CHAPTER 2.DATABASE MANAGEMENT SYSTEM

As mentioned, DBMS is a collection of programs that enables users to create and maintain a database The DBMS is a general purpose software system that facilitates the processes of defining, constructing, manipulating and sharing databases among various users and applications.

- **Characteristics of Database Management Systems:**
  - Self-describing nature.
  - Keeps a tight control on data redundancy.
  - Enforces user defined rules to ensure that integrity of table data.
  - Provides insulation between Programs and data, Data abstraction.
  - Supports multiple views of the data.
  - Helps sharing of data and Multi-user transaction processing.

- **Advantages of using the DBMS approach:**
  - Controlling the redundancy.
  - Restricting unauthorized access.
  - Providing persistent storage for program objects.
  - Providing storage structures for efficient query processing.
  - Providing backup and recovery.
  - Providing multiple users interfaces.
  - Representing complex relationships among data.
  - Enforcing integrity constraints.

2.1 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 state r of R. The constraints is that for any two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$.

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

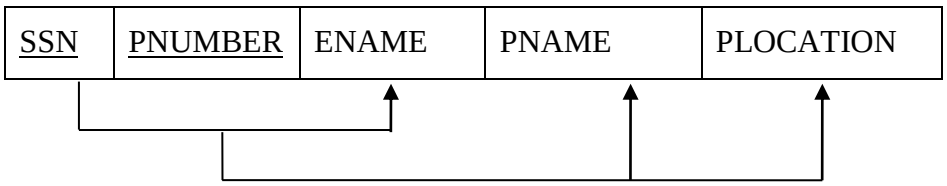

Fig.2.1: Employee Project Schema

In the above schema the functional dependencies

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)

2.2 Normalization:

The normalization process was proposed by Codd, it takes a relation schema through a series of tests to certify whether if 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

(i) Minimizing redundancy.

(ii) Minimizing the insertion, deletion, and update anomalies.

Super key:

A super key of a relation schema R= {A_1, A_2… A_n} is asset of attributes S $\underline{C}$ R with the property that no two tuples t_1 and t_2 in any legal relation state r of R will have $t_1[S] = t_2[S]$. 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

SSN	ENAME	BDATE	ADDRESS	DNUM

Fig.2.2: Employee Schema

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

If a relation schema has more than one key, each key is called **candidate key**. An attribute of relation schema R is called a **prime attribute** of R if it is a member of some candidate key of R. An attribute is called **nonprime** if it is not a prime attribute or it is not a member of any candidate.

Normal Forms:

There are three normal forms

(i) First normal form

(ii) Second normal form

(iii)Third normal form

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

(i) 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

DNAME	DNUMBER	DMRSSN

Fig.2.3: Department Schema

where, the primary key is DNUMBER and suppose that we extend it by including the DLOCATIONS as shown in below fig 2.4. We assume that each department can have a number of locations.

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	3333	{Delhi, Mumbai, Mysore}
Administration	4	4444	Chennai
Headquarters	1	4545	Bangalore

Fig.2.4: Department Table

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

(i) Remove the attribute DLOCATION that violates 1NF and place it in a separate relation DEPT_LOCATIONS as shown in fig 2.5.

(ii) 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 fig 2.6.

DEPT_LOCATIONS

DNUMBER	DLOCATION
1	Bangalore
2	Chennai
4	Delhi
5	Mumbai
5	Mysore

Fig.2.5: Dept_Locations Table

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	3333	Delhi
Research	5	3333	Mumbai
Research	5	3333	Mysore
Administration	4	4444	Chennai
Headquarters	1	4545	Bangalore

Fig.2.6: Department Table

(ii) 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 nonprime attribute An in R is fully functionally dependent on the primary key of R.

The EMP_PROJ in fig 2.7a 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 fig 2.7a can be second normalized into number of 2NF relations EP1, EP2 and EP3 as shown in the fig 2.7.

