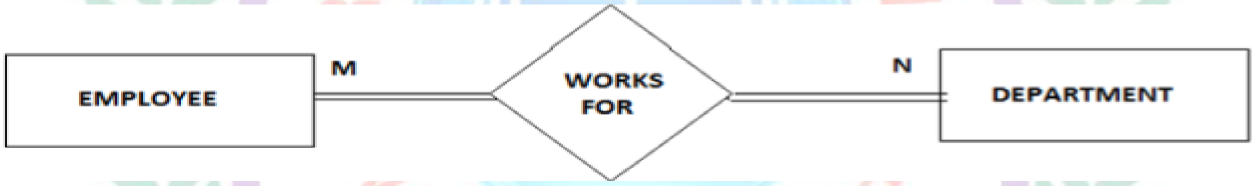


4) **Many-to-Many:** An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.
For example, Employee can assign by many projects and project can have many employees.

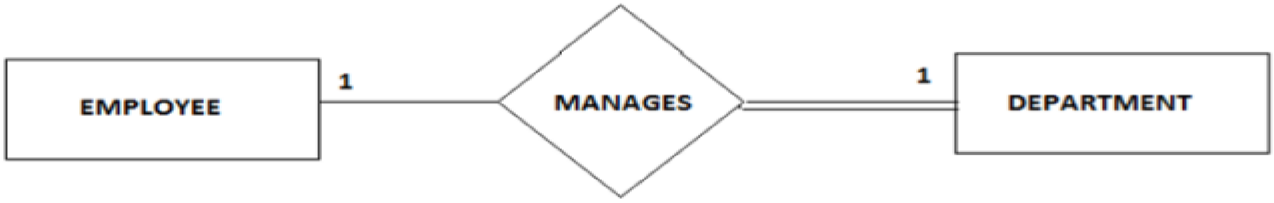


Participation Constraints: There are two types of Participation constraints.

1) **Total Participation:** Each entity in the entity set is involved in at least one relationship in a relationship set. Total Participation is represented by double line in ER diagram.



2) **Partial Participation:** Each entity in entity set may or may not occur in at least one relationship in a relationship set.



ER Design Issues: Users often mislead the concept of the entities, relationships and the design process of the ER diagram. Thus, it leads to a complex structure of the ER diagram and certain issues that does not meet the characteristics of the real-world enterprise model.

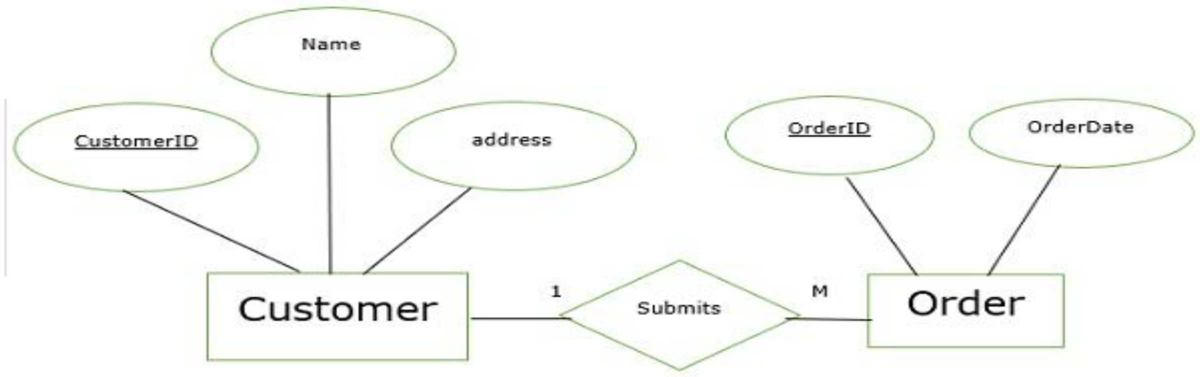
1) **Use of Entity Set vs Attributes:** Use of an entity set or attribute depends on the structure of the real-world enterprise that is being modelled. It leads to a mistake when the user use primary key of an entity set as an attribute of another entity set. Instead, he should use the relationship to do so.
2) **Use of Entity Set vs. Relationship Sets:** It is difficult to examine if an object can be best expressed by an entity set or relationship set. To understand and determine the right use, the user need to designate a relationship set for describing an action that occurs in-between the entities. If

there is a requirement of representing the object as a relationship set, then its better not to mix it with the entity set.

