B.Sc Engineering Thesis Paper

On

“Designing and Interfacing a Hospital-Based Database System”

(A Case Study of BIRDEM)

Department of Computer Science & Engineering

Ahsanullah University of Science & Technology

Dhaka, Bangladesh.

A thesis paper submitted in partial fulfillment of the requirements for the Degree of B.Sc Engineering (Computer Science & Engineering)

Date: - July 12, 2010

Session: - Spring ‘10
“Designing and Interfacing a Hospital-Based Database System”

(A Case Study of BIRDEM)

Submitted By:-

1. Syed Mahboob Nur 06.02.04.013
2. Jahid Hasan 06.02.04.036
3. Kazi Sumaiya 06.02.04.042
4. Tasfia Rahman 06.02.04.044

In Partial Fulfillment for the Degree of
B.Sc Engineering in Computer Science & Engineering
Ahsanullah University of Science & Technology.
Certification

We hereby, proclaim that the thesis on “Designing and Interfacing a Hospital-Based Database System (A Case Study of BIRDEM)” was conducted under the supervision of Ms. Rosina Surovi Khan.

We also declare that neither this nor any part thereof has been submitted elsewhere for the award of any degree.

Approved By:       Submitted By:

-------------------      -------------------

Dr. S. M. Abdullah Al-Mamun              Syed Mahboob Nur
Professor & Head of the Department
Dept of C.S.E, AUST

Supervised By:

-------------------

Ms. Rosina Surovi Khan
Assistant Professor
Dept of C.S.E, AUST

TASFIA RAHMAN
CONTENTS AT A GLANCE

PREFACE

ABSTRACT

1. Introduction

2. Designing the Database System

3. Interfacing the Database System using .NET framework

4. Conclusion and Future work
Acknowledgement

Starting by the name of Almighty Allah......

Authors would like to express their sincere and hearty gratitude and profound indebtedness to their respectful teacher Ms Rosina Surovi Khan, Assistant Professor, AUST, for her constant timely and appropriate guidance, helpful advice, invaluable assistance and endless patience throughout the progress of their work, without which the work could not have been completed.

Authors also acknowledge with hearty thanks to all the members of the BIRDEM hospital for their important information and cooperation.

Finally, authors acknowledge all cooperation of their friends, who helped them through giving their important time, their knowledge and their best advice.

Special thanks to our parents and elders for their help and support.
# Table of Contents

**PREFACE** ......................................................................................................................... ix  

**ABSTRACT** ......................................................................................................................... x  

1. INTRODUCTION ...................................................................................................................... 1  

2. DESIGNING THE DATABASE SYSTEM ...... 4  

   2.1 Determining Entities and Attributes................................. 4  
   2.2 Entity Relationship Diagram .............................................. 7  
   2.3 Relational Model ................................................................. 9  
      2.3.1 Relational Tables' Descriptions .............. 13  
      2.3.2 Explanation of Relational Model ......... 26  
   2.4 Relational Database Design .............................................. 34  
      2.4.1 Functional Dependency ......................... 34  
      2.4.2 Normalization ........................................ 35
2.5 Implementation in SQL Server ........................................49
   2.5.1 Creation of Tables and Insertion of data ..................51
   2.5.2 Sample Data Values of Tables ..................................53
2.6 Complex Queries ..........................................................60

3. INTERFACING THE DATABASE SYSTEM USING .NET FRAMEWORK ..................64

3.1 Research on Interface Design Guidelines ..........................64
3.2 FRONT END Design .........................................................74
   3.2.1 Forms' Design .......................................................75
   3.2.2 Relating Interface Design Guidelines to Front End Design ..........93

3.3 Security feature of FRONT END ......................................96
3.4 Implementation of Insert, Delete, Update buttons & Search Option .........................................................107
3.5 Usage of DLL file ..........................................................113
4. CONCLUSION & FUTURE WORK ..........115

4.1 Conclusion .................................................................115
4.2 Future Work ............................................................115

4.2.1 Gridline View Features ................................. 116
4.2.2 Trigger Features ..............................................129

REFERENCES.................................................................133

APPENDIX ................................................................. 135
Our thesis is about Designing and Interfacing a Hospital-Based Database System. It forms a basic entity of the management of a Hospital. Hence, it is very important for the system to be reliable, user friendly, and should be properly functional for a long time without cropping up of any errors.

To start with the system study we visited Bangladesh Institute of Research and Rehabilitation for Diabetes, Endocrine and Metabolic Disorders (BIRDEM). We saw their system, studied it and tried to develop a better system. Our system is an automated system for Hospital Management. This gave us the idea of the different fields that ought to be in a Hospital Management System such as patient registration, his/her advance payment, the records, the details etc. and also how a software system can make the work easy both for the hospital staff and the patients. Moreover, the evaluation helped us to arrive at the conclusion that the automated software is far more superior to the manual ones.
ABSTRACT

Our motive is to develop a software that is very much user friendly and easy to gather information in a very short time. We try to make our software reliable and comfortable.

As our thesis paper is on Designing and Interfacing a Hospital Management System (A Case Study of BIRDEM) we divide our work into two basic parts Designing part and Interfacing Part.

© We give a flow chart on our work division in THESIS OVERVIEW part.

Chapter 1 → Introduction

In this chapter we discuss the definition of Database and its usefulness. We also describe the reason to take HOSPITAL MANAGEMENT SYSTEM as our thesis work.

Chapter 2 → Designing the Database System

In this chapter we describe the entities and attributes. We draw the Entity Relationship Diagram (ERD) and Tables. We determine the attributes of tables and its data types. We also find functional dependencies and normalize all the tables. Then we implement our database in SQL Server and finally we execute some complex queries on the system.

Chapter 3 → Interfacing the Database System using .Net Framework.

We made a research on Interface Design Guidelines and designed our front end in C#. We applied some of the guidelines in our front end.

We control our software security using C#. We Insert Delete, Update and Search data from the database in our software. We used a DLL file so that we
can easily access to any Operating System and we don’t need to load our database.

Chapter 4  \rightarrow Conclusion and Future Work.

We tried to Save, Delete and Update data using Data Grid view and we also tried to use Trigger in SQL Server but we cannot complete them. So we include it as a part of future work.
INTRODUCTION

❖ What is a Database?

➢ A Database is a collection of records which are stored on a computer; a database organizes the data according to database models such as a relational model. [1]

❖ Why do we need Databases?

➢ Databases collect items on which the user can carry out various operations such as viewing, navigating, creating tables, and searching. Databases can be seen as a symbolic form of the computer age. [2]

We use databases for these reasons. Such as,

1. We use database because we can easily manipulate, edit or delete data.
2. Data are kept organized in a database so we can easily retrieve data.
3. Easy to find out desired data.
4. Data are secured.

❖ Advantages of Database

✓ Reduced Data Redundancy.
✓ Reduced updating errors and increased consistency.
✓ Greater data integrity and independence from applications programs.
Improved data access to users through use of host and query languages.

Improved data security.

Reduced data entry, storage, and retrieval costs.

Facilitated development of new application programs. [3]

In our thesis *Designing and Interfacing a Hospital-Based Database System (A case study of BIRDEM)* we can see two basic parts.

- Designing &
- Interfacing

Our Thesis Teacher Ms. Rosina Surovi Khan decided that we have to complete the design part in semester 4/1 and interfacing part in semester 4/2. In the introductory class of the thesis our respected madam suggested to select a specific database system to work on.

**# Choosing Hospital Management System for our thesis**

We study and select three systems at first. The systems were

- Banking System
- Computer Sales Management System
- Hospital Management System

We saw the demos of the respective systems from different sources and all the group members decided to do the thesis on Hospital Management System (A Case Study of BIRDEM) because the system is less complex and easy to study. Most Banking Systems and Computer Sales Management Systems are controlled using online based software where users can access from any part of the country. But we are determined to make desktop based software. So we decided
to choose Hospital Management System based on a Case Study of BIRDEM. We try our best to make the system efficient and user friendly with the help of our database and front end software.

# Thesis Overview

**DATABASE**

**DESIGNING**
- Determining Entities and Attributes
- Entity Relationship Diagram
- Relational Model
- Normalization
- Implementation in SQL Server
- Complex Queries

**INTERFACING**
- Research on Interface Design Guidelines
- Front End Design
- Security feature of Front End
- Implementation
  - (Insert, Delete, Update Buttons and Search Option)
- Usage of DLL file
DESIGNING THE DATABASE SYSTEM

2.1 Determining Entities and Attributes

❖ Entity

✓ An *entity* is something that has a distinct, separate existence, though it need not be a material existence. In particular, abstractions and legal fictions are usually regarded as entities. In general, there is also no presumption that an entity is animate. Entities are used in system developmental models that display communications and internal processing of, say, documents compared to order processing.

✓ An entity could be viewed as a set containing subsets.

✓ A DBMS *entity* is either a thing in the modeled world or a drawing element in an Entity Relationship Diagram (ERD). [4]
**Attribute**

✓ An attribute is a specification that defines a property of an object, element, or file. It may also refer to or set the specific value for a given instance of such.

✓ Attributes should more correctly be considered metadata. It is frequently and generally a property of an entity.

✓ An attribute of an object usually consists of a name and a value; of an element, a type or class name; of a file, a name and extension.[5]

**Data Type**

✓ A data type (or datatype): In programming, a classification identifying one of various types of data, as floating-point, integer, or Boolean, stating the possible values for that type, the operations that can be done on that type, and the way the values of that type are stored.[6]

We think our best and determine the entities and attributes for our Database System. The Entities and Attributes are given below.
**Fig 1: Determining Entities and Attributes.**

<table>
<thead>
<tr>
<th>Entity</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient</td>
<td>Pat_id, Pat_name, Age, Sex, DOB, MOB</td>
</tr>
<tr>
<td>Accountant</td>
<td>Acct_id, Acct_name, Age, Address, MOB, Working_time, Acct_Salary</td>
</tr>
<tr>
<td>Doctor</td>
<td>Doc_id, Doc_name, Doc_type, Designation, Age, Address, MOB, Passed_from, Salary</td>
</tr>
<tr>
<td>Test</td>
<td>Test_id, Test_name, Date, Rep_date, Time, Fee</td>
</tr>
<tr>
<td>Receptionist</td>
<td>Rcp_id, Rcp_name, Age, Address, MOB, Shifting, Rcp_Salary</td>
</tr>
<tr>
<td>Medicine</td>
<td>Mdcn_id, Mdcn_name, Company, M_date, E_Date, Price</td>
</tr>
<tr>
<td>OT</td>
<td>OT_id, OT_room_name</td>
</tr>
<tr>
<td>Driver</td>
<td>Dr_id, Dr_name, MOB, Address, W_shift, Dr_Salary</td>
</tr>
<tr>
<td>Carriers</td>
<td>Cr_id, Cr_name, MOB, Address, Cr_Salary</td>
</tr>
<tr>
<td>Department</td>
<td>Dept_id, Dept_name, Treatment</td>
</tr>
<tr>
<td>Nurse</td>
<td>Nrs_id, Nrs_name, Age, Address, MOB, Nrs_wc_shift, Experience, Nrs_Salary</td>
</tr>
<tr>
<td>Room</td>
<td>Room_id, Room_num, Room_type, Room_cost</td>
</tr>
<tr>
<td>Bill</td>
<td>Bill_id, Bill_for, Bill_type, Bill_total</td>
</tr>
<tr>
<td>Ward Boy</td>
<td>Wb_id, Wb_name, MOB, W_shift, Wb_Salary</td>
</tr>
</tbody>
</table>
2.2 Entity Relationship Diagram (ERD):

We draw the Entity Relationship Diagram (ERD) very carefully and efficiently for the whole system of BIRDEM. We were able to cover all probable information of BIRDEM in our ERD. The ERD is given below:

Fig2: Entity Relationship Diagram (ERD).
2.3 Relational Model:

After completing the ERD successfully we made the relational model (table schemas) taking into account all the entities and the relationships.

Patient Table:

<table>
<thead>
<tr>
<th>Pat_id</th>
<th>Pat_name</th>
<th>Age</th>
<th>Sex</th>
<th>Address</th>
<th>DOB</th>
<th>MOB</th>
</tr>
</thead>
</table>

Room Table:

<table>
<thead>
<tr>
<th>Room_id</th>
<th>Room_No</th>
<th>Room_type</th>
<th>Room_cost</th>
</tr>
</thead>
</table>

Receptionist Table:

<table>
<thead>
<tr>
<th>Rcp_id</th>
<th>Rcp_name</th>
<th>Age</th>
<th>Address</th>
<th>MOB</th>
<th>shifting</th>
<th>salary</th>
</tr>
</thead>
</table>

Admission Table:

This is a junction table between Patient, Receptionist & Room tables.

<table>
<thead>
<tr>
<th>Admsn_id</th>
<th>Pat_id</th>
<th>Room_id</th>
<th>Rcp_id</th>
<th>date</th>
<th>time</th>
</tr>
</thead>
</table>

Doctor Table:

<table>
<thead>
<tr>
<th>Doc_id</th>
<th>Doc_name</th>
<th>Doc_type</th>
<th>Designation</th>
<th>Age</th>
<th>Address</th>
<th>MOB</th>
<th>Passed_from</th>
<th>Salary</th>
</tr>
</thead>
</table>
Appointment Table:-
This is a junction table between Patient, Receptionist & Doctor tables.

<table>
<thead>
<tr>
<th>Ap_id</th>
<th>Pat_id</th>
<th>Doc_id</th>
<th>Rcp_id</th>
<th>apnmt_date</th>
<th>apnmt_time</th>
</tr>
</thead>
</table>

Bill Table:-

<table>
<thead>
<tr>
<th>Bill_id</th>
<th>Bill_for</th>
<th>Bill_type</th>
<th>Bill_total</th>
</tr>
</thead>
</table>

Accountant Table:-

<table>
<thead>
<tr>
<th>Acct_id</th>
<th>Acct_name</th>
<th>Age</th>
<th>Address</th>
<th>MOB</th>
<th>Working_time</th>
<th>Acct_salary</th>
</tr>
</thead>
</table>

Payment Table:-
This is a junction table between Patient, Bill & Accountant Tables.

<table>
<thead>
<tr>
<th>Pay_id</th>
<th>Bill_for</th>
<th>Pat_id</th>
<th>Acct_id</th>
<th>Pay_type</th>
<th>Pay_date</th>
</tr>
</thead>
</table>

Medicine Table:-

<table>
<thead>
<tr>
<th>Mdcn_id</th>
<th>Mdcn_name</th>
<th>Company</th>
<th>m_date</th>
<th>e_date</th>
<th>price</th>
</tr>
</thead>
</table>

Prescription Table: -
This is a junction table between Patient, Doctor & Medicine tables.

<table>
<thead>
<tr>
<th>Prs_id</th>
<th>Doc_id</th>
<th>Mdcn_id</th>
<th>Pat_id</th>
<th>date</th>
<th>Fee</th>
</tr>
</thead>
</table>
Test Table:-

<table>
<thead>
<tr>
<th>Test_id</th>
<th>Test_name</th>
<th>date</th>
<th>rep_date</th>
<th>fee</th>
</tr>
</thead>
</table>

Assist Table:-

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

<table>
<thead>
<tr>
<th>Serial_no</th>
<th>Pat_id</th>
<th>Doc_id</th>
<th>Test_id</th>
<th>time</th>
<th>date</th>
</tr>
</thead>
</table>

OT Table:-

<table>
<thead>
<tr>
<th>Ot_id</th>
<th>Ot_room_no</th>
</tr>
</thead>
</table>

Operation Table:-

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

<table>
<thead>
<tr>
<th>Op_id</th>
<th>Doc_id</th>
<th>Pat_id</th>
<th>Ot_id</th>
<th>Op_date</th>
<th>Op_time</th>
</tr>
</thead>
</table>

Department Table:-

<table>
<thead>
<tr>
<th>Dept_id</th>
<th>Dept_name</th>
<th>treatment</th>
</tr>
</thead>
</table>

Doctor_from_Department Table:-

This is a junction table between Doctor & Department tables.

