**HOSPITAL MANAGEMENT SYSTEM**

**Submitted By : Akshay Agarwal (IPG-2015006)**

**Aditya Gupta (IPG-2015004)**

**Rishabh Jain (IPG-2015076)**

**Shashank Bothra (IPG-2015092)**

**INDEX**

1. INTRODUCTION

2. RELATIONAL MODEL

1. Relational Tables’ Descriptions
2. Primary Keys and Foreign Keys
3. Cardinality

3. E-R DIAGRAM.

4. RELATIONAL DATABASE DESIGN

* 1. Functional Dependency
  2. Normalization

5.IMPLEMENTATION ON SQL SERVER

* 1. Relational model Implementation on SQL Server

6. IMPLEMENTATION ON A WEBSITE

7. CONCLUSION

Introduction:

The project Hospital Management system includes registration of patients, storing their details into the system, and also computerized billing in the pharmacy, and labs. The software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically.

It includes a facility to add and delete appointments. User can search availability of a doctor and the details of a patient using the id. The Hospital Management System can be entered using a username and password.

It is accessible either by an administrator or receptionist. Only they can add data into the database. The data can be retrieved easily. The interface is very user-friendly. The data are well protected for personal use and makes the data processing very fast. Hospital Management System is powerful, flexible, and easy to use and is designed and developed to deliver real conceivable benefits to hospitals. Hospital Management System is designed for multi-speciality hospitals, to cover a wide range of hospital administration and management processes.

It is an integrated end-to-end Hospital Management System that provides relevant information across the hospital to support effective decision making for patient care, hospital administration and critical financial accounting, in a seamless flow. Hospital Management System is a software product suite designed to improve the quality and management of hospital management in the areas of clinical process analysis and activity-based costing. Hospital Management System enables you to develop your organization and improve its effectiveness and quality of work. Managing the key processes efficiently is critical to the success of the hospital helps you manage your processes.

**Relational Model:**

**Patient Table:-**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pat\_id | Pat\_name | Pat\_Age | Pat\_Sex | Pat\_Address | Pat\_DOB | Pat\_MOB |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Receptionist Table:-**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rcp\_id | Rcp\_name | Rcp\_Age | Rcp\_Address | Rcp\_MOB | shifting | Rcp\_salary |
|  |  |  |  |  |  |  |

**Admission Table:-**

This is a junction table between Patient, Receptionist tables.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Admsn\_id | Pat\_id | Rcp\_id | date | time |
|  |  |  |  |  |

**Doctor Table:-**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Doc\_id | Doc\_name | Doc\_type | Doc\_age | Doc\_address | Doc\_MOB | Passed\_from | Doc\_salary |
|  |  |  |  |  |  |  |  |

**Appointment Table:-**

This is a junction table between Receptionist & Doctor tables.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ap\_id | Pat\_name | Doc\_id | Rcp\_id | apnmt\_date | apnmt\_time |
|  |  |  |  |  |  |

**Bill Table:-**

|  |  |  |  |
| --- | --- | --- | --- |
| Bill\_id | Bill\_for | Bill\_type | Bill\_ total |
|  |  |  |  |

**Payment Table:-**

This is a junction table between Patient, Bill Tables.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pay\_id | Bill\_for | Pat\_id | Pay\_type | Pay\_date |
|  |  |  |  |  |

**Medicine Table:-**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mdcn\_id | Mdcn\_name | Company | m\_date | e\_date | price |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Prescription Table:** -

This is a junction table between Patient, Doctor & Medicine tables.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prs\_id | Doc\_id | Mdcn\_id | Pat\_id | date | Fee |
|  |  |  |  |  |  |

**Department Table:-**

Dept\_id Dept\_name treatment

**Doctor\_from\_Department Table:-**

This is a junction table between Doctor & Department tables.

Dfd\_id Doc\_id Dept\_id

**Relational Tables’ Descriptions:**

Patient table

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Comments |
|  |  |  |
| Pat\_id | int | Unique id for a Patient |
|  |  |  |
| Pat\_name | varchar(20) | Patient’s Name |
|  |  |  |
| Pat\_Age | int | Patient’s Age |
|  |  |  |
| Pat\_Sex | varchar(20) | Patient is Male or |
|  |  | Female |
|  |  |  |
| Pat\_Address | varchar(20) | Patient’s Address |
|  |  |  |
| Pat\_Dob | varchar(20) | Date of Birth |
|  |  |  |
| Pat\_Mob | Bigint | Mobile Number |
|  |  |  |
|  |  |  |

**Receptionist table**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Comments |
|  |  |  |
| Rcp\_id | int | Unique id for a |
|  |  | Receptionist |
|  |  |  |
| Rcp\_name | varchar(20) | Receptionist’s name |
|  |  |  |
| Rcp\_Age | int | Receptionist’s age |
|  |  |  |
| Rcp\_Address | varchar(20) | Receptionist’s Address |
|  |  |  |
| Rcp\_MOB | Bigint | Mobile Number |
|  |  |  |
| Shifting | varchar(20) | Receptionist working shift |
|  |  |  |
| Rcp\_Salary | int | Salary a Receptionist gets |
|  |  |  |