EMP_PROJ (Fig.2.7a: Relation Schema EMP_PROJ)

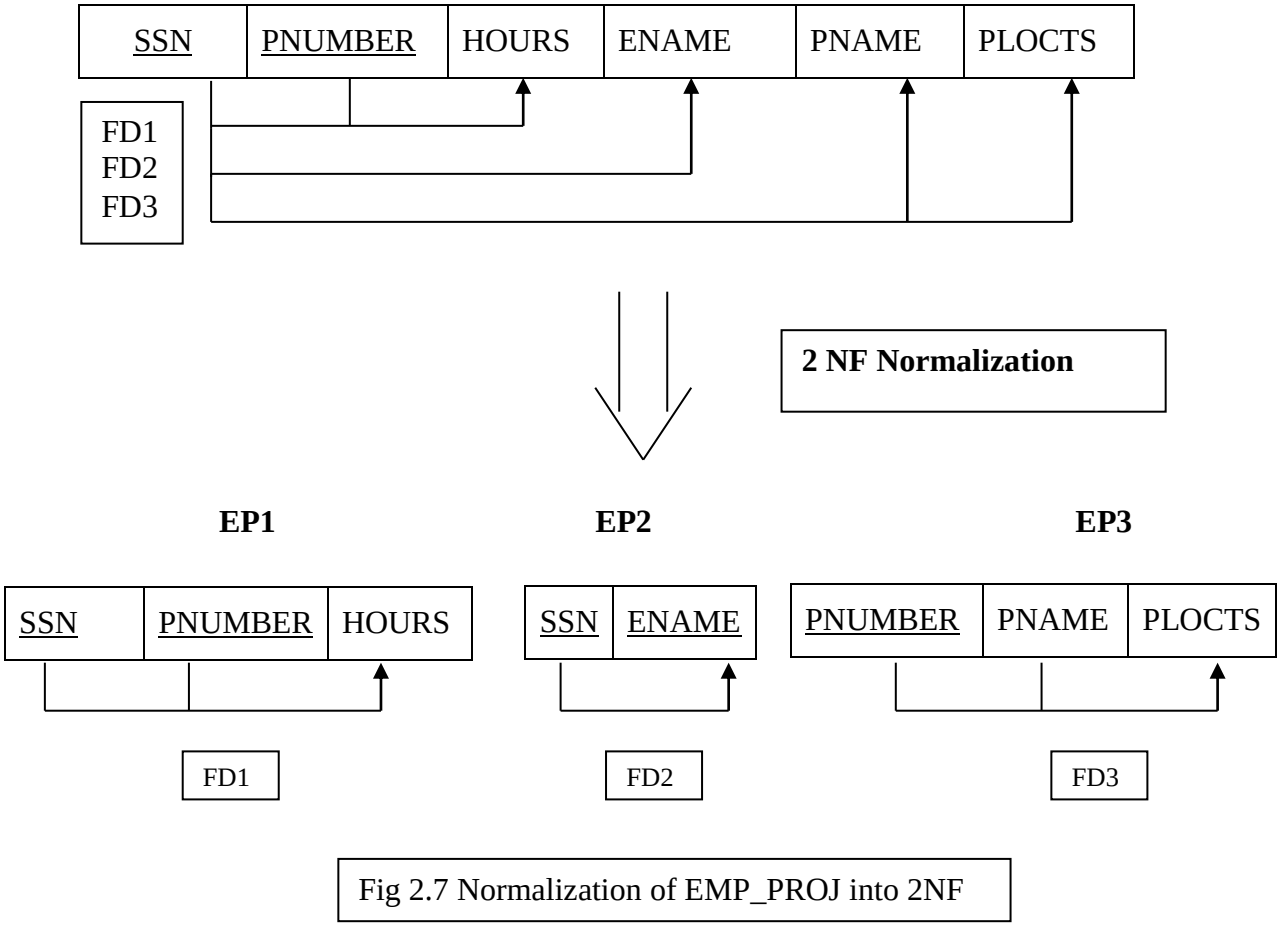

Fig 2.7 Normalization of EMP_PROJ into 2NF

(iii) 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 nonprime attribute of R is transitively dependent on the primary key.

The relation schema EMP_DEPT in fig 2.8a 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 2.8.