<table>
<thead>
<tr>
<th>Dfd_id</th>
<th>Doc_id</th>
<th>Dept_id</th>
</tr>
</thead>
</table>

Nurse Table:-

<table>
<thead>
<tr>
<th>Nrs_id</th>
<th>Nrs_name</th>
<th>Age</th>
<th>Address</th>
<th>MoB</th>
<th>Nrs_wo_shift</th>
<th>experience</th>
<th>Salary</th>
</tr>
</thead>
</table>
Nursing_Service Table:
This is a junction table between Patient, Room & Nurse tables.

<table>
<thead>
<tr>
<th>Ns_id</th>
<th>Pat_id</th>
<th>Nrs_id</th>
<th>Room_id</th>
</tr>
</thead>
</table>

Ward Boy Table:

<table>
<thead>
<tr>
<th>Wb_id</th>
<th>wb_name</th>
<th>MoB</th>
<th>w_shift</th>
<th>Salary</th>
</tr>
</thead>
</table>

Cleaning Service Table:
This is a junction table between Patient, Room & Ward Boy tables.

<table>
<thead>
<tr>
<th>Cls_id</th>
<th>Pat_id</th>
<th>Wb_id</th>
<th>Room_id</th>
</tr>
</thead>
</table>

Driver Table:

<table>
<thead>
<tr>
<th>Dr_id</th>
<th>Dr_name</th>
<th>Mob</th>
<th>Address</th>
<th>Shift</th>
<th>Salary</th>
</tr>
</thead>
</table>

Ambulance Table:

<table>
<thead>
<tr>
<th>Amb_id</th>
<th>Amb_num</th>
<th>Capacity</th>
</tr>
</thead>
</table>

Ambulance Service Table:
This is a junction table between Patient, Driver & Ambulance tables.

<table>
<thead>
<tr>
<th>As_id</th>
<th>Pat_id</th>
<th>Dr_id</th>
<th>Amb_id</th>
</tr>
</thead>
</table>

Carriers Table:

<table>
<thead>
<tr>
<th>Cr_id</th>
<th>Cr_name</th>
<th>MOB</th>
<th>Address</th>
<th>Salary</th>
</tr>
</thead>
</table>
Carrying Service Table:-

This is a junction table between Patient, Ambulance & Carriers tables.

<table>
<thead>
<tr>
<th>CS_id</th>
<th>Cr_id</th>
<th>Amb_id</th>
<th>Pat_id</th>
</tr>
</thead>
</table>

2.3.1 Relational Tables’ Descriptions

Patient table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pat_id</td>
<td>int</td>
<td>Unique id for a Patient</td>
</tr>
<tr>
<td>Pat_name</td>
<td>varchar(20)</td>
<td>Patient’s Name</td>
</tr>
<tr>
<td>Age</td>
<td>int</td>
<td>Patient’s Age</td>
</tr>
<tr>
<td>Sex</td>
<td>varchar(20)</td>
<td>Patient is Male or Female</td>
</tr>
<tr>
<td>Address</td>
<td>varchar(20)</td>
<td>Patient’s Address</td>
</tr>
<tr>
<td>Dob</td>
<td>varchar(20)</td>
<td>Date of Birth</td>
</tr>
<tr>
<td>Mob</td>
<td>int</td>
<td>Mobile Number</td>
</tr>
</tbody>
</table>

Room table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room_id</td>
<td>int</td>
<td>Unique id for a Room</td>
</tr>
<tr>
<td>Room_no</td>
<td>varchar(20)</td>
<td>Room number</td>
</tr>
<tr>
<td>Room_type</td>
<td>varchar(20)</td>
<td>Room is VIP or Normal</td>
</tr>
<tr>
<td>Room_cost</td>
<td>int</td>
<td>Cost of the Room</td>
</tr>
</tbody>
</table>
Receptionist table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rcp_id</td>
<td>int</td>
<td>Unique id for a Receptionist</td>
</tr>
<tr>
<td>Rcp_name</td>
<td>varchar(20)</td>
<td>Receptionist’s name</td>
</tr>
<tr>
<td>Age</td>
<td>int</td>
<td>Receptionist’s age</td>
</tr>
<tr>
<td>Address</td>
<td>varchar(20)</td>
<td>Receptionist’s Address</td>
</tr>
<tr>
<td>MOB</td>
<td>int</td>
<td>Mobile Number</td>
</tr>
<tr>
<td>Shifting</td>
<td>varchar(20)</td>
<td>Receptionist working shift</td>
</tr>
<tr>
<td>Salary</td>
<td>int</td>
<td>Salary a Receptionist gets</td>
</tr>
</tbody>
</table>

Admission table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admsn_id</td>
<td>int</td>
<td>Unique id for an Admission</td>
</tr>
<tr>
<td>Pat_id</td>
<td>int</td>
<td>Unique id for a Patient</td>
</tr>
<tr>
<td>Room_id</td>
<td>int</td>
<td>Unique id for a Room</td>
</tr>
<tr>
<td>Rcp_id</td>
<td>int</td>
<td>Unique id for a Receptionist</td>
</tr>
<tr>
<td>Date</td>
<td>varchar(20)</td>
<td>Date of Admission</td>
</tr>
</tbody>
</table>
Doctor table:

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc_id</td>
<td>int</td>
<td>Unique id for a Doctor</td>
</tr>
<tr>
<td>Doc_name</td>
<td>varchar(20)</td>
<td>Doctor's name</td>
</tr>
<tr>
<td>Doc_type</td>
<td>varchar(20)</td>
<td>Doctor's specialty</td>
</tr>
<tr>
<td>Age</td>
<td>int</td>
<td>Doctor's age</td>
</tr>
<tr>
<td>Address</td>
<td>varchar(20)</td>
<td>Doctor's address</td>
</tr>
<tr>
<td>Mob</td>
<td>int</td>
<td>Mobile Number</td>
</tr>
<tr>
<td>Designation</td>
<td>varchar(20)</td>
<td>Doctor's designation</td>
</tr>
<tr>
<td>Passed_from</td>
<td>varchar(20)</td>
<td>Doctor is passed from which medical college</td>
</tr>
<tr>
<td>Salary</td>
<td>int</td>
<td>Salary of a doctor</td>
</tr>
</tbody>
</table>

Appointment table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apnmt_id</td>
<td>int</td>
<td>Unique id for an Appointment</td>
</tr>
<tr>
<td>Pat_id</td>
<td>int</td>
<td>Unique id for a Patient</td>
</tr>
<tr>
<td>Doc_id</td>
<td>int</td>
<td>Unique id for a Doctor</td>
</tr>
<tr>
<td>Rcp_id</td>
<td>int</td>
<td>Unique id for a Receptionist</td>
</tr>
<tr>
<td>Apnmt_date</td>
<td>varchar(20)</td>
<td>Date of an Appointment</td>
</tr>
</tbody>
</table>
### Bill table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bill_id</td>
<td>int</td>
<td>Unique id for a Bill</td>
</tr>
<tr>
<td>Bill_for</td>
<td>varchar(20)</td>
<td>Purpose of the bill</td>
</tr>
<tr>
<td>Bill_type</td>
<td>varchar(20)</td>
<td>Bill either in Cash or Check</td>
</tr>
<tr>
<td>Bill_total</td>
<td>int</td>
<td>Total amount</td>
</tr>
</tbody>
</table>

### Accountant table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acct_id</td>
<td>int</td>
<td>Unique id for an Accountant</td>
</tr>
<tr>
<td>Acct_name</td>
<td>varchar(20)</td>
<td>Accountant’s Name</td>
</tr>
<tr>
<td>Age</td>
<td>int</td>
<td>Accountant’s age</td>
</tr>
<tr>
<td>Address</td>
<td>varchar(20)</td>
<td>Accountant’s Address</td>
</tr>
<tr>
<td>Mob</td>
<td>int</td>
<td>Mobile Number</td>
</tr>
<tr>
<td>Acct_salary</td>
<td>int</td>
<td>Salary of an Accountant</td>
</tr>
</tbody>
</table>
**Payment table**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pay_id</td>
<td>int</td>
<td>Unique id for a Payment</td>
</tr>
<tr>
<td>Bill_id</td>
<td>int</td>
<td>Unique id for a Bill</td>
</tr>
<tr>
<td>Pat_id</td>
<td>int</td>
<td>Unique id for a Patient</td>
</tr>
<tr>
<td>Acct_id</td>
<td>int</td>
<td>Unique id for an Accountant</td>
</tr>
<tr>
<td>Pay_type</td>
<td>varchar(20)</td>
<td>Payment in Cash or Check</td>
</tr>
<tr>
<td>Pay_date</td>
<td>varchar(20)</td>
<td>Date of Payment</td>
</tr>
</tbody>
</table>

**Medicine table**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mdcn_id</td>
<td>int</td>
<td>Unique id for a Medicine</td>
</tr>
<tr>
<td>Mdcn_name</td>
<td>varchar(20)</td>
<td>Medicine’s Name</td>
</tr>
<tr>
<td>company</td>
<td>varchar(20)</td>
<td>Medicine’s Company</td>
</tr>
<tr>
<td>M_date</td>
<td>varchar(20)</td>
<td>Manufacture Date</td>
</tr>
<tr>
<td>E_date</td>
<td>varchar(20)</td>
<td>Expire Date</td>
</tr>
<tr>
<td>price</td>
<td>int</td>
<td>Price of the Medicine</td>
</tr>
</tbody>
</table>
### Prescription table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prs_id</td>
<td>int</td>
<td>Unique id for a Prescription</td>
</tr>
<tr>
<td>Doc_id</td>
<td>int</td>
<td>Unique id for a Doctor</td>
</tr>
<tr>
<td>Mdcn_id</td>
<td>int</td>
<td>Unique id for a Medicine</td>
</tr>
<tr>
<td>Pat_id</td>
<td>int</td>
<td>Unique id for a Patient</td>
</tr>
<tr>
<td>Date</td>
<td>varchar(20)</td>
<td>Date of the Prescription</td>
</tr>
<tr>
<td>Time</td>
<td>varchar(20)</td>
<td>Time of the Prescription</td>
</tr>
<tr>
<td>Fee</td>
<td>varchar(20)</td>
<td>Prescription Fees</td>
</tr>
</tbody>
</table>

### Test table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test_id</td>
<td>int</td>
<td>Unique id for a Test</td>
</tr>
<tr>
<td>Test_name</td>
<td>varchar(20)</td>
<td>Name of the Test</td>
</tr>
<tr>
<td>Date</td>
<td>varchar(20)</td>
<td>Date of Test</td>
</tr>
<tr>
<td>Rep_date</td>
<td>varchar(20)</td>
<td>Date of the Report</td>
</tr>
<tr>
<td>Fee</td>
<td>int</td>
<td>Test Fees</td>
</tr>
</tbody>
</table>
### Assist table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial_no</td>
<td>int</td>
<td>Unique id for an Assisted Test directed to a Patient by a Doctor</td>
</tr>
<tr>
<td>Pat_id</td>
<td>int</td>
<td>Unique id for a Patient</td>
</tr>
<tr>
<td>Doc_id</td>
<td>int</td>
<td>Unique id for a Doctor</td>
</tr>
<tr>
<td>Test_id</td>
<td>int</td>
<td>Unique id for a Test</td>
</tr>
<tr>
<td>Date</td>
<td>varchar(20)</td>
<td>Date of the Assisted Test</td>
</tr>
<tr>
<td>Time</td>
<td>varchar(20)</td>
<td>Time of the Assisted Test</td>
</tr>
</tbody>
</table>

### OT table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ot_id</td>
<td>int</td>
<td>Unique id for an Operation Theater (OT)</td>
</tr>
<tr>
<td>Ot_room_no</td>
<td>varchar(20)</td>
<td>OT Room Number</td>
</tr>
</tbody>
</table>
### Operation table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op_id</td>
<td>int</td>
<td>Unique id for an Operation</td>
</tr>
<tr>
<td>Doc_id</td>
<td>int</td>
<td>Unique id for a Doctor</td>
</tr>
<tr>
<td>Pat_id</td>
<td>int</td>
<td>Unique id for a Patient</td>
</tr>
<tr>
<td>Ot_id</td>
<td>int</td>
<td>Unique id for an OT</td>
</tr>
<tr>
<td>Op_date</td>
<td>varchar(20)</td>
<td>Date of the Operation</td>
</tr>
<tr>
<td>Op_time</td>
<td>varchar(20)</td>
<td>Time of the Operation</td>
</tr>
</tbody>
</table>

### Department table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dept_id</td>
<td>int</td>
<td>Unique id for a Department</td>
</tr>
<tr>
<td>Dept_name</td>
<td>varchar(20)</td>
<td>Department's name</td>
</tr>
<tr>
<td>treatment</td>
<td>varchar(20)</td>
<td>Treatments of a patient conducted in a Department</td>
</tr>
</tbody>
</table>
### Doctor_from_Department table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dfd_id</td>
<td>int</td>
<td>Unique id for a DoctorsfromDepartment junction table</td>
</tr>
<tr>
<td>Doc_id</td>
<td>int</td>
<td>Unique id for a Doctor</td>
</tr>
<tr>
<td>Dept_id</td>
<td>int</td>
<td>Unique id for a Department</td>
</tr>
</tbody>
</table>

### Nurse table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nrs_id</td>
<td>int</td>
<td>Unique id for a Nurse</td>
</tr>
<tr>
<td>Nrs_name</td>
<td>varchar(20)</td>
<td>Nurse’s Name</td>
</tr>
<tr>
<td>Age</td>
<td>int</td>
<td>Nurse’s age</td>
</tr>
<tr>
<td>Address</td>
<td>varchar(20)</td>
<td>Nurse’s Address</td>
</tr>
<tr>
<td>Mob</td>
<td>int</td>
<td>Mobile Number</td>
</tr>
<tr>
<td>Nrs_wo_shift</td>
<td>varchar(20)</td>
<td>Nurse working Shift example morning, day, evening, night</td>
</tr>
<tr>
<td>Experience</td>
<td>varchar(20)</td>
<td>Nurse’s Experience</td>
</tr>
<tr>
<td>salary</td>
<td>int</td>
<td>Salary of a Nurse</td>
</tr>
</tbody>
</table>
Nursing_Service table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ns_id</td>
<td>int</td>
<td>Unique id for a Nursing Service</td>
</tr>
<tr>
<td>Pat_id</td>
<td>int</td>
<td>Unique id for a Patient</td>
</tr>
<tr>
<td>Nrs_id</td>
<td>int</td>
<td>Unique id for a Nurse</td>
</tr>
<tr>
<td>Room_id</td>
<td>int</td>
<td>Unique id for a Room</td>
</tr>
<tr>
<td>Date</td>
<td>varchar(20)</td>
<td>Date of Nursing Service</td>
</tr>
<tr>
<td>Time</td>
<td>varchar(20)</td>
<td>Time of Nursing Service</td>
</tr>
</tbody>
</table>