**Admission table**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Data type | Comments | |
|  |  |  |  |
| Admsn\_id | int | Unique id for | an |
|  |  | Admission |  |
|  |  |  |  |
| Pat\_id | int | Unique id for | a Patient |
|  |  |  |  |
| Rcp\_id | int | Unique id for | a |
|  |  | Receptionist |  |
|  |  |  | |
| Date | varchar(20) | Date of Admission | |
|  |  |  |  |

**Doctor table:**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Data type |  | Comments |
|  |  |  | |
| Doc\_id | int | Unique id for a Doctor | |
|  |  |  |  |
| Doc\_name | varchar(20) | Doctor’s | name |
|  |  |  | |
| Doc\_type | varchar(20) | Doctor’s specialty | |
|  |  |  |  |
| Doc\_Age | int | Doctor’s | age |
|  |  |  |  |
| Doc\_Address | varchar(20) | Doctor’s | address |
|  |  |  | |
| Doc\_Mob | Bigint | Mobile Number | |
|  |  |  | |
| Designation | varchar(20) | Doctor’s designation | |
|  |  |  | |
| Passed\_from | varchar(20) | Doctor is passed from which | |
|  |  | medical college | |
|  |  |  | |
| Doc\_Salary | int | Salary of a doctor | |
|  |  |  |  |

**Appointment table**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Data type | Comments | |
|  |  |  |  |
| Apnmt\_id | int | Unique id for | an Appointment |
|  |  |  |  |
| Pat\_name | varchar(20) | Patient name |  |
|  |  |  |  |
| Doc\_id | int | Unique id for | a Doctor |
|  |  |  |  |
| Rcp\_id | int | Unique id for | a Receptionist |
|  |  |  | |
| Apnmt\_date | varchar(20) | Date of an Appointment | |
|  |  |  |  |

**Bill table**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Comments |
|  |  |  |
| Bill\_id | int | Unique id for a Bill |
|  |  |  |
| Bill\_for | varchar(20) | Purpose of the bill |
|  |  |  |
| Bill\_type | varchar(20) | Bill either in Cash or |
|  |  | Check |
|  |  |  |
| Bill\_total | int | Total amount |
|  |  |  |

**Payment table**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Data type | Comments | |
|  |  |  |  |
| Pay\_id | int | Unique id for | a Payment |
|  |  |  |  |
| Bill\_for | varchar(20) | Type of Bill |  |
|  |  |  |  |
| Pat\_id | int | Unique id for | a Patient |
|  |  |  |  |
| Pay\_type | varchar(20) | Payment in Cash or Check | |
|  |  |  | |
| Pay\_date | varchar(20) | Date of Payment | |
|  |  |  |  |

**Medicine table**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Comments |
|  |  |  |
| Mdcn\_id | int | Unique id for a Medicine |
|  |  |  |
| Mdcn\_name | varchar(20) | Medicine’s Name |
|  |  |  |
| company | varchar(20) | Medicine’s Company |
|  |  |  |
| M\_date | varchar(20) | Manufacture Date |
|  |  |  |
| E\_date | varchar(20) | Expire Date |
|  |  |  |
| price | int | Price of the Medicine |
|  |  |  |

**Prescription table**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Data type | Comments | |
|  |  |  |  |
| Prs\_id | int | Unique id for | a |
|  |  | Prescription |  |
|  |  |  |  |
| Doc\_id | int | Unique id for | a Doctor |
|  |  |  |  |
| Mdcn\_id | int | Unique id for | a Medicine |
|  |  |  |  |
| Pat\_id | int | Unique id for | a Patient |
|  |  |  | |
| Date | varchar(20) | Date of the Prescription | |
|  |  |  | |
| Time | varchar(20) | Time of the Prescription | |
|  |  |  | |
| Fee | varchar(20) | Prescription Fees | |
|  |  |  |  |

**Department table**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Data type | Comments | |
|  |  |  |  |
| Dept\_id | int | Unique id for | a |
|  |  | Department |  |
|  |  |  |  |
| Dept\_name | varchar(20) | Department’s | name |
|  |  |  | |
| treatement | varchar(20) | Treatments of a patient | |
|  |  | conducted in a Department | |
|  |  |  |  |

**Doctor\_from\_Department table**

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Data type | Comments | |
|  |  |  |  |
| Dfd\_id | int | Unique id for | a |
|  |  | DoctorsfromDepartment junction | |
|  |  | table |  |
|  |  |  |  |
| Doc\_id | int | Unique id for | a Doctor |
|  |  |  |  |
| Dept\_id | int | Unique id for | a Department |
|  |  |  |  |
|  |  |  |  |

