**CHAPTER 3**

**PROPOSED WORK**

**3.1 ER MODEL**

An Entity relationship diagram shows the relationships of entity sets stored in databases. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

An entity relationship diagram is a snapshot of data structure. An entity relationship diagram shows entities (tables) in a database and relationships between tables within that database. For a good database design it is essential to have an entity relationship diagram.

There are three basic elements in entity relationship diagram

* Entities are the things for which we want to store information. An entity is a person, place, thing or event.
* Attributes are the data we want to collect for an entity.
* Relationships describe the relations between the entities.

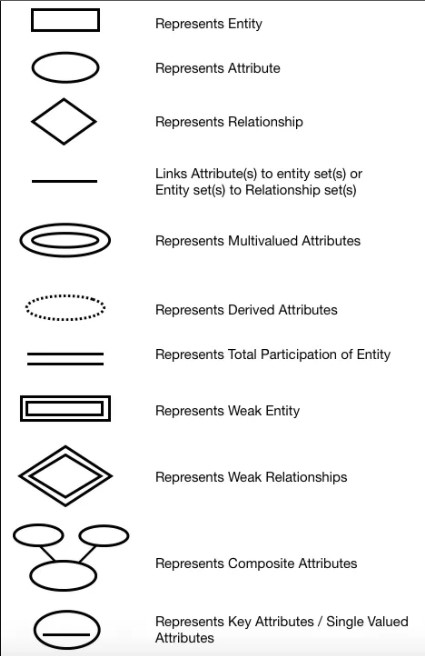
An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system’s entities and the relationships between those entities. An entity-relationship diagram is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an entity-relationship diagram are:

* Entities
* Relationships
* Attributes

Steps involved in creating an entity-relationship diagram include:

* Identifying and defining the entities.
* Determining all interactions between the entities .
* Analyzing the nature of interactions/determining the cardinality of the relationship
* Creating the entity-relationship diagram.



**3.1 Notations for ER Diagram**

**Components of an E-R diagram:**

An E-R diagram constitutes the following Components:

**1. Entity: -** Any real-world object can be represented as an entity about which data can be stored in a database. All the real world objects like a book, an organization, a product, a car, a person are the examples of an entity. Any living or non-living objects can be represented by an entity. An entity is symbolically represented by a rectangle. Entities can be characterized into two types:

* **Strong entity:** A strong entity has a primary key attribute which uniquely identifies each entity. Symbol of a strong entity is the same as an entity.
* **Weak entity:** A weak entity does not have a primary key attribute and depends on another entity via a foreign key attribute.

**2**. **Attribute: -** Each entity has a set of properties. These properties of each entity are termed as attributes. For example, a car entity would be described by attributes such as price, registration number, model number, colour etc. Attributes are indicated by ovals in an e-r diagram.

A primary key attribute is depicted by an underline in the e-r diagram. An attribute can be characterized into following types:

* **Simple attribute: -** An attribute is classified as a simple attribute if it cannot be partitioned into smaller components. For example, age and sex of a person. A simple attribute is represented by an oval.
* **Composite attribute: -** A composite attribute can be subdivided into smaller components which further form attributes. For example, ‘name’ attribute of an entity “person” can be broken down into first name and last name which further form attributes. Grouping of these related attributes forms a composite attribute. ‘name is the composite attribute in this example.
* **Single valued attribute: -** If an attribute of a particular entity represents a single value for each instance, then it is called a single-valued attribute. For example, Ramesh, Kamal and Suraj are the instances of entity ‘student’ and each of them is issued a separate roll number. A single oval is used to represent this attribute.
* **Multi valued attribute: -** An attribute which can hold more than one value, it is then termed as multi-valued attribute. For example, the phone number of a person.
* **Derived attribute:** -A derived attribute calculates its value from another attribute. For example, ‘age’ is a derived attribute if it calculates its value from ‘current date’ & ‘birth date’ attributes. A derived attribute is represented by a dashed oval.
* **Relationships: -** A relationship is defined as a bond or attachment between 2 or more entities. Normally, a verb in a sentence signifies a relationship.