Ward_boy table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wb_id</td>
<td>int</td>
<td>Unique id for a Ward Boy</td>
</tr>
<tr>
<td>Wb_name</td>
<td>varchar(20)</td>
<td>Ward Boy’s Name</td>
</tr>
<tr>
<td>Mob</td>
<td>int</td>
<td>Mobile Number</td>
</tr>
<tr>
<td>W_shift</td>
<td>varchar(20)</td>
<td>Working shift of a Ward Boy</td>
</tr>
<tr>
<td>salary</td>
<td>int</td>
<td>Salary of a Ward boy</td>
</tr>
</tbody>
</table>
**Cleaning_Service table**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cls_id</td>
<td>int</td>
<td>Unique id for a Cleaning Service</td>
</tr>
<tr>
<td>Pat_id</td>
<td>int</td>
<td>Unique id for a Patient</td>
</tr>
<tr>
<td>Wb_id</td>
<td>int</td>
<td>Unique id for a Ward Boy</td>
</tr>
<tr>
<td>Room_id</td>
<td>int</td>
<td>Unique id for a Room</td>
</tr>
<tr>
<td>Date</td>
<td>varchar(20)</td>
<td>Date of Cleaning Service</td>
</tr>
<tr>
<td>Time</td>
<td>varchar(20)</td>
<td>Time of Cleaning Service</td>
</tr>
</tbody>
</table>

**Driver table**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr_id</td>
<td>int</td>
<td>Unique id for a Driver</td>
</tr>
<tr>
<td>Dr_name</td>
<td>varchar(20)</td>
<td>Driver’s Name</td>
</tr>
<tr>
<td>mob</td>
<td>int</td>
<td>Mobile Number</td>
</tr>
<tr>
<td>address</td>
<td>varchar(20)</td>
<td>Driver’s Address</td>
</tr>
<tr>
<td>Shift</td>
<td>varchar(20)</td>
<td>Working shift of a Driver</td>
</tr>
<tr>
<td>salary</td>
<td>int</td>
<td>Salary of a Driver</td>
</tr>
</tbody>
</table>
Ambulance table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amb_id</td>
<td>int</td>
<td>Unique id for an Ambulance</td>
</tr>
<tr>
<td>Amb_num</td>
<td>varchar(20)</td>
<td>Ambulance’s Number</td>
</tr>
<tr>
<td>Capacity</td>
<td>int</td>
<td>Capacity of an Ambulance</td>
</tr>
</tbody>
</table>

Ambulance_Service table

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>As_id</td>
<td>int</td>
<td>Unique id for an Ambulance Service</td>
</tr>
<tr>
<td>Pat_id</td>
<td>int</td>
<td>Unique id for a Patient</td>
</tr>
<tr>
<td>Dr_id</td>
<td>int</td>
<td>Unique id for a Driver</td>
</tr>
<tr>
<td>Amb_id</td>
<td>int</td>
<td>Unique id for an Ambulance</td>
</tr>
<tr>
<td>Date</td>
<td>varchar(20)</td>
<td>Date of the Ambulance Service</td>
</tr>
<tr>
<td>Time</td>
<td>varchar(20)</td>
<td>Time of the Ambulance Service</td>
</tr>
</tbody>
</table>
**Carriers table**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cr_id</td>
<td>int</td>
<td>Unique id for a Carrier who will carry patients inside the hospital’s premises from the ambulance.</td>
</tr>
<tr>
<td>Cr_name</td>
<td>varchar(20)</td>
<td>Carrier’s Name</td>
</tr>
<tr>
<td>Mob</td>
<td>int</td>
<td>Mobile Number</td>
</tr>
<tr>
<td>Address</td>
<td>varchar(20)</td>
<td>Carrier’s Address</td>
</tr>
<tr>
<td>Salary</td>
<td>int</td>
<td>Salary of a Carrier</td>
</tr>
</tbody>
</table>

**Carrying_Service table**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cs_id</td>
<td>int</td>
<td>Unique id for a Carrying Service</td>
</tr>
<tr>
<td>Cr_id</td>
<td>int</td>
<td>Unique id for a Carrier</td>
</tr>
<tr>
<td>Amb_id</td>
<td>int</td>
<td>Unique id for an Ambulance</td>
</tr>
<tr>
<td>Pat_id</td>
<td>int</td>
<td>Unique id for a Patient</td>
</tr>
<tr>
<td>Date</td>
<td>varchar(20)</td>
<td>Date of the Carrying Service</td>
</tr>
<tr>
<td>Time</td>
<td>varchar(20)</td>
<td>Time of the Carrying Service</td>
</tr>
</tbody>
</table>
2.3.2 Explanation of Relational Model

# Relationship between Receptionist, Patient and Room Entities in the ER Model:

- 1 Receptionist can admit 1 Patient in 1 Room in a certain date and time.
- 1 Receptionist can admit in 1 Room 1 Patient in a certain date and time.
- In 1 Room, 1 Patient is admitted by 1 Receptionist in a certain date and time.

So the relationship is a Ternary Relationship named Admission (in the diamond) with cardinality ratio from Patient to Receptionist to Room as 1 to 1 to 1.
# Relational model for Receptionist, Patient and Room Entities:

Receptionist, Patient and Room Entities become Receptionist, Patient and Room tables.

**Patient Table:-**

<table>
<thead>
<tr>
<th>Pat_id</th>
<th>Pat_name</th>
<th>Age</th>
<th>Sex</th>
<th>DOB</th>
<th>MOB</th>
<th>Address</th>
</tr>
</thead>
</table>

**Room Table:-**

<table>
<thead>
<tr>
<th>Room_id</th>
<th>Room_No</th>
<th>Room_type</th>
<th>Room_cost</th>
</tr>
</thead>
</table>

**Receptionist Table:-**

<table>
<thead>
<tr>
<th>Rcp_id</th>
<th>Rcp_name</th>
<th>Age</th>
<th>Address</th>
<th>MOB</th>
<th>shifting</th>
<th>salary</th>
</tr>
</thead>
</table>

The junction *Admission* also becomes a table.

**Admission Table:-**

<table>
<thead>
<tr>
<th>admsn_id</th>
<th>Pat_id</th>
<th>Room_id</th>
<th>Rcp_id</th>
<th>Date</th>
<th>time</th>
</tr>
</thead>
</table>

- Primary Key of the Patient Table goes to Admission Table as Foreign Key.
- Primary Key of the Room Table goes to Admission Table as Foreign Key.
- Primary Key of the Receptionist Table goes to Admission Table as Foreign Key.

Since the Cardinality Ratio from Patient to Receptionist to Room is 1 to 1 to 1,
admsn_id is a Primary key in the Admission Table. Pat_id from Patient Table, Room_id from Room Table and Rcp_id from Receptionist Table become Foreign Keys in the Admission Table.

In a similar way, as cardinality ratio for Receptionist_Patient_Doctor relationship is 1 to 1 to 1, Receptionist, Patient and Doctor entities become separate tables along with a junction Appointment table which has Rcp_id, Pat_id and Doc_id as foreign keys. Similar logic applies to Patient_Ambulance_Driver relationship with cardinality ratio 1 to 1 to 1.

# Relationship between Doctor and Department Entities in the ER Model:

- 1 Doctor can be from 1 or Many Departments.
- 1 Department may have 1 or Many Doctors.

So it is a Many to Many relationship named Doctor from Department (in the diamond).
# Relational model for Doctor and Department Entities:

Doctor and Department Entities become Doctor and Department tables.

**Doctor Table:**

<table>
<thead>
<tr>
<th>Doc_id</th>
<th>Doc_name</th>
<th>Doc_type</th>
<th>Designation</th>
<th>Age</th>
<th>Address</th>
<th>MOB</th>
<th>Passed_from</th>
<th>Salary</th>
</tr>
</thead>
</table>

**Department Table:**

<table>
<thead>
<tr>
<th>Dept_id</th>
<th>Dept_name</th>
<th>treatment</th>
</tr>
</thead>
</table>

The junction table *Doctor from Department* also becomes a table.

**Doctor_from_Department Table:**

<table>
<thead>
<tr>
<th>Dfd_id</th>
<th>Doc_id</th>
<th>Dept_id</th>
</tr>
</thead>
</table>

- Primary Key of the Doctor Table goes to Doctor_from_Department Table as part of Primary Key.
- Primary Key of the Department Table goes to Doctor_from_Department Table as part of Primary Key.

Since the Cardinality Ratio from Doctor to Department is Many to Many, Dfd_id is a part of Primary key in the Doctor_from_Department Table. Doc_id from Doctor Table and Dept_id from Department Table become parts of Primary Key in the Doctor_from_Department Table.
# Relationship between Patient, Doctor and Medicine Entities in the ER Model:

- 1 Doctor gives 1 patient 1 or more medicine.
- 1 patient takes 1 medicine prescribed by 1 doctor.
- 1 medicine is prescribed by 1 doctor to 1 patient.

So the relationship is a Ternary Relationship named Prescription (in the diamond) with a Cardinality Ratio from Patient to Doctor to Medicine 1 to 1 to Many.
# Relational model for Patient, Doctor and Medicine Entities:

Patient, Doctor and Medicine Entities become Patient, Doctor and Medicine tables.

Patient Table:-

<table>
<thead>
<tr>
<th>Pat_Id</th>
<th>Pat_name</th>
<th>Age</th>
<th>Sex</th>
<th>DOB</th>
<th>MOB</th>
<th>Address</th>
</tr>
</thead>
</table>

Doctor Table:-

<table>
<thead>
<tr>
<th>Doc_id</th>
<th>Doc_name</th>
<th>Doc_type</th>
<th>Designation</th>
<th>Age</th>
<th>Address</th>
<th>MOB</th>
<th>Passed_from</th>
<th>Salary</th>
</tr>
</thead>
</table>

Medicine Table:-

<table>
<thead>
<tr>
<th>Mdcn_id</th>
<th>Mdcn_name</th>
<th>company</th>
<th>m_date</th>
<th>e_date</th>
<th>price</th>
</tr>
</thead>
</table>

Prescription Table: -

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

<table>
<thead>
<tr>
<th>Prs_id</th>
<th>Doc_id</th>
<th>Mdcn_id</th>
<th>Pat_id</th>
<th>date</th>
<th>fee</th>
</tr>
</thead>
</table>

- Primary Key of the Patient Table goes to Prescription Table as Foreign Key.
- Primary Key of the Doctor Table goes to Prescription Table as Foreign Key.
- Primary Key of the Medicine Table goes to Prescription Table as part of Primary Key.

Since the Cardinality Ratio from Patient to Doctor to Medicine 1 to 1 to M, Prs_id is a Primary key in the Prescription Table. Pat_id from Patient Table,
**Doc_id** from Doctor Table and **Mdcn_id** from Medicine Table become Foreign Keys in the Admission Table.

In a similar way relational tables have been designed for Patient-Doctor-Test, Patient-OT-Doctor, Patient-Bill-Accountant relationships with cardinality ratio 1 to 1 to M. Similar logic applies for Patient-Ambulance-Carrier relationship with cardinality ratio 1 to 1 to M.

#Relationship Between Patient, Room & Nurse Entities in the ER Model :-

- 1 room is fixed for 1 Patient to provide nursing service for 1 or Many nurses in a certain date.
- 1 patient receives nursing service from 1 Nurse in 1 Room in a certain date.
- 1 nurse can render proper services in 1 room to many patients in a certain date.

So it is a Ternary Relationship named Nursing Services (in the diamond) with cardinality Ratio from Room to Nurse to Patient 1 to M to M.
# Relational model between Patient, Nurse and Room Entities:

**Patient Table:**

<table>
<thead>
<tr>
<th>Pat_id</th>
<th>Pat_name</th>
<th>Age</th>
<th>Sex</th>
<th>DOB</th>
<th>MOB</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Room Table:**

<table>
<thead>
<tr>
<th>Room_id</th>
<th>Room_No</th>
<th>Room_type</th>
<th>Room_cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Nurse Table:**

<table>
<thead>
<tr>
<th>Nrs_id</th>
<th>Nrs_name</th>
<th>Age</th>
<th>Address</th>
<th>Mob</th>
<th>Nrs_wo_shift</th>
<th>experience</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Nursing Service Table:**

This is a junction table between Patient, Room and Nurse Table.

<table>
<thead>
<tr>
<th>Ns_id</th>
<th>Pat_id</th>
<th>Nrs_id</th>
<th>Room_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Primary Key of the Patient Table goes to Nursing Service Table as part of Primary Key.
- Primary Key of the Nurse Table goes to Nursing Service Table as part of Primary Key.
- Primary Key of the Room Table goes to Nursing Service Table as Foreign Key.

Since the Cardinality Ratio from Room to Patient to Nurse is 1 to M to M. Ns_id is a Primary key in the Nursing Service Table. Pat_id from Patient Table, Nrs_id
from Nurse Table become parts of Primary Key in the Nursing Service Table. Room_id from Room Table becomes Foreign Key in the Nursing Service Table. In a similar way relational tables are created for Patient-Room-Wardboy relationship with cardinality ratio 1 to M to M.

2.4 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

2.4.1 Functional Dependencies

Definition of functional dependencies:
Given a relational schema R (A1, A2, ..., An) and X, Y {A1, ..., An}. Then X -> Y means that for every extension of R, the following holds:
R contains no two tuples that are equal in all values of X but differ in at least one value of Y.
(Pronunciation: "X determines Y functionally" "Y is functionally dependent of X").

Example:
Student (matNr, name):
{matNr} -> {name}

Definition of *full* functional dependencies:
Prerequisites as in Definition 1.
Y is said to be fully functionally dependent of X, if there is no proper subset X'
⊂ X,
Where X' -> Y.
Notation: X => Y.

Example:
A University Database:-
Class (classId, room, day, pName)
{classId, room} -> {pName}
{classId, day, pName} -> {room}
{classId} => {pName}
{classId} => {room} [7]

2.4.2 Normalization

Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.
It has mainly two goals:-

✓ First goal: eliminate redundant data
   For example, storing the same data in more than one table

✓ Second Goal: ensure data dependencies make sense
   For example, only storing related data in a table

Benefits of Normalization:

- Less storage space
- Quicker updates
- Less data inconsistency
- Clearer data relationships
- Easier to add data
- Flexible Structure

Bad database designs results in:

- Redundancy: inefficient storage.
- Anomalies: data inconsistency, difficulties in maintenance.[7]

1NF, 2NF, 3NF, BCNF are some of the early forms in the list that address this problem.

First Normal Form (1NF)

Definition:
A relation is in first normal form if it contains only simple, atomic values for attributes, no sets. Example:

<table>
<thead>
<tr>
<th>Name</th>
<th>Offspring</th>
<th>Place</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Child</td>
<td>Age</td>
</tr>
<tr>
<td>Muller</td>
<td>Christa</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Peter</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Iris</td>
<td>9</td>
</tr>
<tr>
<td>Schmidt</td>
<td>Martin</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Rainer</td>
<td>18</td>
</tr>
</tbody>
</table>
The value of an attribute can be a relation by itself.
⇒ Operations in the model are much more complicated
⇒ In order to keep the model simple: 1NF

Ways to normalize the above relation:

First attempt:
Person (name, place, child1, child2, child3)

⇒ Not good. Reason: either not enough available columns for some data records (How many children can a person have??) Or, if there are enough columns to provide for all thinkable cases, waste of much space (many NULL values).