3) **Use of Binary vs n-ary Relationship Sets:** Generally, the relationships described in the databases are binary relationships. However, non-binary relationships can be represented by several binary relationships. For example, a ternary relationship can also be represented by two binary relationships.
4) **Placing Relationship Attributes:** The mapping cardinality can become an affective measure in the placement of the relationship attributes. So, it is better to associate the attributes of one-to-one or one-to-many relationship sets with any participating entity sets, instead of any relationship set.

Conversion of E-R Diagram to Table: The following rules are used to convert the ER diagram to tables and assign the mapping between the tables.

- Entity type becomes a table.
- All single-valued attribute becomes a column for the table.
- A key attribute of the entity type represented by the primary key.
- The multi valued attribute is represented by commas, or by a separate table.
- Composite attribute represented by components.
- Derived attributes are not considered in the table.



The above E-R diagram is converted as:

Customer Relation/Table

<u>CustomerID</u>	Name	Address

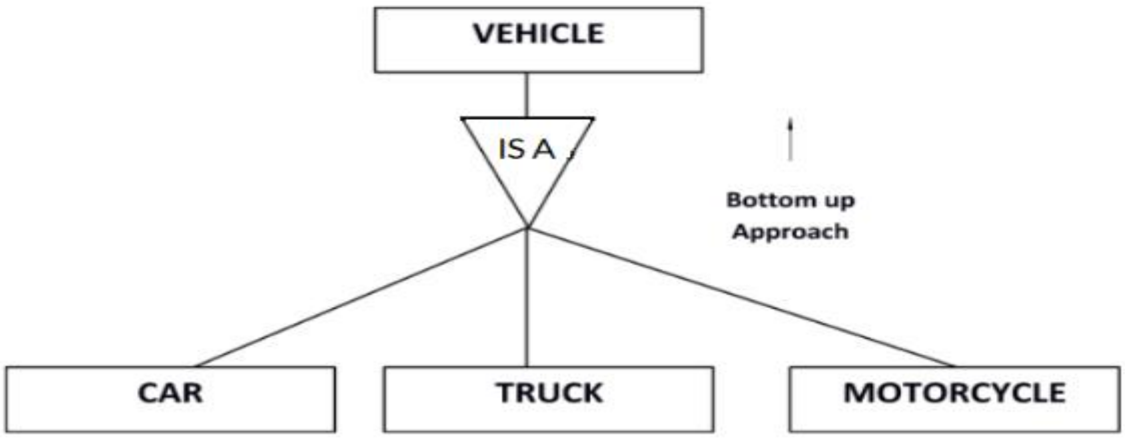
Order Relation/Table

<u>OrderID</u>	OrderDate

EXTENDED ER:

Generalization: Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.

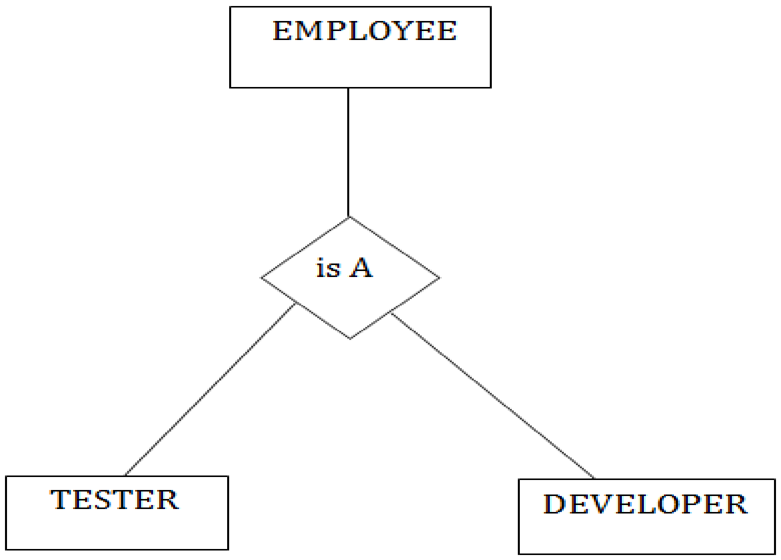
- Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.
- In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.