For example,

3.1 An employee assigned a project.

3.2 Teacher teaches a student.

3.3 Author writes a book.

A diamond is used to symbolically represent a relationship in the e-r diagram.

**3. Various terms related to relationships**

**1. Degree of relationship: -** It signifies the number of entities involved in a relationship. Degree of a relationship can be classified into following types:

* **Unary relationship: -** If only a single entity is involved in a relationship then it is a unary relationship. For example, An employee (manager) supervises another employee.
* **Binary relationships: -** when two entities are associated to form a relation, then it is known as a binary relationship. For example, A person works in a company. Most of the time we use only binary relationships in an e-r diagram.
* Other types of relationships are ternary and quaternary. As the name signifies, a ternary relationship is associated with three entities and a quaternary relationship is associated with four entities.

**4. Connectivity of a relationship: -** Connectivity of a relationship describes how many instances of one entity type are linked to how many instances of another entity type. Various categories of connectivity of a relationship are;

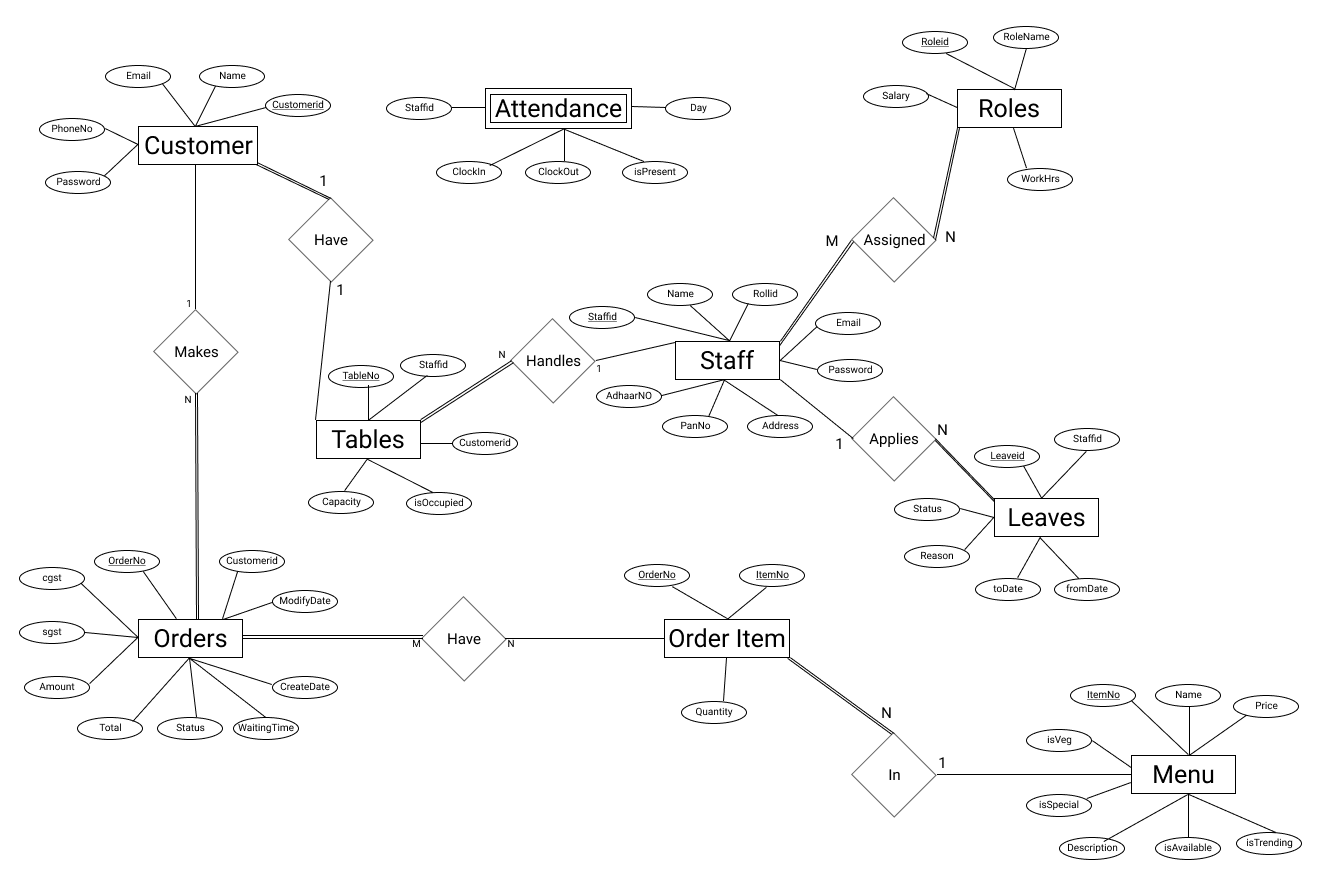
* **One to One (1:1) –** “Student allotted a project” signifies a one-to-one relationship because only one instance of an entity is related with exactly one instance of another entity type.
* **One to Many (1:M) –** “A department recruit’s faculty” is a one-to-many relationship because a department can recruit more than one faculty, but a faculty member is related to only one department.
* **Many to One (M:1) –** “Many houses are owned by a person” is a many-to-one relationship because a person can own many houses but a particular house is owned only a person.
* **Many to Many (M: N) –** “Author writes books” is a many-to-many relationship because an author can write many books and a book can be written by many authors**.**