Second attempt:

Person:-

<table>
<thead>
<tr>
<th>pName</th>
<th>place</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muller</td>
<td>Stuggart</td>
</tr>
<tr>
<td>Schmidt</td>
<td>Trir</td>
</tr>
</tbody>
</table>

Child:-

<table>
<thead>
<tr>
<th>pName</th>
<th>chName</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muller</td>
<td>Christa</td>
<td>12</td>
</tr>
<tr>
<td>Muller</td>
<td>Peter</td>
<td>10</td>
</tr>
<tr>
<td>Muller</td>
<td>Iris</td>
<td>9</td>
</tr>
<tr>
<td>Schmidt</td>
<td>Martin</td>
<td>17</td>
</tr>
<tr>
<td>Schmidt</td>
<td>Rainer</td>
<td>18</td>
</tr>
</tbody>
</table>
Advantage:
This requires just the right amount of space that is actually needed.

Disadvantage:
It requires an additional table. pName is redundantly stored.

Second Normal Form (2NF)
Definitions:
Definition of second normal form (simple version):
A relation is in 2NF, if it is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key of the relation.

Definition of second normal form (extended version):
A relation is in 2NF, if it is in 1NF and every non-candidate-key attribute is fully functionally dependent on every candidate key.

Example:-
A University Database:

TA (matNr, classId, sName, hours, taSalary)

Full functional dependencies:
{matNr, classId} => {hours}
{matNr, classId} => {taSalary}
{matNr} => {sName}

=> TA is not in 2NF
Redundancy since the name is repeated for every occurrence of the same Matrikel Number.

Solution:

Move the dependency \{matNr\} \rightarrow \{name\} to a separate relation.

\rightarrow Relation "Student"

Third Normal Form (3NF)

Definition:-

A functional dependency X \rightarrow Y in a relation R is called a transitive dependency, if R contains a set of attributes, Z for which holds:

- A chain Exists.
- X \rightarrow Z \rightarrow Y
- Y is not a part of primary key
- Z is not a super key and
- X \rightarrow Z \rightarrow Y

Y is then called transitively dependent on X via Z.

Definition of Third Normal Form:

A Relation is in 3NF, if it is in 2NF and no non primary key attributes is transitively dependent on the primary key.

Example:-

TA (matNr, classId, hours, taSalary)
Functional dependencies:
{matNr, classId} => {hours}
{matNr, classId} => {taSalary}

Assumption:
{hours} => {taSalary}
There is the following transitive dependency:
{matNr, classId} => {hours} => {taSalary}
Since taSalary is not an attribute in a candidate key and hours is not a superkey,
TA is not in 3NF.
There is unnecessary redundancy since taSalary is repeated for each occurrence of the same value of hours.
Solution:

Move the dependency {hours} => {taSalary} to a separate relation.
Example:
TANew (matNr, classId, hours) and TASalary (hours, taSalary).

Boyce Coded Normal Form (BCNF)
A relation R is in 3NF relation and for a dependency X->A from an attributes set X to an attributes A holds that,

✓ X is not a super key

✓ In addition, A is a part of a primary key

✓ Then this relation is not also in BCNF.

In all other cases, 3NF and BCNF are identical.

BCNF is a little stronger than 3NF. In most cases, relations in 3NF are also in BCNF.
The alternative definition of BCNF shows in comparison to the 3NF definition how the two differ: in BCNF, X must always be a super key; in 3NF it does not need to be a super key if A is part of a candidate key.

✓ A relation is in BCNF, if and only if, every determinant is a candidate key.
Full functional dependencies:
- \{town, streetSegment\} => \{postCode\}
- \{town, streetSegment\} => \{speed\}
- \{postCode\} => \{town\}
- \{postcode, streetSegment\} => \{speed\}

Candidate keys:
- \{(town, streetSegment)\}
- \{(postCode, streetSegment)\}

Speedlimits is in 3NF:
- 1NF by definition
- 2NF since all non-primary-key attributes are fully functionally dependent on the primary key. For the extended definition: speed is the only attribute that is not part of a Candidate key, and it is fully functionally dependent not only on the primary key, but also on the other candidate.
- 3NF since the only non-candidate-key attribute is speed, and the only transitive Dependencies ending in speed would be from one of the keys to the other and then to speed. However, transitive dependencies where the middle set is a candidate key do not violate the definition of 3NF.

But BCNF is violated:
The problematic dependency is from an attribute \{postcode\} which is not a superkey to a part \{town\} of the primary key.

<table>
<thead>
<tr>
<th>town</th>
<th>streetSegment</th>
<th>postcode</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stuttgart</td>
<td>A-Str</td>
<td>70000</td>
<td>30</td>
</tr>
<tr>
<td>Stuttgart</td>
<td>B-Str</td>
<td>70000</td>
<td>30</td>
</tr>
<tr>
<td>Stuttgart</td>
<td>C-Str</td>
<td>70000</td>
<td>50</td>
</tr>
<tr>
<td>Stuttgart</td>
<td>D-Str</td>
<td>71234</td>
<td>70</td>
</tr>
</tbody>
</table>
Redundancy: postCode implies the town => unnecessary repetition

Transforming to BCNF:

1. Attempt:
   Speedlimit (town, streetSegment, speed)
   Codes (postCode, town)

   Schema is now in BCNF.
   • The dependency \{town, streetSegment\} => \{postCode\} is no longer recognizable.

2. Attempt:
   Speedlimit (town, streetSegment, speed)
   PostCodes (streetsegment, postCode)

   But:
   • The dependency \{town, streetSegment\} => \{postCode\} is again not recognizable.
   • The decomposition is lossy again!

3. Attempt:
   Speedlimit (postCode, streetSegment, speed)
   Codes (postCode, town)

   Now both relations are in BCNF, and the decomposition is lossless.
   However, the dependencies \{town, streetSegment\} => \{postCode\} and \{town, streetSegment\} => \{speed\} are lost.

   It is possible to show:
   • A relation that is not in BCNF can always be losslessly decomposed towards BCNF.
   • A lossless decomposition into BCNF that preserves all dependencies does not always exist. [7]

   In our thesis we will try Normalize all the relational tables.
FULFILMENT OF NORMAL FORMS:

Room Table:-

<table>
<thead>
<tr>
<th>Room_id</th>
<th>Room_no</th>
<th>Room_type</th>
<th>Room_cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

{Room_id} => {Room_no}     Functional Dependency Exist
2 different room no's do not correspond to the same Room_id.

{Room_id} => {Room_type}     Functional Dependency Exist
2 different room types' do not correspond to the same Room_id

{Room_id} => {Room_cost}     Functional Dependency Exist
2 different room cost's do not correspond to the same Room_id

Relation : (Room_id, Room_No, Room_type, Room_cost)

Full Functional Dependencies:

{Room_id} => {Room_no}

{Room_id} => {Room_type}

{Room_id} => {Room_cost}

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:-

<table>
<thead>
<tr>
<th>Bill_id</th>
<th>Bill_for</th>
<th>Bill_type</th>
<th>Bill_total</th>
</tr>
</thead>
</table>

{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, Accountant, Receptionist, Driver, Ambulance, Carriers, OT, Medicine, Test, Department and Nurse Tables fulfill all the normal forms.

**JUNCTION TABLES:**

Admission Room Table:-
This is a junction table between Patient, Room, and Receptionist Table

<table>
<thead>
<tr>
<th>Admsn_id</th>
<th>Room_id</th>
<th>Pat_id</th>
<th>Rcp_id</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
</table>

Full Functional Dependencies:

\{admsn_id\} => \{Room_id\}   Functional Dependency Exist
\{admsn_id\} => \{Rcp_id\}    Functional Dependency Exist
\{admsn_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 Ambulance Service and Appointment Tables fulfill all the normal forms.

Prescription Table:-

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

<table>
<thead>
<tr>
<th>Prs_id</th>
<th>Doc_id</th>
<th>Mdcn_id</th>
<th>Pat_id</th>
<th>Date</th>
<th>Fees</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Full Functional Dependencies:

\{\text{Prs\_id, Mdcn\_id}\} \Rightarrow \{\text{Doc\_id}\} \quad \text{Functional Dependency Exist}

\{\text{Prs\_id, Mdcn\_id}\} \Rightarrow \{\text{Pat\_id}\} \quad \text{Functional Dependency Exist}

\{\text{Prs\_id, Mdcn\_id}\} \Rightarrow \{\text{Date, Fees, Time}\} \quad \text{Functional Dependency Exist}

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

\{\text{Prs\_id, Mdcn\_id}\} \Rightarrow \{\text{Doc\_id}\}

\{\text{Prs\_id, Mdcn\_id}\} \Rightarrow \{\text{Pat\_id}\}

\{\text{Prs\_id, Mdcn\_id}\} \Rightarrow \{\text{Date}\}

\{\text{Prs\_id, Mdcn\_id}\} \Rightarrow \{\text{Time}\}

\{\text{Prs\_id, Mdcn\_id}\} \Rightarrow \{\text{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.
In a similar way Assist, Carrying Service, Cleaning Service, Operation and Nursing Service tables fulfill all normal forms.

VIOLATION OF NORMAL FORM:

Payment Table:-

This is a junction table between Patients, Bill & Accountant tables.

<table>
<thead>
<tr>
<th>Pay_id</th>
<th>Pat_id</th>
<th>Bill_id</th>
<th>Acct_id</th>
<th>Pay_type</th>
<th>Pay_date</th>
</tr>
</thead>
</table>

For Payment relation, the following functional dependencies exist:

{Pay_id} => {Pay_Type, Pay_date, Pat_id}

Two different patient ids, payment dates and payment types cannot correspond to the same payment id. So Pay_Type, Pay-date and Pat_id are fully functionally dependent on Pay_id.

{Bill_id} => {Acct_id, Pat_id}

Similarly two different accountant ids and patient ids cannot correspond to the same bill id. So Acct_Id and Pat_id are fully functionally dependent on Bill_id.

Based on the above functional dependencies:

The relation is in 1NF.

The relation is not in 2NF because all non-primary keys are not fully functionally dependent on the primary key (Pay_id, Bill_id). So we split the relation to make it 2NF.

Payment1 (Pay_id, Pay_Type, Pay_date, Pat_id)
Payment2 (Bill_id, Acct_id, Pat_id)

The relations are now in 2NF.

3NF:

There is no chain.

So the relations are in 3NF.

BCNF:

No Part of the primary key (Pay_Id, Bill_Id) is fully functionally dependent on any non primary key. So the relations are in BCNF.

2.5 Implementation in SQL Server:

After Normalization, we implemented our Database in SQL Server. There were 27 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.
Fig: Relational model Implementation on SQL Server.
2.5.1 Creation of Tables and Insertion of Data:

In our thesis we create tables and insert data using SQL server and SQL Language.

MAIN TABLE

Create Patient Table

```sql
create table Patient(
    Pat_id int primary key,
    Pat_name varchar(20),
    Address varchar(20),
    Sex varchar(20),
    Age int,
    MOB int,
    DOB varchar(20)
);

Command(s) completed successfully.
```
Insert Values into Patient Table

```
insert into Patient values(1, 'Muni', 'Uttera', 'Female', 20, 01914564987, '12/05/10')
```

(1 row(s) affected)

In this way we create all the main tables and insert data in them.

Junction Table

Create Admission Table

```
create table Admission
()

    AD_id int primary key,
    Ad_date varchar(20),
    Pat_name varchar(20) references Patient,
    Rcp_name varchar(20)
        references Receptionist,
    Room_no varchar(20) references Room,

);
```

Command(s) completed successfully.
Insert Values into Admission Table

```
insert into Admission values(2,1,2,3,'06/05/10','11 A.M. ')
```

(1 row(s) affected)

In this way we create all the junction tables and insert data in them.

### 2.5.2 Sample Data values of Tables

#### Patient table

<table>
<thead>
<tr>
<th>Pat_id</th>
<th>Pat_name</th>
<th>Age</th>
<th>Sex</th>
<th>DOB</th>
<th>MOB</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Morin</td>
<td>20</td>
<td>Female</td>
<td>12/1/80</td>
<td>16245658</td>
<td>Uttara</td>
</tr>
<tr>
<td>2</td>
<td>Karim</td>
<td>30</td>
<td>Male</td>
<td>12/12/80</td>
<td>17254638</td>
<td>Mrpur</td>
</tr>
<tr>
<td>3</td>
<td>Mamun</td>
<td>10</td>
<td>Male</td>
<td>10/10/200</td>
<td>NULL</td>
<td>Elephantroad</td>
</tr>
<tr>
<td>4</td>
<td>Rimi</td>
<td>18</td>
<td>Female</td>
<td>1/1/92</td>
<td>NULL</td>
<td>Mrpur</td>
</tr>
<tr>
<td>5</td>
<td>Kamal</td>
<td>50</td>
<td>Male</td>
<td>1/1/1960</td>
<td>19245698</td>
<td>Uttara</td>
</tr>
</tbody>
</table>

#### Room table

<table>
<thead>
<tr>
<th>Room_id</th>
<th>Room_no</th>
<th>Room_type</th>
<th>Room_cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R-101</td>
<td>Normal</td>
<td>2000</td>
</tr>
<tr>
<td>2</td>
<td>R-102</td>
<td>Normal</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>R-103</td>
<td>Normal</td>
<td>2000</td>
</tr>
<tr>
<td>4</td>
<td>R-205</td>
<td>VIP</td>
<td>4000</td>
</tr>
<tr>
<td>5</td>
<td>R-206</td>
<td>VIP</td>
<td>4000</td>
</tr>
</tbody>
</table>
Receptionist table

<table>
<thead>
<tr>
<th>Rcp_id</th>
<th>Rcp_name</th>
<th>Age</th>
<th>Sex</th>
<th>Address</th>
<th>MOB</th>
<th>Shift</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Raj</td>
<td>20</td>
<td>Male</td>
<td>Mrpur</td>
<td>1502235225</td>
<td>Morning</td>
<td>6000</td>
</tr>
<tr>
<td>2</td>
<td>Rasel</td>
<td>25</td>
<td>Male</td>
<td>Mohammadpur</td>
<td>9123545682</td>
<td>Evening</td>
<td>6000</td>
</tr>
<tr>
<td>3</td>
<td>Mir Karm</td>
<td>30</td>
<td>Male</td>
<td>Kazpara</td>
<td>1912456352</td>
<td>Night</td>
<td>6000</td>
</tr>
</tbody>
</table>

Admission room table

<table>
<thead>
<tr>
<th>Adm_id</th>
<th>Pat_id</th>
<th>Room_id</th>
<th>Rcp_id</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10/01/09</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>10/01/06</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>11/01/06</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>12/01/09</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>10/01/09</td>
</tr>
</tbody>
</table>