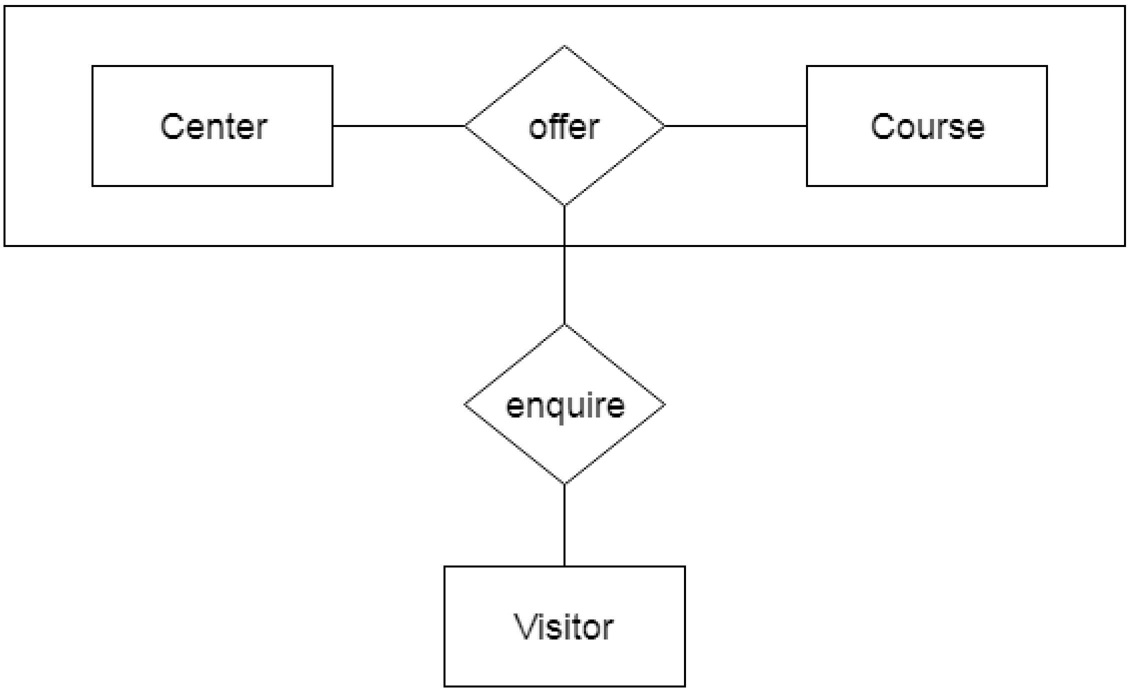
Specialization: Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.

- Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

For example, an EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.



Aggregation: In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.
For example, Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



FUNCTIONAL DEPENDENCIES:

The functional dependency is a relationship in which all non key attributes of the relation are dependent on Key attribute. In functional dependency, the key attribute determines all the non key attribute values. It typically exists between the primary key and non-key attribute within a table.

Functional dependency is usually shown as, A → B, where the left side of FD (A) is known as a determinant, and the right side of FD (B) is known as a dependent.

For example, we have a Student table with attributes: SNo, SName, Age. Here SNo attribute can uniquely identify the SName attribute of Student table because if we know the SNo, we can tell that student name associated with it.

Functional dependency can be written as: SNo → SName

We can say that SName is functionally dependent on SNo.

Types of Functional dependency:

1. **Trivial functional dependency:** A → B has trivial functional dependency if B is a subset of A. Dependencies like A → A, B → B are also trivial.

   Example: {Emp_Name, Exp} → Emp_Name

2. **Non-trivial functional dependency:** A → B has a non-trivial functional dependency if B is not a subset of A.

   Example: ID → Name, Name → DOB

- When A intersection B is NULL, then A → B is called as complete non-trivial.

Inference Rule (IR) / Armstrong's axioms: Armstrong's axioms are used to conclude functional dependencies on a relational database. There are 6 types of inference rules:

1. **Reflexive Rule (IR_1):** In the reflexive rule, if Y is a subset of X, then X determines Y.

   If X ⊇ Y then X → Y

   Example: X = {a, b, c, d, e}
   Y = {a, b, c}

2. **Augmentation Rule (IR_2):** The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.

   If X → Y then XZ → YZ

   Example: For R(ABCD), if A → B then AC → BC

3. **Transitive Rule (IR_3):** In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

   If X → Y and Y → Z then X → Z

4. **Union Rule (IR_4):** Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

   If X → Y and X → Z then X → YZ

5. **Decomposition Rule (IR_5):** Decomposition rule is also known as project rule. It is the reverse of union rule. This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

   If X → YZ then X → Y and X → Z

6. **Pseudo/Advanced transitive Rule (IR_6):** In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

   If X → Y and YZ → W then XZ → W

Canonical Cover: “A canonical cover or irreducible a set of functional dependencies FD is a simplified set of FD that has a similar closure as the original set FD”.

Extraneous attributes: “An attribute of an FD is said to be extraneous if we can remove it without changing the closure of the set of FD”.

NORMALIZATION: “Normalization is the process of organizing the data in the database to minimize the redundancy and eliminate Insertion, Update, and Deletion Anomalies from a relation”.