**Database Tables / Relations**

A table is a collection of related data held in a structured format within a database. It consists of columns, and rows. In relational databases, and flat file databases, a table is a set of data elements (values) using a model of vertical columns (identifiable by name) and horizontal rows, the cell being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows.

Each row in a table is identified by one or more values appearing in a particular column subset. The columns subset which uniquely identifies a row is called the primary key. A table can be considered as a convenient representation of a relation. A table is perceived a two-dimensional structure composed of rows and columns. The order of rows and columns is immaterial to the DBMS.

The Entity Relationship diagram for this project is as follows:



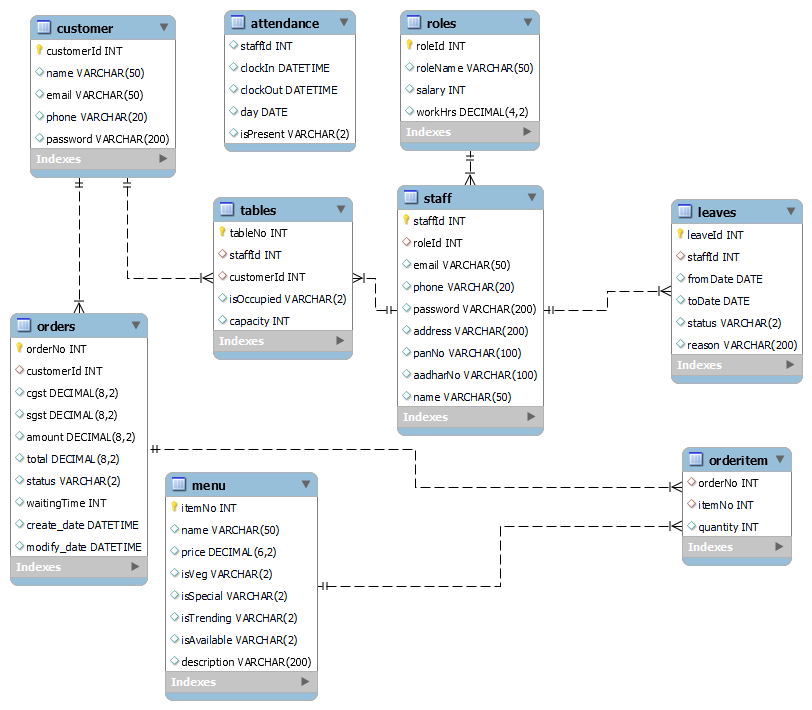
**3.3 ER Diagram for Restaurant Management System**

**3.2 RELATIONAL MODEL**

The relational model (RM) for [database](https://en.wikipedia.org/wiki/Database) management is an approach to managing [data](https://en.wikipedia.org/wiki/Data) using a [structure](https://en.wikipedia.org/wiki/Structure_(mathematical_logic)) and language consistent with [first-order predicate logic](https://en.wikipedia.org/wiki/First-order_logic), first described in 1969 by English computer scientist [Edgar F. Codd](https://en.wikipedia.org/wiki/Edgar_F._Codd), where all data is represented in terms of [tuples](https://en.wikipedia.org/wiki/Tuple), grouped into [relations](https://en.wikipedia.org/wiki/Relation_(database)). A database organized in terms of the relational model is a [relational database](https://en.wikipedia.org/wiki/Relational_database).