Doctor table

<table>
<thead>
<tr>
<th>Doc_id</th>
<th>Doc_name</th>
<th>Doc_type</th>
<th>Designation</th>
<th>Age</th>
<th>Address</th>
<th>MOB</th>
<th>Passed_from</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Selma</td>
<td>Gynecologist</td>
<td>Assistant Professor</td>
<td>50</td>
<td>Dhanmondi</td>
<td>1815749885</td>
<td>DMC</td>
<td>20000</td>
</tr>
<tr>
<td>2</td>
<td>Rahat</td>
<td>Oncologic</td>
<td>Assistant Professor</td>
<td>45</td>
<td>Mrpur</td>
<td>17137871496</td>
<td>DMC</td>
<td>20000</td>
</tr>
<tr>
<td>3</td>
<td>Mohammed</td>
<td>Heart Specialist</td>
<td>Assistant Professor</td>
<td>50</td>
<td>Dhanmondi</td>
<td>1675584562</td>
<td>DMC</td>
<td>20000</td>
</tr>
<tr>
<td>4</td>
<td>Khirma</td>
<td>Medicine</td>
<td>Assistant Professor</td>
<td>40</td>
<td>Mohammadpur</td>
<td>1897545632</td>
<td>DMC</td>
<td>18000</td>
</tr>
<tr>
<td>5</td>
<td>Nusrat</td>
<td>Medicine</td>
<td>Assistant Professor</td>
<td>45</td>
<td>Mrpur</td>
<td>1013654573</td>
<td>DMC</td>
<td>18000</td>
</tr>
</tbody>
</table>

Appointment table

<table>
<thead>
<tr>
<th>Ap_id</th>
<th>Pat_id</th>
<th>Doc_id</th>
<th>Rcp_id</th>
<th>Ap_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10/01/09</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>11/01/09</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>12/01/09</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>13/01/09</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>14/04/09</td>
</tr>
</tbody>
</table>

Bill table

<table>
<thead>
<tr>
<th>Bill_id</th>
<th>Bill_for</th>
<th>Bill_type</th>
<th>Bill_total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doctor Fee</td>
<td>Cash</td>
<td>500</td>
</tr>
<tr>
<td>2</td>
<td>Test Fee</td>
<td>Cash</td>
<td>800</td>
</tr>
<tr>
<td>3</td>
<td>Test</td>
<td>Cash</td>
<td>600</td>
</tr>
<tr>
<td>4</td>
<td>Doctor Fee</td>
<td>Cash</td>
<td>500</td>
</tr>
<tr>
<td>5</td>
<td>Medicince cost</td>
<td>Cash</td>
<td>100</td>
</tr>
</tbody>
</table>
### Accountant table

<table>
<thead>
<tr>
<th>Acct_id</th>
<th>Acct_name</th>
<th>Age</th>
<th>Address</th>
<th>MOB</th>
<th>Working_time</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rajib</td>
<td>35</td>
<td>Mohammadpur</td>
<td>1954568125</td>
<td>Morning</td>
<td>15000</td>
</tr>
<tr>
<td>2</td>
<td>Rajin</td>
<td>50</td>
<td>Mirpur</td>
<td>1715845789</td>
<td>Evening</td>
<td>15000</td>
</tr>
<tr>
<td>3</td>
<td>Monir</td>
<td>45</td>
<td>Mirpur</td>
<td>1815645362</td>
<td>Night</td>
<td>15000</td>
</tr>
</tbody>
</table>

### BillPay table

<table>
<thead>
<tr>
<th>BP_id</th>
<th>Bill_id</th>
<th>Acct_id</th>
<th>Amount</th>
<th>Pay</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>800</td>
<td>200</td>
<td>600</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>600</td>
<td>600</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>600</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>600</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>800</td>
<td>800</td>
<td>0</td>
</tr>
</tbody>
</table>

### Payment Table

<table>
<thead>
<tr>
<th>Pay_id</th>
<th>Pay_type</th>
<th>Pay_date</th>
<th>Pay_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cash</td>
<td>12/05/09</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Cash</td>
<td>13/05/09</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Cheque</td>
<td>14/05/09</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Cash</td>
<td>15/05/09</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>Cash</td>
<td>16/05/09</td>
<td>5</td>
</tr>
</tbody>
</table>

### Medicine table

<table>
<thead>
<tr>
<th>Mdcn_id</th>
<th>Mdcn_name</th>
<th>Company</th>
<th>M_Date</th>
<th>E_Date</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Napa</td>
<td>Gaxo</td>
<td>12/12/08</td>
<td>8/11/11</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>Omidon</td>
<td>Incepta</td>
<td>12/1/08</td>
<td>12/1/10</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>Ace</td>
<td>Incepta</td>
<td>12/1/08</td>
<td>12/1/10</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>Yamadin</td>
<td>Gaxo</td>
<td>1/12/10</td>
<td>1/12/12</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>Zymet</td>
<td>Gaxo</td>
<td>1/11/09</td>
<td>1/11/10</td>
<td>35</td>
</tr>
</tbody>
</table>
### Prescription table

<table>
<thead>
<tr>
<th>Prs_id</th>
<th>Mdcn_id</th>
<th>Doc_id</th>
<th>Pat_id</th>
<th>Date</th>
<th>Time</th>
<th>Fees</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12/3/09</td>
<td>8.15AM</td>
<td>500</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>13/3/09</td>
<td>9.00 AM</td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>16/3/09</td>
<td>12.00PM</td>
<td>500</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>17/3/09</td>
<td>1.00PM</td>
<td>500</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>20/3/09</td>
<td>9.00AM</td>
<td>500</td>
</tr>
</tbody>
</table>

### Test table

<table>
<thead>
<tr>
<th>Test_id</th>
<th>Test_name</th>
<th>Date</th>
<th>Rep_date</th>
<th>Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Blood</td>
<td>12/5/09</td>
<td>14/5/09</td>
<td>500</td>
</tr>
<tr>
<td>2</td>
<td>Urine</td>
<td>13/5/09</td>
<td>14/5/09</td>
<td>300</td>
</tr>
<tr>
<td>3</td>
<td>X-Ray</td>
<td>14/5/09</td>
<td>15/5/09</td>
<td>800</td>
</tr>
<tr>
<td>4</td>
<td>Ultra Sono</td>
<td>15/5/09</td>
<td>16/5/09</td>
<td>400</td>
</tr>
<tr>
<td>5</td>
<td>Engiogram</td>
<td>15/5/09</td>
<td>16/5/09</td>
<td>600</td>
</tr>
</tbody>
</table>

### Assist table

<table>
<thead>
<tr>
<th>Serial_no</th>
<th>Test_id</th>
<th>Pat_id</th>
<th>Doc_id</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12/5/09</td>
<td>9.00AM</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>13/5/09</td>
<td>10.00AM</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>14/5/09</td>
<td>11.00AM</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>15/5/09</td>
<td>11.00AM</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>16/5/09</td>
<td>12.00PM</td>
</tr>
</tbody>
</table>

### OT table

<table>
<thead>
<tr>
<th>Ot_id</th>
<th>Ot_room_no</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R-200</td>
</tr>
<tr>
<td>2</td>
<td>R-300</td>
</tr>
<tr>
<td>3</td>
<td>R-400</td>
</tr>
<tr>
<td>4</td>
<td>R-500</td>
</tr>
<tr>
<td>5</td>
<td>R-600</td>
</tr>
</tbody>
</table>
**Operation table**

<table>
<thead>
<tr>
<th>Op_id</th>
<th>Doc_Id</th>
<th>Pat_id</th>
<th>Ot_id</th>
<th>Op_date</th>
<th>Op_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12/05/09</td>
<td>12:00PM</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>12/05/09</td>
<td>12:00PM</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>12/05/09</td>
<td>12:00PM</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>13/05/09</td>
<td>1:00 PM</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>14/05/09</td>
<td>2:00PM</td>
</tr>
</tbody>
</table>

**Department table**

<table>
<thead>
<tr>
<th>Dept_id</th>
<th>Dept_name</th>
<th>Treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Orthopedics</td>
<td>Bones</td>
</tr>
<tr>
<td>2</td>
<td>Burning</td>
<td>Minimize Burn</td>
</tr>
<tr>
<td>3</td>
<td>Gyne</td>
<td>Pregnancy and s...</td>
</tr>
<tr>
<td>4</td>
<td>Cardiology</td>
<td>HEART</td>
</tr>
<tr>
<td>5</td>
<td>Medicine</td>
<td>Give proper med...</td>
</tr>
</tbody>
</table>

**Doctor_form_department table**

<table>
<thead>
<tr>
<th>Dfd_id</th>
<th>Doc_id</th>
<th>Dept_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

**Nurse table**

<table>
<thead>
<tr>
<th>Nrs_id</th>
<th>Nrs_name</th>
<th>Age</th>
<th>Address</th>
<th>MOB</th>
<th>W_Shift</th>
<th>Experience</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shari</td>
<td>30</td>
<td>Mirpur</td>
<td>181556245322</td>
<td>Morning</td>
<td>5</td>
<td>7000</td>
</tr>
<tr>
<td>2</td>
<td>Sma</td>
<td>25</td>
<td>Mirpur</td>
<td>17251478562</td>
<td>Evening</td>
<td>3</td>
<td>6000</td>
</tr>
<tr>
<td>3</td>
<td>Koli</td>
<td>25</td>
<td>Mohammadpur</td>
<td>19152945822</td>
<td>Morning</td>
<td>2</td>
<td>4000</td>
</tr>
<tr>
<td>4</td>
<td>Kekay</td>
<td>22</td>
<td>Mirpur</td>
<td>11919173822</td>
<td>Night</td>
<td>6</td>
<td>7000</td>
</tr>
<tr>
<td>5</td>
<td>Pritty</td>
<td>32</td>
<td>Uttara</td>
<td>181459276215</td>
<td>Night</td>
<td>8</td>
<td>9000</td>
</tr>
</tbody>
</table>
### Nursing service table

<table>
<thead>
<tr>
<th>No</th>
<th>Dp_id</th>
<th>Nrs_id</th>
<th>Room_id</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12/05/09</td>
<td>3:00PM</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>12/05/09</td>
<td>3:00PM</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>12/05/09</td>
<td>6:00PM</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>12/05/09</td>
<td>6:00PM</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>13/05/09</td>
<td>7:00PM</td>
</tr>
</tbody>
</table>

### Ward_boy table

<table>
<thead>
<tr>
<th>Wb_id</th>
<th>Wb_name</th>
<th>MOB</th>
<th>W_shift</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Das</td>
<td>15585658222</td>
<td>Morning</td>
<td>2000</td>
</tr>
<tr>
<td>2</td>
<td>Kader</td>
<td>16752845521</td>
<td>Morning</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>Saber</td>
<td>17869522121</td>
<td>Evening</td>
<td>2000</td>
</tr>
<tr>
<td>4</td>
<td>Salam</td>
<td>19145638452</td>
<td>Evening</td>
<td>2000</td>
</tr>
<tr>
<td>5</td>
<td>Shaf</td>
<td>15525252352</td>
<td>Night</td>
<td>2000</td>
</tr>
</tbody>
</table>

### Cleaning service table

<table>
<thead>
<tr>
<th>Cls_id</th>
<th>Dp_id</th>
<th>Wb_id</th>
<th>Room_id</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12/05/09</td>
<td>6:00PM</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>12/05/09</td>
<td>6:00PM</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>13/05/09</td>
<td>8:00PM</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>14/05/09</td>
<td>8:00PM</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>16/05/09</td>
<td>7:00PM</td>
</tr>
</tbody>
</table>

### Driver table

<table>
<thead>
<tr>
<th>Dri_id</th>
<th>Dri_name</th>
<th>Address</th>
<th>MOB</th>
<th>Shift</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Karm</td>
<td>mrpur</td>
<td>1234</td>
<td>Morning</td>
<td>6000</td>
</tr>
<tr>
<td>2</td>
<td>Kader</td>
<td>mrpur</td>
<td>2345</td>
<td>Evening</td>
<td>6000</td>
</tr>
<tr>
<td>3</td>
<td>Kasen</td>
<td>mrpur</td>
<td>456897</td>
<td>Night</td>
<td>6000</td>
</tr>
<tr>
<td>4</td>
<td>Kanal</td>
<td>rampura</td>
<td>456321</td>
<td>Evening</td>
<td>6000</td>
</tr>
<tr>
<td>5</td>
<td>Dipon</td>
<td>rampura</td>
<td>12345698</td>
<td>Night</td>
<td>6000</td>
</tr>
</tbody>
</table>

57
<table>
<thead>
<tr>
<th>Amb_id</th>
<th>Amb_num</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17-1232</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>17-1233</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>17-1234</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>17-1235</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>17-1235</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>As_id</th>
<th>Pat_id</th>
<th>Dri_id</th>
<th>Amb_id</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12/05/09</td>
<td>11:00AM</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>13/05/09</td>
<td>12:00PM</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>14/05/05</td>
<td>1.00AM</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>15/05/05</td>
<td>12:00AM</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>20/05/09</td>
<td>10:00AM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Crr_id</th>
<th>Cri_name</th>
<th>Mob</th>
<th>Address</th>
<th>Shift</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jobbar</td>
<td>159876252</td>
<td>Mirpur</td>
<td>Morning</td>
<td>5000</td>
</tr>
<tr>
<td>2</td>
<td>Jemal</td>
<td>165874522</td>
<td>Mohammadpur</td>
<td>Morning</td>
<td>5000</td>
</tr>
<tr>
<td>3</td>
<td>Karin</td>
<td>1745698585</td>
<td>Mirpur</td>
<td>Morning</td>
<td>5000</td>
</tr>
<tr>
<td>4</td>
<td>Aliif</td>
<td>181478546</td>
<td>Mohammadpur</td>
<td>Evening</td>
<td>5000</td>
</tr>
<tr>
<td>5</td>
<td>Ask</td>
<td>1754609859</td>
<td>Mirpur</td>
<td>Evening</td>
<td>5000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Crs_id</th>
<th>Crr_id</th>
<th>Amb_id</th>
<th>Pat_id</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>12/05/09</td>
<td>12:00AM</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>12/05/09</td>
<td>12:00AM</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>13/05/09</td>
<td>11:00AM</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>10/05/09</td>
<td>12:00AM</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>10/05/09</td>
<td>12:00AM</td>
</tr>
</tbody>
</table>
2.6 Complex 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 tests are suggested by doctor Selima to which Patients?

Query 1:

```sql
select Pat_name, Doc_name, Test_name from tbl_Patient, tbl_Doctor, tbl_Test, tbl_Assist
where Doc_name = 'Selima' and tbl_Doctor.Doc_id = tbl_Assist.Doc_id and Tbl_Patient.Pat_id = Tbl_Assist.Pat_id and tbl_Test.Test_id = tbl_Assist.Test_id
```

Output:

<table>
<thead>
<tr>
<th>Pat_name</th>
<th>Doc_name</th>
<th>Test_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mori</td>
<td>Selima</td>
<td>Blood</td>
</tr>
<tr>
<td>Karim</td>
<td>Selima</td>
<td>Blood</td>
</tr>
<tr>
<td>Mori</td>
<td>Selima</td>
<td>Enopyram</td>
</tr>
</tbody>
</table>

Question 2

Which doctors prescribed which medicine to patient Mamun?

Query 2:

```sql
select Pat_name, Doc_name, Mdcn_name from tbl_Patient, tbl_Doctor, tbl_Medicine, tbl_Prescription
```
Output 2:

<table>
<thead>
<tr>
<th>Pat_name</th>
<th>Doc_name</th>
<th>Mdn_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mamun</td>
<td>Mohammed</td>
<td>Onidion</td>
</tr>
</tbody>
</table>

Question 3:

Which Doctors are from which Department and they passed from which college and got salaries below 20000 taka?