**Primary Keys and Foreign Keys:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **Table** | **Primary Key** | **Foreign Keys** | |
| **Column** | **References** |
| 1 | Patient | Pat\_id | - | - |
| 2 | Appointment | Ap\_id | Doc\_id | Doctor.Doc\_id |
| Rcp\_id | Receptionist.Rcp\_id |
| 3 | Doctor | Doc\_id | - | - |
| 4 | Bill | Bill\_id | - | - |
| 5 | Department | Dept\_id | - | - |
| 6 | Admission | Admsn\_id | Pat\_id | Patient. Pat\_id |
| Rcp\_id | Receptionist.Rcp\_id |
| 7 | Receptionist | Rcp\_id | - | - |
| 8 | Medicine | Mdcn\_id | - | - |
| 9 | Payment | Pay\_id | Bill\_for | Bill. Bill\_for |
| Pat\_id | Patient. Pat\_id |
| 10 | Prescription | Prs\_id,  Mdcn\_id | Doc\_id | Doctor.Doc\_id |
| Pat\_id | Patient. Pat\_id |
| 11 | Doctor\_from\_Department | Dfd\_id | Doc\_id | Doctor.Doc\_id |
| Dept\_id | Department.Dept\_id |

# 

# Cardinality

Cardinality notations:

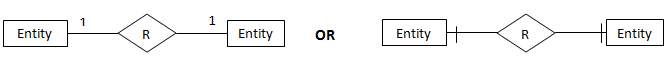
M,N = many

|  |  |  |  |
| --- | --- | --- | --- |
| **SN** | **Relationship** | **Entities** | **Cardinality** |
| 1. | Doctor\_From\_Department | Doctor & Departments | N : 1 |
| 2. | Admission | Patient & Receptionist | N : 1 |
| 3. | Payment | Patient & Bill | 1 : M |
| 4. | Prescription | Patient & Doctor & Medicine | 1 : 1 : M |
| 5. | Appointment | Doctor & Receptionist | N : M |

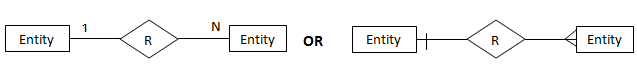
**ER Diagram**

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Shape Name** | **Symbol Description** |
| **Entities** | | |
|  | Entity | An entity is represented by a rectangle which contains the entity’s name. |
|  | Weak Entity | An entity that cannot be uniquely identified by its attributes alone. The existence of a weak entity is dependent upon another entity called the owner entity. The weak entity’s identifier is a combination of the identifier of the owner entity and the partial key of the weak entity. |
|  | Associative Entity | An entity used in a many-to-many relationship (represents an extra table). All relationships for the associative entity should be many |
| **Attributes** | | |
|  | Attribute | In the Chen notation, each attribute is represented by an oval containing atributte’s name |
|  | Key attribute | An attribute that uniquely identifies a particular entity. The name of a key attribute is underscored. |
|  | Multivalued attribute | An attribute that can have many values (there are many distinct values entered for it in the same column of the table). Multivalued attribute is depicted by a dual oval. |
|  | Derived attribute | An attribute whose value is calculated (derived) from other attributes. The derived attribute may or may not be physically stored in the database. In the Chen notation, this attribute is represented by dashed oval. |
| **Relationships** | | |
|  | Strong relationship | A relationship where entity is existence-independent of other entities, and PK of Child doesn’t contain PK component of Parent Entity. A strong relationship is represented by a single rhombus |
|  | Weak (identifying) relationship | A relationship where Child entity is existence-dependent on parent, and PK of Child Entity contains PK component of Parent Entity. This relationship is represented by a double rhombus. |

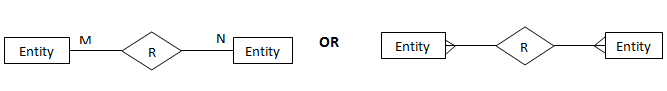
**One-to-one relation: -** A one-to-one relationship is represented by adding ‘1’ near the entities on the line joining the relation. In another type of notation one dash is added to the relationship line at both ends.



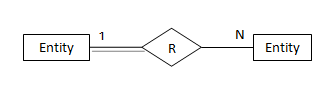
**One-to-Many relation**: A one-to-many relationship is represented by adding ‘1’ near the entity at left hand side of relation and ‘N’ is written near the entity at right side. Other type of notation will have dash at LHS of relation and three arrow kind of lines at the RHS of relation as shown below.



**Many-to-Many relation**: A one-to-many relationship is represented by adding ‘M’ near the entity at left hand side of relation and ‘N’ is written near the entity at right side. Other type of notation will have three arrow kinds of lines at both sides of relation as shown below.



**Participation Constraints**: Total participation constraints are shown by double lines and partial participations are shown as single line.



**Relational Database Design**

Relational databases are the most commonly used database today. It uses the table to structure information so that it can be readily and easily searched through.