The purpose of the relational model is to provide a [declarative](https://en.wikipedia.org/wiki/Declarative_programming) method for specifying data and queries: users directly state what information the database contains and what information they want from it, and let the database management system software take care of describing data structures for storing the data and retrieval procedures for answering queries.

Most relational databases use the [SQL](https://en.wikipedia.org/wiki/SQL) data definition and query language; these systems implement what can be regarded as an engineering approximation to the relational model. A table in an SQL [database schema](https://en.wikipedia.org/wiki/Database_schema) corresponds to a predicate variable; the contents of a table to a relation; key constraints, other constraints, and SQL queries correspond to predicates. However, SQL databases [deviate from the relational model in many details](https://en.wikipedia.org/wiki/Relational_model#SQL_and_the_relational_model), and Codd fiercely argued against deviations that compromise the original principles.

The Relational Model for this Project is shown below:

**3.3 Relational Model for Restaurant Management System**

**3.3 NORMALIZATION**

The normalization process was proposed by Codd, it takes a relation schema through a series of tests to certify whether it satisfies a certain normal form. The process proceeds in a top down fashion by evaluating each relation against the criteria for normal forms and decomposing relations as necessary can thus be considered a relational design by analysis.

Normalization of data can be looked upon as a process of analyzing the given relation schemas based on their Functional Dependencies and primary keys to achieve the desirable properties of:

1. Minimizing redundancy.

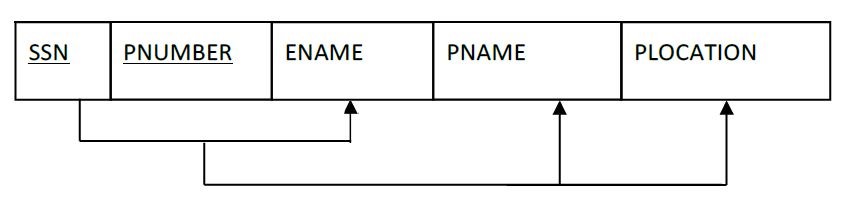
2. Minimizing the insertion, deletion, and update anomalies.

**Functional Dependency:**

The Functional Dependency denoted by X →Y, between two sets of attributes X and Y that are subsets of R species a constraint on the possible tuples that can form a relation stare r of R. The constraints is that for any two tuples t1 and t2 in r that have t1[X] = t2[X].

This means that the values of the Y component of a tuple in r depend on, or determined by the values of the X components. Alternatively the values of the X component of a tuple uniquely determine the values of the Y component. Consider the following schema,

### **EMP\_PRJ**



In the above schema the functional dependencies are:

1. SSN → ENAME

The value of an employee’s social security number (SSN) uniquely determines the employee name (ENAME).

2. PNUMBER →{ PNAME, PLOCATION }

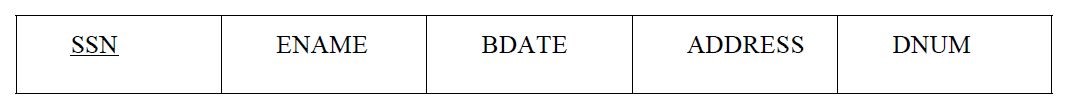
These values of project’s number uniquely determines the project name (PNAME) and project locations (PLOCATIONS).

**Super key:**

A super key of a relation schema R= {A1, A2… An} is an asset of attributes S *C*​​R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[S]. A key K is a super key with the additional property that removal of any attribute from K will cause K not to be a super key anymore

For example consider the following schema,

### **EMPLOYEE**



Here{SSN} is the primary for EMPLOYEE, where as {SSN}, {SSN, ENAME}, {SSN, ENAME, BDATE} and any set of attributes that includes SSN are called super keys.