Query 3

```sql
select Doc_name, Passed_from, Dept_name from tbl_Doctor, tbl_Department, tbl_DFD
where Salary < 20000 and tbl_Doctor.Doc_id = tbl_DFD.Doc_id and tbl_Department.Dept_id = tbl_DFD.Dept_id
```

Output 3:

<table>
<thead>
<tr>
<th>Doc_name</th>
<th>Passed_from</th>
<th>Dept_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kibria</td>
<td>SMC</td>
<td>Cardiology</td>
</tr>
<tr>
<td>Rahima</td>
<td>KMC</td>
<td>Medicine</td>
</tr>
</tbody>
</table>

Question-4

Which doctor conducted the Urine Test for which Patient at 11.00 AM?

Query -4

```sql
select pat_name, doc_name from tbl_Patient, tbl_Doctor, tbl_Test, tbl_Assist
where tbl_Assist.Time = '11.00AM' and tbl_Test.Test_name = 'Urine' and
```
Output 4:-

<table>
<thead>
<tr>
<th>pat_name</th>
<th>doc_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mamun</td>
<td>Mohammed</td>
</tr>
</tbody>
</table>

Question 5:

Which Patient is carried by which driver in Ambulance serial no 5?

Query -5:

```sql
select Pat_name, Dri_name from tbl_Patient,tbl_Driver,tbl_AmbulanceService where Amb_id = 5 and tbl_Patient.Pat_id=tbl_AmbulanceService.Pat_id and tbl_Driver.Dri_id = tbl_AmbulanceService.Dri_id
```

Output 5 :-

<table>
<thead>
<tr>
<th>Pat_name</th>
<th>Dri_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kamal</td>
<td>Dipon</td>
</tr>
</tbody>
</table>

Question 6:

In which time receptionist Rasel appointed patient Kamal to Doctor Selima?

Query 6:

```sql
Select pat_name,Ap_time from tbl_patient,tbl_Receptionist,tbl_Appoinment,tbl_Doctor where Rcp_name='Rasel' and Doc_name='Selima'and pat_name='kamal' and tbl_patient.pat_id=tbl_Appoinment.pat_id and tbl_Receptionist.Rcp_id=tbl_Appoinment.Rcp_id
```

Output 6:

<table>
<thead>
<tr>
<th>pat_name</th>
<th>Ap_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kamal</td>
<td>8.00PM</td>
</tr>
</tbody>
</table>
CHAPTER 3

INTERFACING THE DATABASE SYSTEM USING .NET FRAMEWORK

3.1 Research on Interface Design Guidelines

❖ User Interface

User interface should be designed to match the skills, experience and expectations of its anticipated users. System users often judge a system by its interface rather than its functionality.

❖ Objectives

- To suggest some general design principles for user interface design.
- To explain different interaction styles and their use.
- To explain when to use graphical and textual information presentation.
- To explain the principal activities in the user interface design process.
- To introduce usability attributes and approaches to system evaluation.[8]
User Interface Design Principle

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User familiarity</td>
<td>The interface should use terms and concepts which are drawn from the experience of the people who will make most use of the system.</td>
</tr>
<tr>
<td>Minimal Surprise</td>
<td>Users should never be surprised by the behavior of a system.</td>
</tr>
<tr>
<td>Recoverability</td>
<td>The interface should include mechanisms to allow users to recover from errors.</td>
</tr>
<tr>
<td>User Guidance</td>
<td>The interface should provide meaningful feedback when errors occur and provide context-sensitive user help facilities.</td>
</tr>
<tr>
<td>User diversity</td>
<td>The interface should provide appropriate interaction facilities for different types of system users.</td>
</tr>
</tbody>
</table>

User Interface Design Guidelines

1. Consistency
   - It is known as ("Principle of least astonishment").
   - Certain aspects of an interface should behave in consistent ways at all times for all screens
   - Terminology should be consistent between screens
   - Icons should be consistent between screens
   - Colors should be consistent between screens of similar function.[9]
2. Simplicity

- Break complex tasks into simpler tasks
- Break long sequences into separate steps
- Keep tasks easy by using icons, words etc.
- Use icons/objects that are familiar to the user.[9]

3. Match between system and the real world

- The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms.
- Follow real-world conventions, making information appear in a natural and logical order.[9]

4. Human Memory Limitations

- Organize information into a small number of "chunks"
- Try to create short linear sequences of tasks
- Don't flash important information onto the screen for brief time periods
- Organize data fields to match user expectations, or to organize user input (e.g. auto formatting phone numbers)
- Provide cues/navigation aids for the user to know where they are in the software or at what stage they are in an operation
- Provide reminders, or warnings as appropriate
- Provide ongoing feedback on what is and/or just has happened
- Let users recognize rather than recall information
- Minimize working memory loads by limiting the length of sequences and quantity of information - avoid icon mania![9]
5. Display issues

- Maintain display inertia - make sure the screen changes little from one screen to the next within a functional task situation
- Organize screen complexity
- Eliminate unnecessary information
- Use concise, unambiguous wording for instructions and messages
- Use easy to recognize icons
- Use a balanced screen layout - don't put too much information at the top of the screen - try to balance information in each screen quadrant
- Use plenty of 'white space' around text blocks - use at least 50% white space for text screens
- Group information logically
- Structure the information rather than just presenting a narrative format (comprehension can be 40% faster for a structured format).[9]

6. Error prevention

- Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
- Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.[9]

7. Help and documentation:

- Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation.
o Any such information should be easy to search, focused on the user’s task, list concrete steps to be carried out, and not be too large.[9]

8. System messages:

o Provide user-centered wording in messages (e.g. "there was a problem in copying the file to your disk" rather than "execution error 159")

o Avoid ambiguous messages (e.g. hit 'any' key to continue - there is no 'any' key and there’s no need to hit a key, reword to say 'press the return key to continue')

o Avoid using threatening or alarming messages (e.g. fatal error, run aborted, kill job, catastrophic error)

o Use specific, constructive words in error messages (e.g. avoid general messages such as 'invalid entry' and use specific phrases such as 'please enter your name')

o Make the system 'take the blame' for errors (e.g. "illegal command" versus "unrecognized command").[9]

9. Attention

o Use attention grabbing techniques cautiously (e.g. avoid overusing 'blinks' on web pages, flashing messages, bold colors etc.)

o Don’t use more than 4 different font sizes per screen

o Use serif or sans serif fonts appropriately as the visual task situation demands.

o Don’t use all uppercase letters - use and uppercase/lowercase mix

o Don’t overuse audio or video

o Use colors appropriately and make use of expectations (e.g. don’t have an OK button colored red! use green for OK, yellow for ‘caution, and red for ‘danger’ or ‘stop’)

66
o Don’t use more than 4 different colors on a screen
o Don’t use blue for text (hard to read), blue is a good background color.

o Don’t put red text on a blue background
o Use high contrast color combinations
o Use colors consistently
o Use only 2 levels of intensity on a single screen
o On text screens don’t use more than 3 fonts on a single screen. [9]

10. Anthropomorphization

o Don’t anthropomorphize (i.e. don’t attribute human characteristics to objects) - avoid the "Have a nice day" messages from your computer. [9]

11. Choose specific fonts, font sizes and font characteristics to represent certain types of information

With the proliferation of high resolution display devices, designers no longer need to be as concerned about the technical problems associated with what types of fonts and font characteristics are used on the monitor. Using a particular font in a particular location or for a particular portion of a program can aid users when searching for screens that contain the type of information they are searching for. Font characteristics such as bold, italic, and underlining can be used to designate key words that are hot or active. [10]
12. Provide selectable areas to allow users to access information

Some possible selectable areas to consider are buttons and hot text within a text field. The location of these elements on the screen will depend on the available screen real estate and the function of the selectable areas. It is recommended that the placement of selectable areas be tested with users to find out what is the optimal location for them. The selectable area will be a control element for users to access information. The control chosen will depend on the task to be done. Be consistent in implementing particular controls for particular functions. [10-15]

13. Provide visual effects to give users visual feedback that their choices have been made and registered by the program

Buttons, icons, and menus can be highlighted or animated to show users that a choice has been made. Keep the highlighting or animation simple. The duration of a highlight or animation should be long enough to be registered visually by the users, but short enough so that users are not waiting for an animation to be over so that they can get to the information they want.

Visual effects, such as wipes, fades, and zooms may be used to indicate access to a particular piece of information. The use of these visual effects should be consistent. Do not use them simply because they are available, but rather use them to indicate a particular action of the program. Additionally, be consistent in the use of a visual effect. If wipes are used when clicking on a right arrow, use them throughout the program. If zoom outs are used when clicking on a menu item, then use zoom INS when returning to the menu. Above all, make the visual effect have meaning and be consistent with its use throughout the program. [10-15]
Human Factors in Interface Design

- **Limited Short-term memory**
  - People can instantaneously remember about 7 items of information. If you present more than this, they are more liable to make mistakes.

- **People make mistakes**
  - When people makes mistakes and systems go wrong, inappropriate alarms and messages can increase stress and hence the likelihood of more mistakes.

- **People are different**
  - People have a wide range of physical capabilities. Designers should not just design for their own capabilities.

- **People have different interaction preferences**
  - Some people like picture and some like text. [16]

Sample of Interfaces
Here we show some samples of Interfaces:

SAMPLE-1

DOS-Based Q&A[16]

SAMPLE-2

Begin Search Form Created with MS Access [16]

*Tippecanoe County Probation Department* has made DOS Based Q&A and A Search Form using Access as shown in Sample-1 and Sample-2.
SAMPLE-3:

Connection Dialog Box using VB.Net[16]

#Usability Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learnability</td>
<td>How long does it take a new user to become productive with the system?</td>
</tr>
<tr>
<td>Speed of operation</td>
<td>How well does the system response match the user’s work practice?</td>
</tr>
<tr>
<td>Robustness</td>
<td>How tolerant is the system of user error?</td>
</tr>
<tr>
<td>Recoverability</td>
<td>How good is the system at recovering from user errors?</td>
</tr>
<tr>
<td>Adaptability</td>
<td>How closely is the system tied to a single model of work?</td>
</tr>
</tbody>
</table>

Source: [16]
Summary

We can say that we have to design interfaces clearly and efficiently according to the user choice. A poorly designed interface can cause a user to make catastrophic errors. Poor user interface design is the reason why so many software systems are never used.

3.2 FRONT END Design

Introduction:

Front end and Back End are generalized terms that refer to the initial and the end stages of a process. The front end is responsible for collecting input in various forms from the user and processing it to conform to a specification the back end can use. The front end is an interface between the user and the back end.

- The separation of software systems into front and back ends simplifies development and separates maintenance.
- For major computer subsystems, a graphical file manager is a front end to the computer’s file system. The front end faces the user and the backend launches the programs of the operating system in response.[17]
We have completed the backend design using SQL Server and now we have designed the front end using .NET Framework/(C#).

3.2.1 FORMS DESIGN:

Front end Forms Design includes

- Login Form
- Form Menu
- Admin Part

- Accountant Form
- Receptionist Form
- Nurse Form
- Room Form
- Ward boy Form
- Ambulance Form
- Carrier Form
- Driver Form
- Bill Form
- Admission Form
- Appointment Form
- Ambulance Service Form
- Carrying Service Form
- Nursing Service Form
- Cleaning Service Form
- Payment Form
□ Medical Part

✓ Patient Form
✓ Doctor Form
✓ Department Form
✓ Medicine Form
✓ Test Form
✓ Operation Theater Form
✓ Doctor’s from Department Form
✓ Prescription Form
✓ Assist Form
✓ Operation Form

□ Search Option

□ Login Form:

This form comes at the very beginning of the software:
Fig: Login page

When Designation and password will match we can switch to the Form Menu.

Form Menu:
In this form we can see a menu strip and there are many menu options like Entry, Search, View, Tools, Windows, Help and other icons.

Example: In Patient Form which comes under Medical Part of Entry menu bar, we can enter the new patient data.

In Search option under Search menu bar we can retrieve information of different tables of our choice according to Search criteria.

**ADMINISTRATION PART**

The way we enter data in the administration forms is given below.
Room Form:

![Image of a hospital management software interface with a dropdown menu for different departments and a table showing room details]
Bill Form:

Accountant Form:
Receptionist Form:

Driver Form
Ambulance Form
Carriers Form

Nurse Form
Ward boy Form

Admission Form
Appointment Form

Ambulance Service Form
Nursing Service Form

Cleaning Service Form
Carrying Service Form

Payment Form
MEDICAL PART

The way we enter data in the Medical forms is given below.
OT Form

Medicine Form
Department Form

Test Form
Patient Form
DFD Form

![DFD Form Diagram]

Prescription Form

![Prescription Form Diagram]

Assist Form

![Assist Form Diagram]
Operation Form

Search Form

We can search the data in the way given below
In this form we can search different information of our software according to search criteria.
3.2.2 Relating Interface Design Guidelines to our Front end Design:

In our front end we refer to the User Interface Guidelines that we researched

1. Match between system and the real world

   o The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms.[9]

   ➔ In our system we tried to make it more users friendly and familiar to the user, so that it should speak the user’s language.

   o Follow real-world conventions, making information appear in a natural and logical order.[9]

   ➔ To match between real world and the system we tried to arrange all the information of our system appears in a natural and logical order.

2. Help and Documentation

   o Any such information should be easy to search, focused on the user’s task, list concrete steps to be carried out, and not be too large.[9]

   ➔ To follow this guideline we tried to make our information list small and easy to search.
3. Attention
   o Don't use more than 4 different font sizes per screen.[9]
      ➤ In our front ends we use a single font (Comic Sans MS) in 4 Different sizes.
   o Don't use all uppercase letters - use and uppercase/lowercase mix.[9]
      ➤ If we use all uppercase or lowercase letters it is not so comfortably visible for users. That's why we have followed the instructions and mixed the upper and lower cases.
   o Don't overuse audio or video.
      ➤ We do not use any audios or videos in the forms.
   o Use colors appropriately and make use of expectations (e.g. don't have an OK button colored red! use green for OK, yellow for 'caution, and red for 'danger' or 'stop').
      ➤ We do not use buttons in red color in our front end, we use system color in the button and it looks good with the background color.
   o Don't use more than 4 different colors on a screen.
      ➤ We use two different colors in our front ends. The colors are Lavender (as background), Linen (in the Groupbox).
o Don't use blue for text (hard to read), blue is a good background color.

  ➔ We use blue as our front end background and black as text color.

o Don't put red text on a blue background.

  ➔ Red is not matchable on a blue background. To follow the guidelines we don't use it in our system.

o Use italic, underlining, bold, inverse video or other markers sparingly.

  ➔ We use italic and bold in our form texts.

o Use colors consistently.[9]

  ➔ We tried to use colors consistently.

Summary

We try our level best to follow the guidelines which were very helpful for us in our form design. We hope a user friendly and efficient interface has been developed.

3.3 Security feature of FRONT END
Security has to be compared to related concepts: Safety, continuity, reliability. The key difference between security and reliability is that security must take into account the actions of people attempting to cause destruction.

Here, we discuss about security for any Computer Software System. To start this topic we must have to know about Computer system security and Database Security. [26]

**Computer System Security**

- The term computer system security means the collective processes and mechanisms by which sensitive and valuable information and services are protected from publication, tampering or collapse by unauthorized activities or untrustworthy individuals and unplanned events respectively.

- Computer security is critical in almost any technology-driven industry which operates on computer systems. Computer security can also be referred to as computer safety. [19]

- Database security includes the system, processes, and procedures that protect a database from unintended activity.

- Data security is the means of ensuring that data is kept safe from corruption and that access to it is suitably controlled.

  - Data security helps to ensure privacy.
  - Helps in protecting personal data.