To make a Relational database design we have to be clear about two parts:

1. Functional Dependency
2. Normalization

**FULFILMENT OF NORMAL FORMS:**

**Medicine Table**:-

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mdcn\_id | Mdcn\_name | Company | m\_date | e\_date | price |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| { Mdcn\_id } => { Mdcn\_name } | Functional Dependency Exist |
| 2 different Mdcn\_name do not correspond to the same Mdcn\_id | |
| { Mdcn\_id } => { Company } | Functional Dependency Exist |
| 2 different medicine Company do not correspond to the same Mdcn\_id | |
| { Mdcn\_id => { price } | Functional Dependency Exist |

2 different medicine price do not correspond to the same Mdcn\_id

Relation :( Mdcn\_id, Mdcn\_name, Company, m\_date, e\_date, price)

Full Functional Dependencies: {Mdcn\_id } => { Mdcn\_name }

{Mdcn\_id } => { Company }

{Mdcn\_id} => { price }

1NF:-

Attributes do not have sub attributes. So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key.

So the relation is in 2NF. 3NF:-

No chain Exists.

So the relation is in 3NF.

BCNF:-

No part of the primary key is Fully Functional Dependent on the non-primary keys. So the relation is in BCNF.

**Bill Table**:-

|  |  |  |  |
| --- | --- | --- | --- |
| Bill\_id | Bill\_for | Bill\_type | Bill\_ total |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
| {Bill\_id} => {Bill\_for} | Functional Dependency Exist. |
| 2 different Bill\_for’s do not correspond to the same Bill\_id. | |
| {Bill \_id} => {Bill\_type} | Functional Dependency Exist. |
| 2 different Bill\_type do not correspond to the same Bill\_id. | |
| {Bill \_id} => {Bill total | Functional Dependency Exist. |

2 different Bill total do not correspond to the same Bill\_id. Relation :( Bill\_id, Bill\_for, Bill total, Bill\_type)

Full Functional Dependency: {Bill\_id} => {Bill\_for} {Bill\_id} => {Bill\_type}

{Bill \_id} => {Bill total}

1NF:-

Attributes do not have sub attributes.

So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key.

So the relation is in 2NF

3NF:-

No chain Exists.

So the relation is in 3NF.

BCNF:-

No part of the primary key is Fully Functional Dependent on the non-primary key. So the relation is in BCNF.

In a similar way Bill, Doctor, Receptionist, Medicine, Department fulfil all the normal forms.

**JUNCTION TABLES:**

**Admission Room Table:-**

This is a junction table between Patient and Receptionist Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Admsn\_id | Pat\_id | Rcp\_id | Date | Time |
|  |  |  |  |  |

Full Functional Dependencies:

|  |  |
| --- | --- |
| {admsn\_id} => {Rcp\_id} | Functional Dependency Exist |
| {adsn\_id} => {Date} | Functional Dependency Exist |

|  |  |
| --- | --- |
| {admsn\_id} => {Time} | Functional Dependency Exist |
| {admsn\_id} => {Pat\_id} | Functional Dependency Exist |
| 1NF:- |  |

Attributes do not have sub attributes.

So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key. So the relation is in 2NF.

3NF:-

No chain Exists.

So the relation is in 3NF. BCNF:-

No part of the primary key is Fully Functional Dependent on the non-primary keys. So the relation is in BCNF.

In a similar way Appointment Tables fulfil all the normal forms.

**Prescription Table:-**

This is a junction table between Patient, Medicine & Doctor Table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Prs\_id | Doc\_id | Mdcn\_id | Pat\_id | Date | Fees | Time |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| Full Functional Dependencies: |  |
| {Prs\_id, Mdcn\_id}=> {Doc\_id} | Functional Dependency Exist |
| {Prs\_id, Mdcn\_id}=> {Pat\_id} | Functional Dependency Exist |
| {Prs\_id, Mdcn\_id}=> {Date, Fees, Time} | Functional Dependency Exist |

Relation: (Prs\_id, Mdcn\_id, Doc\_id, Pat\_id, Date, Fees, Time)

{Prs\_id, Mdcn\_id}=> {Doc\_id}

{Prs\_id, Mdcn\_id}=> {Pat\_id}

{Prs\_id, Mdcn\_id}=> {Date}

{Prs\_id, Mdcn\_id}=> {Time}

{Prs\_id, Mdcn\_id}=> {Fees}

1NF:-

Attributes do not have sub attributes. So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key. So the relation is in 2NF.

3NF:-

No chain Exists.

So the relation is in 3NF. BCNF:-

No part of the primary key is Fully Functional Dependent on the non-primary keys. So the relation is in BCNF.

**Patient Table :**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pat\_id | Pat\_name | Pat\_Age | Pat\_Sex | Pat\_Address | Pat\_DOB | Pat\_MOB |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

{ Pat\_id } => { Pat\_name }   Functional Dependency Exist

2 different patient names do not correspond to the same patient id.

{ Pat\_id } => { Age }   Functional Dependency Exist

2 different ages do not correspond to the same patient id.

{ Pat\_id } => { Sex }   Functional Dependency Exist

2 different sexes do not correspond to the same patient id.

{ Pat\_id } => { Address }   Functional Dependency Exist

2 different addresses do not correspond to the same patient id.

{ Pat\_id } => { DOB }   Functional Dependency Exist

2 different date of birth do not correspond to the same patient id.

{ Pat\_id } => { MOB }   Functional Dependency Exist

2 different mobile numbers do not correspond to the same patient id.