**Normal Forms:**

There are three normal forms

1. First normal form
2. Second normal form
3. Third normal form

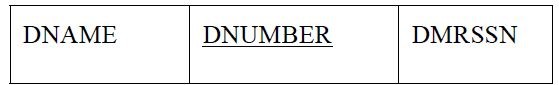
These were proposed by Codd as a sequence to achieve the desirable state of 3NF​ relations by progressing through the intermediate states of 1Nf and 2NF if needed.

* **First Normal Form (1NF):**

It states that the domain of attribute must include only atomic be values and that the value of any attribute in a tuple must be a single value from the domain of the attribute. Hence 1NF disallows having a set of values a tuple of values or a combination of both as an attribute value for a single tuple.

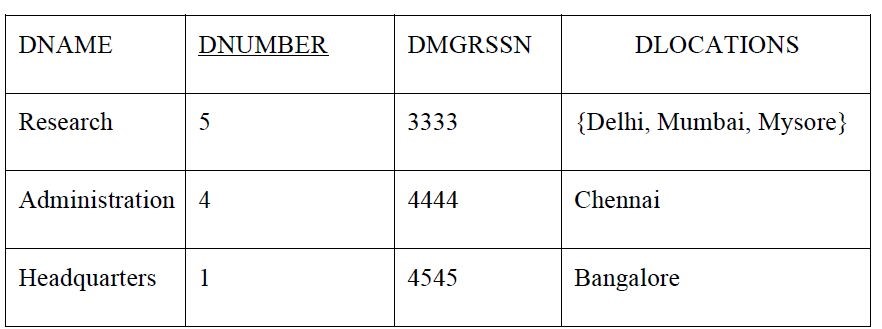
Consider the following department relation schema,

### **DEPARTMENT**



where the primary key is DNUMBER and suppose that we extend it by including the DLOCATIONS as shown in below Figure. We assume that each department can have a number of locations.

### **DEPARTMENT**



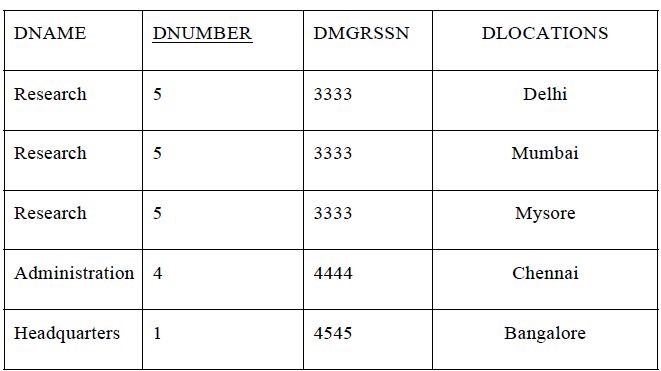
Above relation table is not in 1NF because DLOCATIONS are not atomic attributes. It can be converted into 1NF by following methods:

* Remove the attribute DLOCATION that violates 1NF and place it in a separate relation DEPT\_LOCATIONS as shown in Figure above.
* Expand the key so that there will be a separate tuple in the original DEPARTMENT relation for each location of a DEPARTMENT as shown in Figure below.