To control and work with the Database Security we need an administrator. [20]
Features of Database Administrator:-

- Database administrators work with database management software and determine ways to store, organize, analyze, use, and present data.
- Identify user needs and set up new computer databases. Database administrators must integrate data from old systems into a new system.
- Test and coordinate modifications to the system when needed. [21]

An organization's database administrator ensures the performance of the system, understands the platform on which the database runs, and adds new users to the system.

Our Software is about BIRDEM Hospital Management System. After comprehending the importance of security we try to secure our system from any type of unintended activity.

In our security panel there are 3 types of members.

1. Administrator,
2. Receptionist,
3. Accountant.
Fig: Login page of our software.

Fig: When we run our software we can see 3 options
Fig: Administrator option is selected and password is entered

Fig: The password is matched
Fig: When the password is matched we can switch to the Form Menu.

Fig: Now the Accountant option is selected and password is entered.
Fig: The password is matched
Fig: When the password is matched we can switch to the Form Menu

Fig: Now Receptionist option is selected and password is entered
Fig: The password Matched.

Fig: When the password is matched we can switch to the Form Menu.
Fig: If the password does not match Error Message is showed.

- **Security Code:**

```csharp
int password = Convert.ToInt32(passwordText.Text);

if (DesignationText.Text == "Administrator" && password == 62413)
{
    MessageBox.Show("Password Matched");
    frmMenubar f = new frmMenubar();
    f.Show();
}

else if (DesignationText.Text == "Accountant" && password == 62436)
{
    MessageBox.Show("Password Matched");
    frmMenubar f = new frmMenubar();
    f.Show();
}
```
else if (DesignationText.Text == "Receptionist" && password == 62444)
{
    MessageBox.Show("Password Matched");
    frmMenubar f = new frmMenubar();
    f.Show();
}
else
{
    MessageBox.Show("Invalid Password");
}

We design the security part of our Software by following a Point which is taken from the User Guidelines Interfaces. It is very much helpful for us to think and design the interface of our software in this respect. The point is given below:-

- Provide selectable areas to allow users to access information

  - Some possible selectable areas to consider are buttons and hot text within a text field. The location of these
elements on the screen will depend on the available screen real estate and the function of the selectable areas.

✓ Here we use the 'Login' and 'Cancel' as a 'Button' and also use group box, where we include combo box, textbox and label.

➢ It is recommended that the placement of selectable areas be tested with users to find out what is the optimal location for them.

➢ The selectable area will be a control element for users to access information. The control chosen will depend on the task to be done. Be consistent in implementing particular controls for particular functions. [10-15]

To control and work with the Database Security we do some tasks which are given below ->

✓ Determine ways to store, organize, analyze, use, and present data.
✓ Identify user needs and set up new computer databases,
✓ Ensure privacy, to protect personal data by Testing and coordinate modifications to the system when we need so.

❖ Summary:
Security is very important in software development. We apply security in our software so that any user cannot access the information, entered by the input users. We control the security from the front end. It works efficiently.

### 3.4 Implementation of Insert, Delete, Update buttons and Search Option

In our software save, delete and update buttons are very common features and search option is a special feature. These buttons carry out the actions as their names imply, Search option helps to search info according to selections of id and table name.
Login Form

Action:

✓ When the Designation and Password matched we can go to the next step “Form Menu”.

® Refer to the codes and Description in the Appendix.

Form Menu

Action:

✓ We can easily switch to the different forms of our software.

® Refer to the codes and Description in the Appendix.

Administration part:

a. Room Form
b. Bill Form
c. Accountant Form
d. Receptionist Form
e. Driver Form
f. Ambulance Form
g. Carriers Form
h. Nurse Form
i. WardBoy Form
j. Admission Service Form
k. Appointment Service Form
l. Ambulance Service Form
m. Nursing Service Form
n. Cleaning Service Form
o. Carrying Service Form
p. Payment Form
**Medical Part:**

i. Doctor Form  
ii. OT Form  
iii. Medicine Form  
iv. Department Form  
v. Test Form  
vi. Patient Form  
vii. DFD Form  
viii. Prescription Form  
ix. Assist Form  
x. Operation Form

Save, Delete, Update and Search codes are similar for all the forms. So we are describing the codes of the Room form as an example.

**Save Action for Room Form**

**Code**

```csharp
private void btSave_Click(object sender, EventArgs e)
{
    if (!Validation()) return;
    SetRoomInstant();
    roomInstant.Save();
    dataGridView1.DataSource = roomInstant.GetAllData();
    ClearTextBox();
    ButtonControl(false);
}
```

When Save button is clicked these codes are executed.

We can see three functions
a. Validation()

This function checks all the insert data in the form is valid or not.

```csharp
private bool Validation()
{
    if (textBox1.Text == "")
    {
        MASICEIU.MessageShow.Information("Select Item from room list.");
        return false;
    }
    else if (textBox2.Text == "")
    {
        MASICEIU.MessageShow.Information("Room No");
        textBox2.Focus();
        return false;
    }
    else if (textBox3.Text == "")
    {
        MASICEIU.MessageShow.Information("Room Type");
        textBox3.Focus();
        return false;
    }
    else if (textBox4.Text == "")
    {
        MASICEIU.MessageShow.Information("Room Cost");
        textBox4.Focus();
        return false;
    }
    return true;
}
```

b. SetRoomInstant();

This function sets instances and convert variables to string if necessary.

```csharp
private void SetRoomInstant()
{
    roomInstant.Room_id1 = Convert.ToInt16(textBox1.Text);
    roomInstant.Room_no1 = textBox2.Text;
    roomInstant.Room_type1 = textBox3.Text;
    roomInstant.Room_cost1 = Convert.ToInt16(textBox4.Text);
}
```

c. ClearTextBox()

This function clears all the textbox of the form after Save button is clicked.
private void ClearTextBox()
{
    textBox1.Text = "";
    textBox2.Text = "";
    textBox3.Text = "";
    textBox4.Text = "";
}

Delete Action for Room Form

Code
private void btDelete_Click(object sender, EventArgs e)
{
    if (!Validation()) return;
    SetRoomInstant();
    roomInstant.Delete();
    dataGridView1.DataSource = roomInstant.GetAllData();
    ClearTextBox();
}

We can also see three functions

a. Validation()
b. SetRoomInstant();
c. ClearTextBox()

* The descriptions of these functions have been described earlier.

Update Action for Room Form

Code
private void btUpdate_Click(object sender, EventArgs e)
{
    if (!Validation()) return;
    SetRoomInstant();
    roomInstant.Update();
    dataGridView1.DataSource = roomInstant.GetAllData();
    ClearTextBox();
}
We can also see three functions

a. Validation()

b. SetRoomInstant();

c. ClearTextBox()

The descriptions of these functions have been described earlier.

Search Action

In the search form combobox2 we can select a form’s data grid view as shown as page 104. Then we can search id from the combobox1 as shown as page 104. Accordingly single row is displayed. The code is given below:

Code

```csharp
private void Search_Click(object sender, EventArgs e)
{
    if (comboBox2.Text != "" && comboBox1.SelectedIndex > -1)
    {
        dataGridView1.DataSource =
        CommonDataAccess.GetData(comboBox1.Text, comboBox2.Text);
    }
}
```

In order to do the Save, Delete, Update and Search we use 3 helping files

- RoomDataAccess.cs
- RoomDataObject.cs
- RoomService.cs

The description of these classes and codes are described in the Appendix part.
3.5 Usage of DLL file

❖ DLL File

Dynamic-link library (also written without the hyphen), or DLL, is Microsoft's implementation of the shared library concept in the Microsoft Windows and OS/2 operating systems. These libraries usually have the file extension DLL, OCX (for libraries containing ActiveX controls), or DRV (for legacy system drivers). The file formats for DLLs are the same as for Windows EXE files — that is, Portable Executable (PE) for 32-bit and 64-bit Windows, and New Executable (NE) for 16-bit Windows. As with EXEs, DLLs can contain code, data, and resources, in any combination. [22]
Fig: the DLL file used in our Software

Source: [22]

- **The purpose of using the DLL file**
  - Using DLL file we can easily carry our database with our software.
  - We don't need to load the database first.
  - The software becomes more efficient and user friendly.
  - After using DLL file we do not need to load the database to interface with the front end in different PC's.
CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

By the glance of Allah, the Almighty we have come to the end of our thesis report. It is not the work of one day. In fact it took us a year to complete. The group members worked hard to make it a good and improvised thesis.

Summing up, we worked on a case study of BIRDEM Hospital Management, designing and storing its information in a sample database of our creation. We designed ER models, Relational Models and Normalized tables of the relational model and finally implemented the SQL Server Diagram, filled the server tables with data values and queried different useful information from the database.

The second part of the thesis involved developing a user friendly and efficient interface to the backend database in SQL Server. We researched User Interface Guidelines and applied some of those to our front end forms design. We have taken into account issues of security too.

4.2 Future Work

While an efficient user friendly interface to SQL-based backend database has been successfully developed, we have in mind some scope for future work involving Guideline View Features and Trigger Features. These are explained as follows.
4.2.1 Data GridView:

The DataGridView control provides a customizable table for displaying data. The DataGridView class allows customization of cells, rows, columns, and borders through the use of properties such as DefaultCellStyle, ColumnHeadersDefaultCellStyle, CellBorderStyle, and GridColor.

We can use a DataGridView control to display data with or without an underlying data source. Without specifying a data source, we can create columns and rows that contain data and add them directly to the DataGridView using the Rows and Columns properties. You can also use the Rows collection to access DataGridViewRow objects and the DataGridViewRow.Cells property to read or write cell values directly. The Item indexer also provides direct access to cells.

As an alternative to populating the control manually, we can set the DataSource andDataMember properties to bind the DataGridView to a data source and automatically populate it with data.

When working with very large amounts of data, you can set the VirtualMode property to true to display a subset of the available data. Virtual mode requires the implementation of a data cache from which the DataGridView control is populated. [23]
Use DataGridView in .NET FRAMEWORK

1. Retrieve Data from the Database:

![Fig: When A DataGridView is loaded in a form.](image)
Fig: When we run the form, GridView retrieves data from the database.

Here we can see Operation Theater Information where Ot_id and Ot_room_no are the information. We manage to add two more columns named Edit and Delete. Edit Column contain Edit Button and Delete Column Contains Delete Button.
See the recently entered data:

![Figure: Inserting New Data in the form and clicking the Save Button.](image-url)
Fig: Confirmation of Data Insertion in the Database

Fig: Recently inserted data is seen in the DataGridView.
Delete Data from the DataGridView:

Fig: A Column is selected to Delete.

Fig: A Message Box is Displayed for the User Confirmation
Fig: Confirmation that the information or data is deleted successfully.

Fig: The Picture of Grid View after the data is deleted.
Edit and Update Data from the DataGridView:

Fig: When the Edit Button is clicked the data is seen in the form. The Save Button is changed to Update Button.
Fig: After Edit the data *Update* button is clicked.

Fig: Confirmation that the data is updated.
Fig: The Picture of the DataGridView after data update.

#Codes for Data Gridline View:

**Code_OT Class:**

```csharp
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace OperationTheature
{
    public partial class OT : Form
    {
        public OT()
        {
            InitializeComponent();
        }
    }
}
```
public int otid;
OperationBasic ob = new OperationBasic();
OTGateway og = new OTGateway();
OTManager om = new OTManager();
bool isTrue = false;
public string msg = null;

private void OT_Load(object sender, EventArgs e)
{
	this.tbl_OTTableAdapter1.Fill(this.db_PatientDataSet1.tbl_OT);
	this.AddColumns();
}

private void LoadInitializes()
{
	OTManager om = new OTManager();
	OTDataGridView.DataSource = om.ShowOperation();
}
private void AddColumns()
{
	DataGridViewButtonColumn EditCol = new DataGridViewButtonColumn();
	EditCol.Name = "Edit";
	EditCol.Text = "Edit";
	EditCol.UseColumnTextForButtonValue = true;
	this.OTDataGridView.Columns.Add(EditCol);
	DataGridViewButtonColumn DeleteCol = new DataGridViewButtonColumn();
	DeleteCol.Name = "Delete";
	DeleteCol.Text = "Delete";
	DeleteCol.UseColumnTextForButtonValue = true;
	this.OTDataGridView.Columns.Add(DeleteCol);
}

public void Clear()
{
	operationidText.Text = null;
	otroomnoText.Text = null;
	this.Save_Button.Text = "Save";
}

private void Save_Button_Click(object sender, EventArgs e)
{
	OTManager om = new OTManager();
	OperationBasic ob = new OperationBasic();
ob.ot_id = Convert.ToInt32(operationidText.Text);
ob.ot_room_no = otroomnoText.Text;

if (this.otid == 0)
{
    msg = om.SaveOperation(ob);
    MessageBox.Show(msg);
    this.OTDataGridView.Columns.Remove("Edit");
    this.OTDataGridView.Columns.Remove("Delete");
}
else
{
    OTGateway og = new OTGateway();
    og.UpdateOperation(ob);
    MessageBox.Show("Updated Successfully");
    this.OTDataGridView.Columns.Remove("Edit");
    this.OTDataGridView.Columns.Remove("Delete");
}
Clear();
OT_Load(null, null);

private void DeleteButton_Click(object sender, EventArgs e)
{
    ob = new OperationBasic();
    om = new OTManager();
    int operationcode = ob.ot_id = Convert.ToInt32(OTDataGridView.Rows[OTDataGridView.SelectedCells[0].RowIndex].Cells["Ot_id"].Value.ToString());

    msg = om.DeleteOperation(operationcode);
    Close();
    MessageBox.Show(msg);
}

private void OTDataGridView_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.ColumnIndex == this.OTDataGridView.Columns["Edit"].Index)
    {
        EditAction(e);
    }
    else if (e.ColumnIndex == this.OTDataGridView.Columns["Delete"].Index)
    {
        DeleteAction(e);
    }
private void EditAction(DataGridViewCellEventArgs e)
{
    otid = int.Parse(this.OTDataGridView.Rows[e.RowIndex].Cells[0].Value.ToString());
    this.operationidText.Text = otid.ToString();
    this.otroomnoText.Text = this.OTDataGridView.Rows[e.RowIndex].Cells[1].Value.ToString();
    this.Save_Button.Text = "Update";
}

private void DeleteAction(DataGridViewCellEventArgs e)
{
    if (MessageBox.Show("Are you sure want to Delete?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
    {
        otid = int.Parse(this.OTDataGridView.Rows[e.RowIndex].Cells[0].Value.ToString());
        og.DeleteOperation(otid);
        MessageBox.Show("Information Deleted");
        this.OTDataGridView.Columns.Remove("Edit");
        this.OTDataGridView.Columns.Remove("Delete");
        OT_Load(null, null);
    }
}

Difficulties:

- This process works well but some times changes of commands make forms disable and invalid.
- If we give more time and afford we can complete the software using Grid view control in the future.
4.2.2 TRIGGER Features:

A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for keeping the integrity of the information on the database. For example, when a new record (representing a new worker) is added to the employees table, new records should be created also in the tables of the taxes, vacations, and salaries.

We can write triggers that fire whenever one of the following operations occurs:

1. DML statements (INSERT, UPDATE, DELETE) on a particular table or view, issued by any user.
2. DDL statements (CREATE or ALTER primarily) issued either by a particular schema/user or by any schema/user in the database.
3. Database events, such as logon/logoff, errors, or startup/shutdown, also issued either by a particular schema/user or by any schema/user in the database.