1NF:-

Attributes do not have sub attributes. So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key. So the relation is in 2NF.

3NF:-

No chain Exists.

So the relation is in 3NF. BCNF:-

No part of the primary key is Fully Functional Dependent on the non-primary keys. So the relation is in BCNF.

**Receptionist Table :**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rcp\_id | Rcp\_name | Rcp\_Age | Rcp\_Address | Rcp\_MOB | shifting | Rcp\_salary |
|  |  |  |  |  |  |  |

{ Rec\_id } => { Rec\_name }   Functional Dependency Exist

2 different receptionist names do not correspond to the same receptionist id.

{ Rec\_id } => { Age }   Functional Dependency Exist

2 different ages do not correspond to the same receptionist id.

{ Rec\_id } => { Shifting }   Functional Dependency Exist

2 different shifts do not correspond to the same receptionist id.

{ Rec\_id } => { Salary }   Functional Dependency Exist

2 different salaries do not correspond to the same receptionist id.

{ Rec\_id } => { MOB }   Functional Dependency Exist

2 different mobile numbers do not correspond to the same receptionist id.

{ Rec\_id } => { Address }   Functional Dependency Exist

2 different addresses do not correspond to the same receptionist id.

1NF:-

Attributes do not have sub attributes. So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key. So the relation is in 2NF.

3NF:-

No chain Exists.

So the relation is in 3NF. BCNF:-

No part of the primary key is Fully Functional Dependent on the non-primary keys. So the relation is in BCNF.

**Doctor Table :**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Doc\_id | Doc\_name | Doc\_type | Doc\_age | Doc\_address | Doc\_MOB | Passed\_from | Doc\_salary |
|  |  |  |  |  |  |  |  |

{ Doc\_id } => { Doc\_name }   Functional Dependency Exist

2 different Doctor names do not correspond to the same Doctor id.

{ Doc\_id } => { Doc\_type }   Functional Dependency Exist

2 different Doctor types do not correspond to the same Doctor id.

{ Doc\_id } => { Designation }   Functional Dependency Exist

2 different Doctor designations do not correspond to the same Doctor id.

{ Doc\_id } => { Age }   Functional Dependency Exist

2 different ages do not correspond to the same Doctor id.

{ Doc\_id } => { Address }   Functional Dependency Exist

2 different addresses do not correspond to the same Doctor id.

{ Doc\_id } => { MOB }   Functional Dependency Exist

2 different mobile numbers do not correspond to the same Doctor id.

{ Doc\_id } => { Passed \_from }   Functional Dependency Exist

2 different Passed from  do not correspond to the same Doctor id.

{ Doc\_id } => { Salary }   Functional Dependency Exist

2 different salaries do not correspond to the same Doctor id.

1NF:-

Attributes do not have sub attributes. So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key. So the relation is in 2NF.

3NF:-

No chain Exists.

So the relation is in 3NF. BCNF:-

No part of the primary key is Fully Functional Dependent on the non-primary keys. So the relation is in BCNF.

**Appointment Table :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ap\_id | Pat\_name | Doc\_id | Rcp\_id | apnmt\_date | apnmt\_time |
|  |  |  |  |  |  |

{ Apnmt\_id } => { Pat\_id }   Functional Dependency Exist

2 different patient id’s do not correspond to the same appointment id.

{ Apnmt\_id } => { Doc\_id }   Functional Dependency Exist

2 different doctor id’s do not correspond to the same appointment id.

{ Apnmt\_id } => { Rcp\_id }   Functional Dependency Exist

2 different receptionist id’s do not correspond to the same appointment id.

{ Apnmt\_id } => { Apnmt\_date }   Functional Dependency Exist

2 different appointment id’s do not correspond to the same appointment id.

1NF:-

Attributes do not have sub attributes. So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key. So the relation is in 2NF.

3NF:-

No chain Exists.

So the relation is in 3NF. BCNF:-

No part of the primary key is Fully Functional Dependent on the non-primary keys. So the relation is in BCNF.

**Payment Table :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pay\_id | Bill\_for | Pat\_id | Pay\_type | Pay\_date |
|  |  |  |  |  |

{ Pay\_id } => { Bill\_id }   Functional Dependency Exist

2 different bill id’s do not correspond to the same payment id.

{ Pay\_id } => { Pat\_id }   Functional Dependency Exist

2 different patient  id’s do not correspond to the same payment id.

{ Pay\_id } => { Acct\_id }   Functional Dependency Exist

2 different accountant id’s do not correspond to the same payment id.

{ Pay\_id } => { Pay\_type }   Functional Dependency Exist

2 different Pay types  do not correspond to the same payment id.

{ Pay\_id } => { Pay\_date }   Functional Dependency Exist

2 different pay dates  do not correspond to the same payment id.

1NF:-

Attributes do not have sub attributes. So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key. So the relation is in 2NF.

3NF:-

No chain Exists.

So the relation is in 3NF. BCNF:-

No part of the primary key is Fully Functional Dependent on the non-primary keys. So the relation is in BCNF.