### **DEPT\_LOCATIONS**



### **DEPARTMENT**

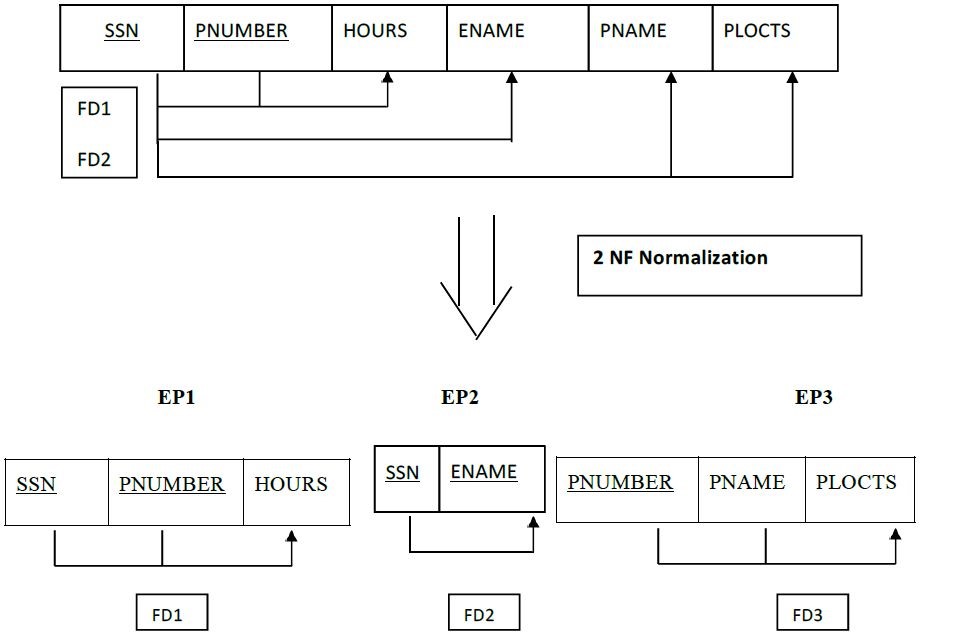


**● Second Normal Form (2NF):**

This normal form is based on the **full**​ **functional dependency**.​ A functional dependency X→Y is full functional dependency if removal for any attribute A from X means that dependency does not hold any more.

**A relation schema R is in 2NF if every non prime attribute A in R is fully functionally dependent on the primary key of R.**

The EMP\_PROJ in figure is in 1NF but not in 2NF. The non prime attribute ENAME violates 2NF because of FD3. The functional dependencies FD2 and FD3 make ENAME, PNAME and PLOCATION partially dependent on the primary key {SSN,PNUMBER} of EMP\_PROJ thus violating 2NF. The relational schema in Figure can be second normalized into a number of 2NF relations EP1, EP2 and EP3 as shown in Figure below.



**3.4 Normalization of EMP\_PROJ into 2NF**

* **Third Normal Form:**

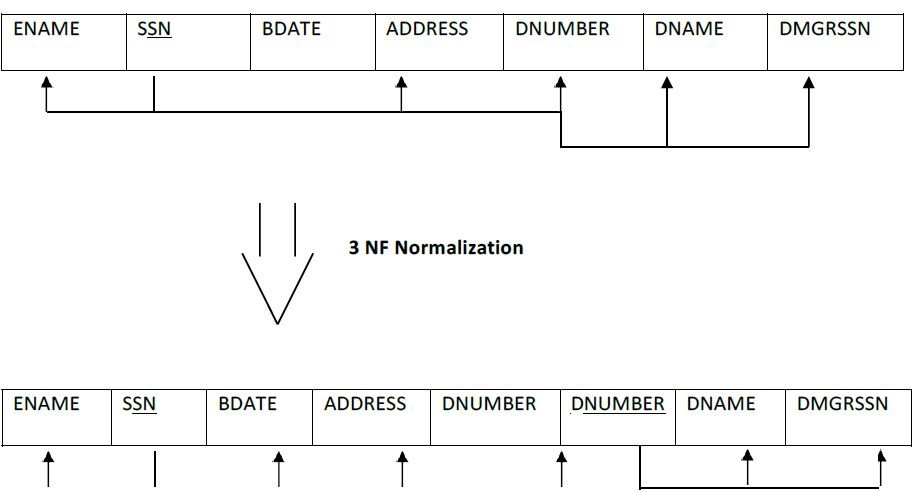
Third normal form (3NF) is based on the concept of transitive dependency. A functional dependency X→Y in a relation schema R is a **transitive**​ **dependency** if​ there is a set of attributes Z that is neither a candidate key nor a subset of any key of R and both X→Z and Z→Y hold.

**A relation schema R is in 3Nf if it satisfies 2NF and no non prime attribute of R**

**is transitively dependent on the primary key.**

The relation schema EMP\_DEPT in Fig 3.4 is in 2NF however it is not in 3NF because transitive dependency of DMRSSN on SSN via DNUMER. We can normalize EMP\_DEPT by decomposing it into two 3NF relation schema ED1 and ED2 as shown in the Fig 3.5.

### **EMP\_DEPT**



**3.5 Normalizing EMP\_DEPT into 3NF relations**