Triggers are similar to stored procedures. A trigger stored in the database can include SQL and PL/SQL or Java statements to run as a unit and can invoke stored procedures. However, procedures and triggers differ in the way that they are invoked. A procedure is explicitly run by a user, application, or trigger. Triggers are implicitly fired by Oracle SQL server when a triggering event occurs, no matter which user is connected or which application is being used. [24, 25]
Fig: Triggers

Trigger Structure:
Sample Code:

```
create trigger overdraft-trigger after update on account
referencing new row as nrow
for each row
when nrow.balance < 0
begin atomic
    insert into borrower
    (select customer-name, account-number
    from depositor
    where nrow.account-number =
        depositor.account-number);
    insert into loan values
    (n.row.account-number, nrow.branch-name,
    - nrow.balance);
    update account set balance = 0
    where account.account-number = nrow.account-number
end
```

Source: [25]

Applying Triggers in our Database:

The trigger we may apply in our database is similar for all tables. So trigger applied on Room Table and Admission Table can be given as an example:

```
Create or replace trigger Admission after insert on Room
for each row
begin
    insert into Admission
    (select * from Admission
```
where Admission.Room_no=:new.Room_no);
end;
Source: [25]

**Difficulties:**

- The triggers are created but when we insert values it does not work properly.
- This is left as a part of future work.

**Summary**

We can say that Data Grid view is very essential in .NET Framework. We can do a lot of things easily and efficiently using Data Grid view. Though the coding is not so easy but it will help us to make user friendly software. On the other hand trigger is a very essential approach in database. We can make a database for functional and efficient using Triggers.
REFERENCES

7. Prof. Dorothee Koch, Lecture Notes : HfT Stuttg­gart-Normalaization.e.fm
   McGRAW-HILL INTERNATIONAL EDITION by ROGER S. PRESSMAN
   (Chapter-26)
don't? In B. Laurel (Ed.), the art of human-computer interface design. (pp. 113-
    of human-computer interface design. (pp. 113-123). Maidenhead Birkshire:
Pergamon Infotech Limited.


16. Presentation Slide (Judical Database System) by Kipp Scott and Michael Sinks


25. http://download.oracle.com/docs/cd/B19306_01/server.102/b14220/triggers.htm#CNCPT017
APPENDIX

Login Form:

Code for Login Form:

```csharp
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using WindowsFormsBirdem.UI;

namespace WindowsFormsBirdem
{
    public partial class LOGIN : Form
    {
        public void dis()
        {
            LOGIN l = new LOGIN();
            l.WindowState = FormWindowState.Maximized;
        }

        public enum Designation
        {
            Administrator,
            Accountant,
            Receptionist
        }

        public LOGIN()
        {
            InitializeComponent();
            this.DesignationCombo.DataSource = Enum.GetNames(typeof(Designation));
        }

        private void btAddNew_Click(object sender, EventArgs e)
        {
            int password = Convert.ToInt32(passwordText.Text);

            if (DesignationText.Text == "Administrator" && password == 62413)
            {
                MessageBox.Show("Password Matched");
                frmMenubar f = new frmMenubar();
                f.Show();
            }
        }
    }
}
```
else if (DesignationText.Text == "Accountant" && password == 62436) {
    MessageBox.Show("Password Matched");
    frmMenubar f = new frmMenubar();
    f.Show();
}
else if (DesignationText.Text == "Receptionist" && password == 62444) {
    MessageBox.Show("Password Matched");
    frmMenubar f = new frmMenubar();
    f.Show();
}
else {
    MessageBox.Show("Invalid Password");
}

//this.Close();

private void DesignationCombo_SelectedIndexChanged(object sender, EventArgs e) {
    Designation des = (Designation)Enum.Parse(typeof(Designation), DesignationCombo.Text);
    switch (des) {
    case Designation.Administrator:
        DesignationText.Text = "Administrator";
        break;
    case Designation.Accountant:
        DesignationText.Text = "Accountant";
        break;
    case Designation.Receptionist:
        DesignationText.Text = "Receptionist";
        break;
    }
}
}
Form Menu:

Codes for the Form Menu:

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Forms;

namespace WindowsFormsbirdem.UI
{
    public partial class frmMenubar : Form
    {
        private int childFormNumber = 0;

        public frmMenubar()
        {
            InitializeComponent();
        }

        private void ShowNewForm(object sender, EventArgs e)
        {
            Form childForm = new Form();
            childForm.MdiParent = this;
            childForm.Text = "Window " + childFormNumber++;
            childForm.Show();
        }

        private void OpenFile(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            openFileDialog.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*";
            if (openFileDialog.ShowDialog(this) == DialogResult.OK)
            {
                string FileName = openFileDialog.FileName;
            }
        }

        private void SaveAsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            saveFileDialog.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*";
            if (saveFileDialog.ShowDialog(this) == DialogResult.OK)
            {
                string FileName = saveFileDialog.FileName;
            }
        }
    }
}
private void ExitToolsStripMenuItem_Click(object sender, EventArgs e)
{
    this.Close();
}
private void ToolBarToolStripMenuItem_Click(object sender, EventArgs e)
{
    toolStrip.Visible = toolBarToolStripMenuItem.Checked;
}
private void StatusBarToolStripMenuItem_Click(object sender, EventArgs e)
{
    statusStrip.Visible = statusBarToolStripMenuItem.Checked;
}
private void CascadeToolStripMenuItem_Click(object sender, EventArgs e)
{
    LayoutMdi(MdiLayout.Cascade);
}
private void TileVerticalToolStripMenuItem_Click(object sender, EventArgs e)
{
    LayoutMdi(MdiLayout.TileVertical);
}
private void TileHorizontalToolStripMenuItem_Click(object sender, EventArgs e)
{
    LayoutMdi(MdiLayout.TileHorizontal);
}
private void ArrangeIconsToolStripMenuItem_Click(object sender, EventArgs e)
{
    LayoutMdi(MdiLayout.ArrangeIcons);
}
private void CloseAllToolStripMenuItem_Click(object sender, EventArgs e)
{
    CloseAllChildForm();
}
private void CloseAllChildForm()
{
    foreach (Form childForm in MdiChildren)
    {
        childForm.Close();
    }
}
private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{
private void roomToolStripMenuItem_Click(object sender, EventArgs e)
{
    Show(new frmRoom());
}

private void Show(Form frm)
{
    CloseAllChildForm();
    frm.WindowState = FormWindowState.Maximized;
    frm.MdiParent = this;
    frm.Show();
}

private void roomToolStripMenuItem1_Click(object sender, EventArgs e)
{
    Show(new Search_info());
}

private void billToolStripMenuItem_Click(object sender, EventArgs e)
{
    Show(new frmBill());
}

private void driverToolStripMenuItem_Click(object sender, EventArgs e)
{
    Driver d = new Driver();
    d.Show();
}

private void frmMenubar_Load(object sender, EventArgs e)
{
    BackColor = Color.Lavender;
    Show(new Form1(this.menuStrip, this.toolStrip));
    BackColor = Color.Lavender;
}

private void accountantToolStripMenuItem_Click(object sender, EventArgs e)
{
    Show(new frmAccountant());
}

private void receptionistToolStripMenuItem_Click(object sender, EventArgs e)
{
    Show(new frmReceptionist());
}

private void ambulanceToolStripMenuItem_Click(object sender, EventArgs e)
{
private void carriersToolStripMenuItem_Click(object sender, EventArgs e) {
    Show(new frmCarriers());
}

private void nurseToolStripMenuItem_Click(object sender, EventArgs e) {
    Show(new frmNurse());
}

private void wardboyToolStripMenuItem_Click(object sender, EventArgs e) {
    Show(new frmWardboy());
}

private void doctorToolStripMenuItem1_Click(object sender, EventArgs e) {
    Show(new frmDoctor());
}

private void oTToolStripMenuItem_Click(object sender, EventArgs e) {
    Show(new frmOT());
}

private void medicineToolStripMenuItem_Click(object sender, EventArgs e) {
    Show(new frmMedicine());
}

private void departmentToolStripMenuItem_Click(object sender, EventArgs e) {
    Show(new frmDepartment());
}

private void testToolStripMenuItem_Click(object sender, EventArgs e) {
    Show(new frmTest());
}

private void patientToolStripMenuItem_Click(object sender, EventArgs e) {
    Show(new frmPatient());
}

private void doctorToolStripMenuItem_Click(object sender, EventArgs e) {
}
private void dFDToolStripMenuItem_Click(object sender, EventArgs e)
{
    Show(new frmDFD());
}

Room Form:

Codes for the Room Form:

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using WindowsFormsbirdem.DAL;

namespace WindowsFormsbirdem.UI
{
    public partial class frmRoom : Form
    {
        public static RoomDataObject roomInstant = new RoomDataObject();
        public frmRoom()
        {
            InitializeComponent();
            dataGridView1.DataSource = roomInstant.GetData();
        }
        private void toolStripButton4_Click(object sender, EventArgs e)
        {
            ClearTextBox();
            ButtonControl(false);
        }
        private void dataGridView1_MouseDoubleClick(object sender, MouseEventArgs e)
        {
            textBox1.Text = dataGridView1.Rows[dataGridView1.SelectedCells[0].RowIndex].Cells[0].Value.ToString();
            ButtonControl(false);
        }
        private bool Validation()
        {
            if (textBox1.Text == "")
            {
                return false;
            }
            return true;
        }
    }
}
{ MASICEIU.MessageShow.Information("Select Item from room list.");
    return false;
}
else if (textBox2.Text == "")
{
    MASICEIU.MessageShow.Information("Room No");
    textBox2.Focus();
    return false;
}
else if (textBox3.Text == "")
{
    MASICEIU.MessageShow.Information("Room Type");
    textBox3.Focus();
    return false;
}
else if (textBox4.Text == "")
{
    MASICEIU.MessageShow.Information("Room Cost");
    textBox4.Focus();
    return false;
}
return true;

private void ButtonControl(bool boolValue)
{
    btSave.Enabled = boolValue;
    btUpdate.Enabled = !boolValue;
    btDelete.Enabled = !boolValue;
    btAddNew.Enabled = !boolValue;
}
private void ClearTextBox()
{
    textBox1.Text = "";
    textBox2.Text = "";
    textBox3.Text = "";
    textBox4.Text = "";
}
private void SetRoomInstant()
{
    roomInstant.Room_id1 = Convert.ToInt16(textBox1.Text);
    roomInstant.Room_no1 = textBox2.Text;
    roomInstant.Room_type1 = textBox3.Text;
    roomInstant.Room_cost1 = Convert.ToInt16(textBox4.Text);
}
private void toolStripButton5_Click(object sender, EventArgs e)
{
    textBox1.Text = new RoomDataAccess().NextID("Select max(Room_id) from tbl_Room").ToString();
    ButtonControl(true);
}
private void toolStripButton1_Click(object sender, EventArgs e)
{
    if (!Validation()) return;
    SetRoomInstant();
    roomInstant.Save();
    dataGridView1.DataSource = roomInstant.GetAllData();
    ClearTextBox();
    ButtonControl(false);
Actions:

- **Save, Delete, Update, Addnew buttons are controlled in this form.**
- **DataGridView is controlled from this form.**
- **Helping file RoomDataObject is called from this class.**

**Code of RoomDataObject.cs class**

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MASICEIU.BaseDataLayer;
using WindowsFormsbirdem.DAL.DOL;

namespace WindowsFormsbirdem.DAL
{
    // Code here
}
```
public class RoomDataObject : DataObject
{
    RoomService service = new RoomService();
    private int Room_id;
    private string Room_no;
    private string Room_type;
    private int Room_cost;

    public int Room_id1 { get { return Room_id; } set { Room_id = value; } }
    public string Room_no1 { get { return Room_no; } set { Room_no = value; } }
    public string Room_type1 { get { return Room_type; } set { Room_type = value; } }
    public int Room_cost1 { get { return Room_cost; } set { Room_cost = value; } }

    public override List<object> GetAllData()
    {
        return service.GetAllData();
    }
    public override void Save()
    {
        service.Save(WindowsFormsbirdem.UI.frmRoom.roomInstant);
    }
    public void Update()
    {
        service.Update(WindowsFormsbirdem.UI.frmRoom.roomInstant);
    }
    public override void Delete()
    {
        service.Delete(WindowsFormsbirdem.UI.frmRoom.roomInstant.Room_id1.ToString());
    }
}

Actions:

✓ All Private variables are used as public variables using set and get methods.
✓ Helping file RoomService is called.
✓ Save, Delete, Update instances are called.
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MASICEIU.BaseDataLayer;
using MASICEIU;
using System.Data;

namespace WindowsFormsbirdem.DAL.DOL
{
    public class RoomService : Service
    {
        RoomDataAccess dataAccess = new RoomDataAccess();

        public override List<object> GetAllData()
        {
            DataTable dt = dataAccess.GetAllData();
            return MapObject(dt);
        }

        public override List<object> MapObject(System.Data.DataTable dataTable)
        {
            List<object> list = new List<object>();
            foreach (DataRow row in dataTable.Rows)
            {
                RoomDataObject roomDataObject = new RoomDataObject();
                roomDataObject.Room_id1 = NullHandler.GetInt(row["Room_id"]);
                roomDataObject.Room_no1 = NullHandler.GetString(row["Room_no"]);
                roomDataObject.Room_type1 = NullHandler.GetString(row["Room_type"]);
                roomDataObject.Room_cost1 = NullHandler.GetInt(row["Room_cost"]);
                list.Add(roomDataObject);
            }
            return list;
        }

        public override void Save(object objectValue)
        {
            dataAccess.Save(objectValue);
        }

        public void Update(object objectValue)
        {
            dataAccess.Update(objectValue);
        }

        public override bool Delete(string query)
        {
            dataAccess.Delete(query);
            return true;
        }
    }
}
Actions:

✓ All variable types are set.
✓ Helping class RoomDataAccess is called
✓ Save, Delete and Update are ensured.

Code of RoomDataAccess.cs Class

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MASICEIU.BaseDataLayer;
using System.Data;
using System.Data.SqlClient;

namespace WindowsFormsbirdem.DAL
{
    public class RoomDataAccess : DataAccess
    {
        public override DataTable GetAllData()
        {
            return ConnectionManager.DatabaseInstant.GetTable("Select *from tbl_Room");
        }

        public override void Save(object objectValue)
        {
            RoomDataObject obj = (RoomDataObject)objectValue;
            string query = "insert into tbl_Room values(" + obj.Room_id1 + ",'," + obj.Room_no1 + ",'," + obj.Room_type1 + ",'," + obj.Room_cost1 + ")";
            ConnectionManager.DatabaseInstant.Insert(query);
            }

        public override int NextID(string query)
        {
            return ConnectionManager.DatabaseInstant.NextID(query);
        }

        public void Update(object objectValue)
        {
            RoomDataObject obj = (RoomDataObject)objectValue;
            string query = 
                "Update tbl_Room set Room_no='" + obj.Room_no1 + ",Room_type='" + obj.Room_type1 + ",Room_cost=" + obj.Room_cost1 + " where Room_id='" + obj.Room_id1 + ";
            ConnectionManager.DatabaseInstant.Update(query);
        }

        public override bool Delete(string query)
        {
            ConnectionManager.DatabaseInstant.Delete("delete from tbl_Room where Room_id='" + query + ");
            return true;
        }
    }
```
Actions:

✓ All sql queries like Save, Delete, Update are done here.

✓ Data from Sql Server is controlled using this class.

THE END