VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi, Karnataka 590018

DBMS MINI-PROJECT REPORT

on

PHARMACY MANAGEMENT SYSTEM

Submitted in partial fulfillment of the requirement
for the award of the degree of

Bachelor of Engineering

in

Information Science & Engineering

by

Sarthak Sureka (1BG15IS044)

Vidyaya Amrutham Ashnuthe

B.N.M. Institute of Technology

12th Main, 27th Cross, Banashankari II Stage, Bangalore 560 070.

Department of Information Science and Engineering

2017 – 2018

B.N.M. Institute of Technology

12th Main, 27th Cross, Banashankari II Stage, Bangalore - 560070

Department of Information Science & Engineering

Vidyaya Amrutham Ashnuthe

CERTIFICATE

Certified that the Mini-project entitled, **Pharmacy Management System** is carried out by Mr. **Sarthak Sureka** USN **1BG15IS044**, the bonafide student of **B.N.M Institute of Technology** in partial fulfillment for the award of **Bachelor of Engineering** in **Information Science & Engineering** of the **Visvesvaraya Technological University**, Belagavi during the year 2017-2018. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The mini-project report has been approved as it satisfies the academic requirements in respect of mini-project prescribed for the said Degree.

Prof.Varalatchoumy.M	Dr. Surabhi Narayan
Asst. Prof., Dept. of ISE	**Prof & Head, Dept. of ISE**
BNMIT	BNMIT

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

Pharmacy management system is a management system that is designed to improve accuracy and to enhance safety and efficiency in the pharmaceutical store. This program can be used in any pharmaceutical shops having a database to maintain. It is a computer based system which helps the pharmacist to improve inventory management, cost, medical safety etc. The system allows the user to enter a manufacturing and expiry date for a particular product or drug during opening stock and sales transaction. The software can print invoices, bills, receipts etc. It can also maintain the record of supplies sent in by the supplier. The system services and goals are established by consultation with system user. It also involves manual entry upon arrival of new batches of drugs and upon drug movement out of the pharmacy for a certain period. Pharmacy management system is robust, integrated technology. Pharmacy management system deals with the maintenance of drugs and consumables in the pharmacy unit. This pharmacy management system is user friendly.

The main aim of the project is to manage the database of a pharmaceutical shop. This project provides insight into the design and implementation of a Pharmacy Management System. It is done by creating a database of the available medicines and equipments in the shop. The primary aim of pharmacy management system is to improve accuracy and enhance safety and efficiency in the pharmaceutical store and to develop software for its effective management. Pharmacy management system is useful to maintain correct database by providing an option to update the drugs in stock and is used to manage most pharmacy related activities.

1.1 Requirements Analysis

The schema for keeping records of all the medicines and equipments sold in a pharmacy can be taken from the details given below :-

- The database needs to keep track of each customer's details (name, sex, city, age, phone) and doctor's details like name, speciality, age, mobile and gender who prescribe mediquipments to customers.
- The details of mediquipments should include its trade name, product type, expiry & manufacturing date and price.
- The database should provide information about the manufacturer(cid, name, phone, email, city) and the supplier(name, city, mobile, email) who supplies the consignment to pharmacy(phid, name, city, phone, fax).

- The database should display details about the employee(name, city, date of joining, mobile, salary, age, sex) who works in the pharmacy and the hospitals(name, email, phone, city) to which it supplies the stocks of medicines and equipments.
- The database should also display and save details of the bills(bid, date, product, amount, quantity) produced by pharmacy.

Consider any relational entity between two entities using suitable attributes. The domain for city is{..all INDIAN states} and the domain of gender/sex is {M,F}.