Fig.2.8a: Relation Schema EMP_DEPT

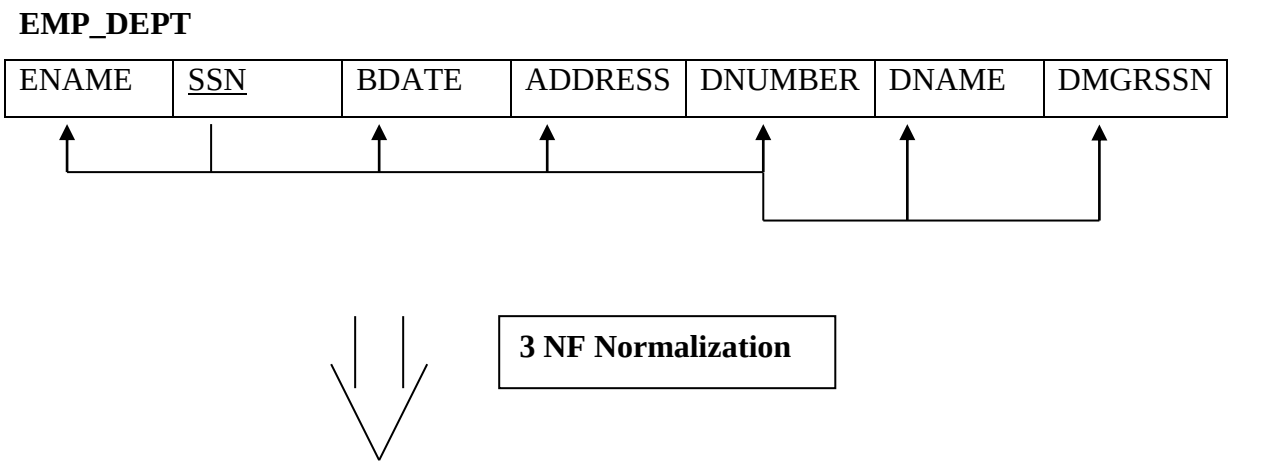

Fig 2.8 Normalizing EMP_DEPT into 3NF relations

2.3 Structured Query Language (SQL):

The ANSI standard SQL provides basic functions for data manipulation, transaction control, and record retrieval from the database. However, most end users interact with Oracle through application that provides an interface that hides the underlying SQL and its complexity.

SQL uses the terms table, row, and column for relation, tuple, and attribute, respectively. The SQL commands for data definition are CREATE, ALTER and DROP.

- **CREATE**

This command is used to create table or view by giving it a name and specifying its attributes and constraints. The attributes are specified first, and each attribute is given a name, a data type to specify its domain values, and any attribute constraints such as NOT NULL.

SYNTAX: CREATE TABLE <TNAME> (ATR1 TYP1 CONST1, ATR2 TYP2 CONST,…)

- **ALTER**

The definition of a base table can be altered by ALTER command which is a Schema Evolution command. The possible ALTER TABLE include adding or dropping a column (attribute), changing a column definition, and adding or dropping table constraints.

Example: ALTER TABLE *STUDENT* ADD *NAME* VARCHAR (12)

- **DROP**

If a whole schema is not needed any more, the DROP SCHEMA command can be used. There are two drip behavior options: CASCADE and RESTRICT.

CASCADE option is used to remove the database schema and all its tables, domains and other elements.

If the RESTRICT option is chosen in place of CASCADE, the schema is dropped only it has no elements in it; otherwise, the DROP command will not be executed.

SYNTAX: DROP TABLE *STUDENT* CASCADE

Statements in SQL:

Following are the important statements used in SQL.

(i) SELECT - Used to retrieve the information from the relation.

(ii) INSERT - Used to insert the new values to the relation.

(iii) DELETE - used to delete one or more existing tuples from the relation.

(iv) UPDATE - Used to update already existing values in the relation.

Aggregate Functions in SQL:

Following aggregate functions are provided by the SQL.

(i) COUNT - Returns number of tuples.

(ii) SUM - Returns sum of entries in a column.

(iii) MAX - Returns Maximum value from an entire column.

(iv) MIN - Returns Minimum value from an entire column.

(v) AVG - Returns Average of all the entries in a column.

Constraints in SQL

Following constraints are provided by the SQL.

(i) NOT NULL - Column should contain some value.

(ii) PRIMARY KEY - Should not allow duplicate and null values to a column.

(iii) UNIQUE - Each value of a column should be unique.