Normalization divides the larger table into smaller and links them using relationships. The normal form is used to reduce redundancy from the database table.

Relational Decomposition: Relational Decomposition is the process of breaking down a bigger relation/database into mulitple smaller relations. These smaller relations can be linked using relationships.

- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.

Types of Decomposition:

1. **Lossless Decomposition:** If the data is not lost from the relation that is decomposed, then the decomposition will be lossless. The lossless decomposition guarantees that the join of relations will result in the same relation as it was decomposed.
   - ✓ The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.
2. **Lossy Decomposition:** If the data is lost from the relation that is decomposed, then the decomposition will be a lossy decomposition. In this decomposition, natural joins of all the decomposition will not give the original relation as some data is lost.

Anomaly: If a database design is not perfect, it may contain anomalies which are usually the undesirable characteristics for a relation. Managing a database with anomalies is next to impossible. The types of anomalies in DBMS include:

- **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
- **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
- **Updatation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

✓ Normalization is a method to remove all these anomalies and bring the database to a consistent state.

Types of Normal Forms: Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints. Types of Normal Forms in DBMS are:

1. 1NF (First Normal Form)
2. 2NF (Second Normal Form)
3. 3NF (Third Normal Form)
4. BCNF (Boyce Codd Normal Form)
5. 4NF (Fourth Normal Form)
6. 5NF (Fifth Normal Form)

1. **1NF (First Normal Form):** A relation will be 1NF if it contains an atomic value. It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.

Example: Relation Student is not in 1NF as it has a multi valued attribute MobileNo.

Student Table:

SNo	SName	Age	MobileNo
101	Ravi	20	9966012345, 9963012345
102	Rani	21	9848012345
103	Raju	19	9849012345, 9866012345

The decomposition of Student relation into 1NF is shown as:

Student Table:

SNo	SName	Age	MobileNo
101	Ravi	20	9966012345
101	Ravi	20	9963012345
102	Rani	21	9848012345
103	Raju	19	9849012345
103	Raju	19	9866012345

2. **2NF (Second Normal Form):** A relation is said to be in 2NF, if it is already in 1NF and if it does not contain any Partial Functional Dependency. In 2NF, all non-key attributes are fully functional dependent on the primary key.

Example: As given in class notes

3. **3NF (Third Normal Form):** A relation is said to be in 3NF, if it is already in 2NF and if it does not contain any Transitive Dependency. 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.

- ✓ A relation is in 3NF if for every non-trivial function dependency X → Y. X may be a super key.

Example: As given in class notes

4. **BCNF (Boyce Codd Normal Form):** BCNF is the advance version of 3NF. It is stricter than 3NF. For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

- ✓ A table is in BCNF if every functional dependency X → Y, X must be a super key of the table.

5. **4NF (Fourth Normal Form):** A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
    - For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

Example: As given in class notes

6. **5NF (Fifth Normal Form):** A relation is in 5NF if it is in 4NF and does not contain any join dependency and joining should be lossless. 5NF is also known as Project-join normal form (PJ/NF).

- ✓ 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.

Example: As given in class notes

Advantages of Normalization:

- ✓ Normalization helps to minimize data redundancy.
- ✓ Greater overall database organization.
- ✓ Data consistency within the database.
- ✓ Much more flexible database design.
- ✓ Enforces the concept of relational integrity.

Disadvantages of Normalization:

- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.

DOMAIN KEY NORMAL FORM: A relation is in DKNF it satisfies all the constraints from 1NF to 5NF, i.e., when the relation does not contain any anomalies and has minimum redundancy is it said to be in DKNF. Domain-Key Normal Form is the highest form of Normalization. The reason is that the insertion and updation anomalies are removed. It is often known as 6NF (Sixth Normal Form).

DKNF is a normal form used which requires that the database contains only domain constraints and key constraints.

Domain Constraints: These are constraints on domain values such as values of an attribute had some set of values. For example, EmpID should be four digits long.

EmpID	**EmpName**	**Age**
0111	Virat	33
0222	Rohit	34
0333	Rahul	29

Key Constraint: These are constraints on the type of key to be used for the database.

DENORMALIZATION: Denormalization is a technique that allows DBAs to add redundant data into a normalized database to alleviate issues with database queries that merge data from several tables into a single table.

Advantages:

1. Enhance Query Performance
2. Make database more convenient to manage

Disadvantages:

1. It takes large storage due to data redundancy.
2. It makes it expensive to updates and inserts data in a table.
3. It makes update and inserts code harder to write.