It is a user friendly application for pharmacist which reduces the burden and helps to manage all sections of pharmacy like medicine management and billing etc., which improve processing efficiency. This will enhance the efficiency of clinical work and ease patient's convenience. In Pharmacy, billing management is the key process. In addition, Pharmacy management system will be able to process drug prescription with ease, including safe data store about medicine as well as fast searching, delete and update of medicines. The pharmacy management system is built for the sake of ensuring effective and clear data saving and manipulating as well as neat work on the pharmacy medical products. It highly minimizes time taken to manage data and is highly resourceful, which helps to manage pharmacy data in least time. The main objective of the PMS is making the pharmacy organization computerized by creating neat work through minimizing or eliminating wastage of time as well as removing the resources such as papers, for data saving, that decreases malfunctioned work on the medical usage by giving correct information on each medicine. This system is also helpful to improve the efficiency of the system by ensuring effective monitoring of services and activities. A summarized list of drugs dispensed to patient can be viewed for monitoring purposes.

CHAPTER 2

DESIGN

Design is the creation of a plan or convention for the construction of an object, system or measurable human interaction. Designing often necessitates considering the aesthetic, functional, economic and sociopolitical dimensions of both the design object and design process.

Thus "design" may be a substantive referring to a categorical abstraction of a created thing or things (the design of something), or a verb for the process of creation. It is an act of creativity and innovation.

2.1 ENTITY-RELATIONSHIP DIAGRAM

An entity–relationship model (ER model) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

ER Diagram for this project consists of 9 regular entities and 3 relational/associative entities.

The regular entities are:

1: PHARMACY

2: CUSTOMER

3: DOCTOR

4: MANUFACTURER

5: MEDIQUIPMENT

6: EMPLOYEE

7: HOSPITAL

8: BILL

9: SUPPLIER

The associative entities are:

1: PRESCRIBE

2: CONTRACT

3: WORKS

The attributes of each of these entities are:

1: PHARMACY {PHID, NAME, CITY, FAX, PHONE} where PHID is primary key.

2: CUSTOMER {PID, NAME, SEX, CITY, PHONE, AGE, DID} where PID is primary key and DID is foreign key.
3: DOCTOR {DID, DNAME, SPECIALITY, AGE, MOBILE, GENDER} where DID is primary key.
4: MANUFACTURER {CID, NAME, EMAIL, MOBILE, CITY, PHARID} where CID is primary key and PHARID is foreign key.
5: MEDIQUIPMENT {CODE, TRADE_NAME, PRODUCT_TYPE, MFG_DATE, EXP_DATE, PRICE, CID} where CODE is primary key and CID is foreign key.
6: EMPLOYEE {NAME, CITY, DOJ, MOBILE, SALARY, AGE, SEX, PHARID} where MOBILE is primary key and PHARID is foreign key.
7: HOSPITAL {HID, NAME, EMAIL, PHONE, CITY, PHARID} where HID is primary key and PHARID is foreign key.
8: BILL {BID, DOB, AGE, PNAME, MOBILE, CITY, PRODUCT, AMOUNT, PHARID} where BID is primary key.
9: SUPPLIER {NAME, CITY, MOBILE, EMAIL, CID, PHARID} where MOBILE is primary key and CID, PHARID is foreign key.
10: PRESCRIBE {DATE, MEDICINE, DID, PID} where DID, PID is foreign key.
11: CONTRACT {PHARID, CID, START_DATE, END_DATE} where PHARID, CID is foreign key.
12: WORKS {PHARID, START_DATE, END_DATE} where PHARID is foreign key.

The relations between various entities are:

1. DOCTOR: CUSTOMER->PRESCRIBE

(M: N) Many doctors can prescribe to many customers.

2. DOCTOR: MEDIQUIPMENT->ADVISE

(1: N) One doctor can advise many medicines to a customer.

3. MANUFACTURER: MEDIQUIPMENT->MAKE

(1: N) One manufacturer can make many mediquipments.

4. MANUFACTURER: SUPPLIER->DEALS

(1: N) One manufacturer can have many suppliers.

5. PHARMACY: MEDIQUIPMENT->SELL

(M: N) Many pharmacies can sell many medicines.

6. PHARMACY: SUPPLIER->SUPPLIES

(1: N) One pharmacy can have many suppliers.

7. PHARMACY: MANUFACTURER->CONTRACT

(1: N) One pharmacy can have many manufacturers.

8. PHARMACY: HOSPITAL->SUPPLY TO

(1: N) One pharmacy can supply to many hospitals.