**Department Table :**

Dept\_id Dept\_name treatment

{ Dept\_id } => { Dept\_name }   Functional Dependency Exist

2 different department names do not correspond to the same department id.

{ Dept\_id } => { Treatment }   Functional Dependency Exist

2 different treatments  do not correspond to the same department id.

1NF:-

Attributes do not have sub attributes. So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key. So the relation is in 2NF.

3NF:-

No chain Exists.

So the relation is in 3NF. BCNF:-

No part of the primary key is Fully Functional Dependent on the non-primary keys. So the relation is in BCNF.

**Doctor\_from\_department Table :**

Dfd\_id Doc\_id Dept\_id

{ Dfd\_id } => { Doc\_id }   Functional Dependency Exist

2 different doctor id’s do not correspond to the same doctor from department id.

{ Dfd\_id } => { Dept\_id }   Functional Dependency Exist

2 different department id’s do not correspond to the same doctor from department id.

1NF:-

Attributes do not have sub attributes. So the relation is in 1NF.

2NF:-

Every non primary key is Fully Functional Dependent on the primary key. So the relation is in 2NF.

3NF:-

No chain Exists.

So the relation is in 3NF. BCNF:-

No part of the primary key is Fully Functional Dependent on the non-primary keys. So the relation is in BCNF.

**Implementation in SQL Server:**

After Normalization, we implemented our Database in SQL Server. There were 11 tables and each of them was connected accurately in the SQL Server’s Entity Relationship Diagram. Then we entered the data in the corresponding database tables.

**CREATE COMMANDS:**

**Patient table** :

CREATE TABLE Patient(

Pat\_id int ,

Pat\_name varchar (20) ,

Pat\_ Age int ,

Pat\_Sex varchar(20) ,

Pat\_Address varchar(20) ,

Pat\_Dob varchar(20) ,

Pat\_Mob Bigint ,

PRIMARY KEY (Pat\_id),

);

**Receptionist table** :

CREATE TABLE Receptionist(

Rcp\_id int ,

Rcp\_name varchar (20) ,

Rcp\_Age int ,

Rcp\_ Address varchar(20) ,

Rcp\_Mob Bigint,

Shifting varchar(20),

Rcp\_Salary int

PRIMARY KEY (Rcp\_id),

);

**Admission table:**

CREATE TABLE Admission(

Admsn\_id int ,

Pat\_id int ,

Room\_id INT ,

Rcp\_id int ,

Date varchar(20) ,

PRIMARY KEY (Admsn\_id),

FOREIGN KEY(Pat\_id) REFERENCES Patient(Pat\_id),

FOREIGN KEY(Rcp\_id) REFERENCES Receptionist(Rcp\_id),

);

**Doctor table:**

CREATE TABLE Doctor(

Doc\_id int ,

Doc\_name varchar (20) ,

Doc\_type varchar(20) ,

Doc\_Age int ,

Doc\_Address varchar(20) ,

Doc\_Mob Bigint ,

Designation varchar(20) ,

Passed\_from varchar(20) ,

Doc\_Salary int,

PRIMARY KEY (Doc\_id)

);

**Appointment table:**

CREATE TABLE Appointment(

Apnmt\_id int ,

Pat\_id int ,

Doc\_id int ,

Rcp\_id int ,

Apnmt\_date varchar(20) ,

PRIMARY KEY (Apnmt\_id),

FOREIGN KEY(Pat\_id) REFERENCES Patient(Pat\_id),

FOREIGN KEY(Doc\_id) REFERENCES Doctor(Doc\_id),

FOREIGN KEY(Rep\_id) REFERENCES Receptionist(Rep\_id),

);

**Bill table:**

CREATE TABLE Bill(

Bill\_id int ,

Bill\_for varchar (20) ,

Bill\_type varchar(20) , ,

Bill\_total int ,

PRIMARY KEY (Bill\_id)

);

**Payment table:**

CREATE TABLE Payment(

Pay\_id int ,

Bill\_id int ,

Pat\_id int ,

Pay\_type varchar(20) ,

Pay\_date varchar(20) ,

PRIMARY KEY (Pay\_id),

FOREIGN KEY(Pat\_id) REFERENCES Patient(Pat\_id),

FOREIGN KEY(Bill\_for) REFERENCES Bill(Bill\_for),

);

**Medicine table:**

CREATE TABLE Medicine(

Mdcn\_id int ,

Mdcn\_name varchar (20) ,

company varchar(20) ,

M\_date varchar(20) ,

E-date varchar(20) ,

price int ,

PRIMARY KEY (Mdcn\_id)

);

**Prescription table**:

CREATE TABLE Prescription(

Prs\_id int ,

Doc\_id int ,

Mdcn\_id int ,

Pat\_id int ,

Date varchar(20) ,

Time varchar(20) ,

Fee varchar(20) ,

PRIMARY KEY (Prs\_id, Mdcn\_id),

FOREIGN KEY(Pat\_id) REFERENCES Patient(Pat\_id),

FOREIGN KEY(Doc\_id) REFERENCES Doctor(Doc\_id),

);

**Department table:**

CREATE TABLE Department(

Dept\_id int ,

Dept\_name varchar (20) ,

treatment varchar(20) ,

PRIMARY KEY (Dept\_id)

);

**Doctor\_from\_Department table:**

CREATE TABLE Doctor\_from\_Department(

Dfd\_id int ,

Doc\_id int ,

Dept\_id int ,

PRIMARY KEY (Dfd\_id),

FOREIGN KEY(Doc\_id) REFERENCES Doctor(Doc\_id),

FOREIGN KEY(Dept\_id) REFERENCES Department(Dept\_id),

);

**INSERT STATEMENTS:**

**Admission table:**

INSERT INTO `admission`(`Admsn\_id`, `Pat\_id`, `Rcp\_id`, `date`, `time`) VALUES (1,1,1,” 10/1/2017”,” 15:50”);

**Appointment table:**

INSERT INTO `appointment`(`Ap\_id`, `Doc\_id`, `Rcp\_id`, `apnmt\_date`, `apnmt\_time`, `Pat\_name`) VALUES (1,1,1, “10/1/2017”,”10:30”,”akshay”);

**Bill table:**

INSERT INTO `bill`(`Bill\_id`, `Bill\_for`, `Bill\_type`, `Bill\_total`) VALUES (1,”Doctor Fee”,”Cash”,500);

**Department table:**

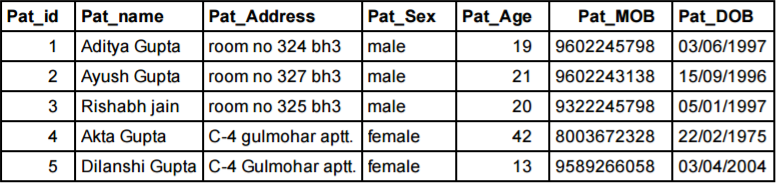
INSERT INTO `department`(`Dept\_id`, `Dept\_name`, `treatment`) VALUES (1,”Orthopedics”,”bones”);

**Prescription table:**

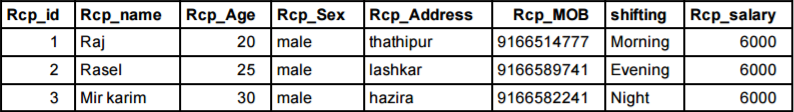
INSERT INTO `prescription`(`Prs\_id`, `Doc\_id`, `Mdcn\_id`, `Pat\_id`, `date`, `Fee`, `time`) VALUES (1,1,1,1,” 12/05/2017”,500,”8:15”);

**Sample Data values of Tables**

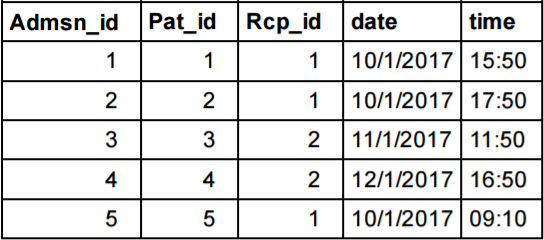
**Patient table**



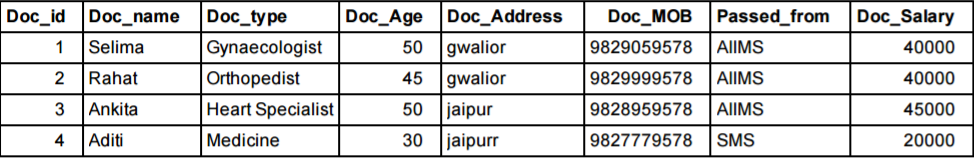
**Receptionist table**



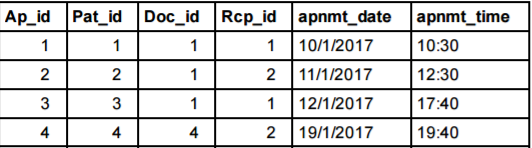
**Admission room table**



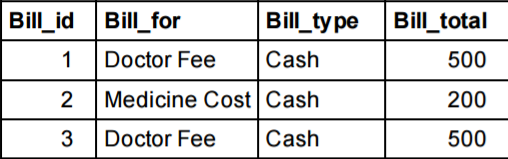
**Doctor table**



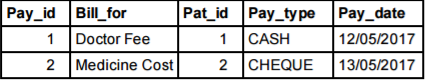
**Appointment table**



**Bill table**



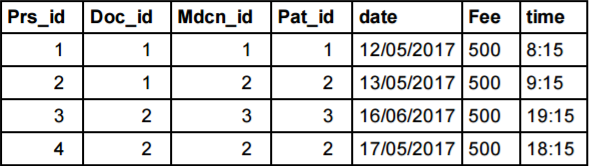
**Payment Table**



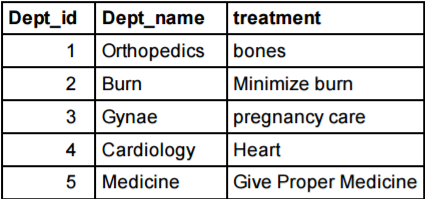
**Medicine table**



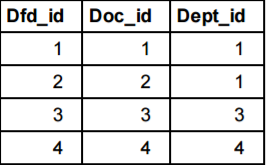
**Prescription table**



**Department table**



**Doctor\_form\_department table**



**Queries**

After completing the implementation we retrieved different information from the system by joining 2 or more tables of the system. Sample Examples are given below:

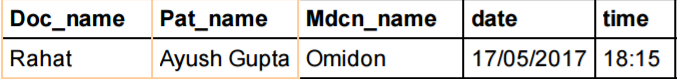
**Question 1**

Which medicine is prescribed by doctor Rahat to Patient Ayush Gupta and when ?

SQL:

select doctor.Doc\_name,patient.Pat\_name,medicine.Mdcn\_name ,prescription.date,prescription.time from prescription,doctor,patient,medicine where prescription.Doc\_id = doctor.Doc\_id and prescription.Pat\_id = patient.Pat\_id and medicine.Mdcn\_id = prescription.Mdcn\_id and doctor.Doc\_name = "Rahat" and patient.Pat\_name = "Ayush Gupta"

Output:



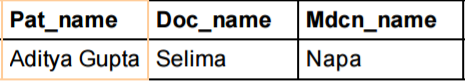
**Question 2**

Which doctors prescribed which medicine to patient Aditya Gupta?

SQL:

select Pat\_name, Doc\_name, Mdcn\_name from Patient, Doctor,Medicine, Prescription where Pat\_name = 'Aditya Gupta' and Patient.Pat\_id = Prescription.Pat\_id and Doctor.Doc\_id = Prescription.Doc\_id and Medicine.Mdcn\_id = Prescription.Mdcn\_id;

Output 2:



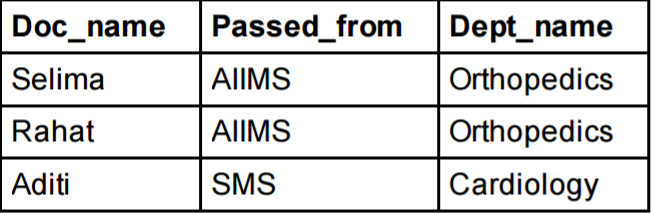
**Question 3:**

Which Doctors are from which Department and they passed from which college and got salaries below 45000 Rs?

SQL

select Doc\_name,Passed\_from,Dept\_name from Doctor, Department, doctor\_from\_department where Doc\_Salary <45000 and Doctor.Doc\_id = doctor\_from\_department.Doc\_id and Department.Dept\_id = doctor\_from\_department. Dept\_id

Output 3:



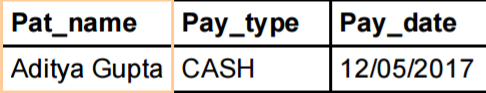
**Question-4**

On which date Patient Aditya gupta paid the Doctor Fee and also find the mode of payment?

SQL:

select patient.Pat\_name ,payment.Pay\_type,payment.Pay\_date from payment , patient where payment.Pat\_id = patient.Pat\_id and patient.Pat\_name="Aditya Gupta"

Output 4:-



**Question 5:**

In which time receptionist Raj appointed patients to Doctor Selima?

SQL:

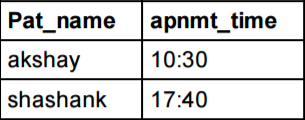
Select Appointment.Pat\_name,apnmt\_time from Appointment ,doctor, receptionist

WHERE

appointment.Doc\_id = doctor.Doc\_id and appointment.Rcp\_id = receptionist.Rcp\_id

and doctor.Doc\_name = "Selima" and receptionist.Rcp\_name = "Raj"

Output 5:



**Implementation on a Website**

The website has been made including the basic functionalities in a Hospital Management System. The database for the website has been made on MYSQL (PhpMyAdmin) . First the receptionist will login for the website using his/her login id and password.

The main page of the website includes a user interface which includes navigation items like Patients, Doctors, Appointments, Departments, Contact Us and the logout button.

Under patients we have a page in which we can search for patient details using the patient id and add a new patient to the hospital.

Under doctors we have a page in which we can view all the departments in a particular department in the hospital and add details for a new doctor of the hospital.

Under appointments we have a page in which we can search for appointment details of a patient using the appointment id , add new appointment and delete appointment details using the appointment id.

Under departments and contact us we can view the departments in the hospital and the contact details of the hospital.

The front end has been made using HTML, CSS , Bootstrap and javascript.

The backend has been done using PHP and the SQL queries has been written in the PHP script to execute the differ functions of adding and searching for patients and doctors .

**CONCLUSION:**

Since we are entering details of the patients electronically in the” Hospital Management System”, data will be secured. Using this application we can retrieve patient’s details with a single click. Thus processing information will be faster. It guarantees accurate maintenance of Patient details. It easily reduces the book keeping task and thus reduces the human effort and increases accuracy speed.