9. PHARMACY: BILL->GIVES

(1: N) One pharmacy can produce many bills.

10. PHARMACY: EMPLOYEE->WORKS

(1: N) One pharmacy can have many employees.

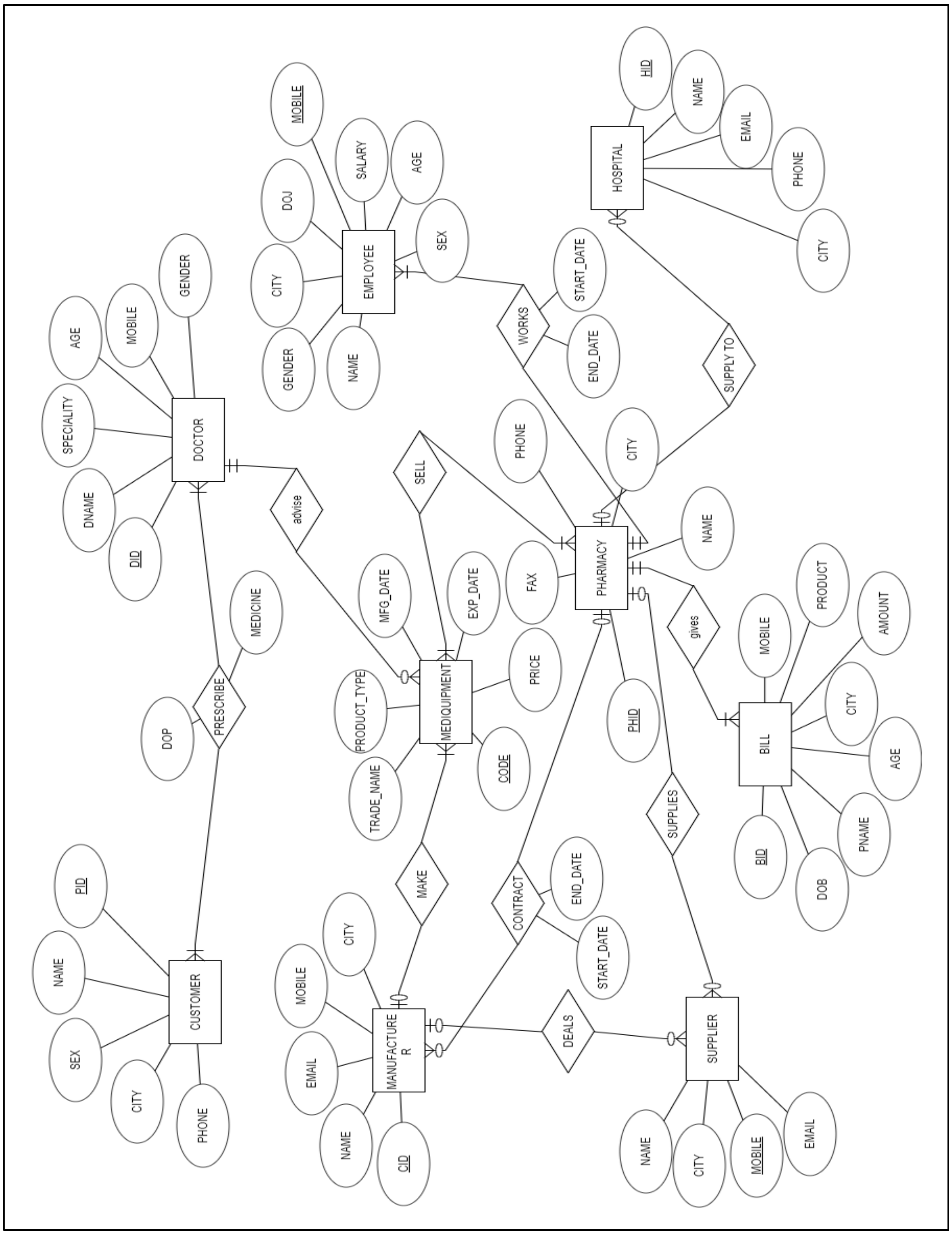


Fig.2.1 – ER Diagram of Pharmacy Management System

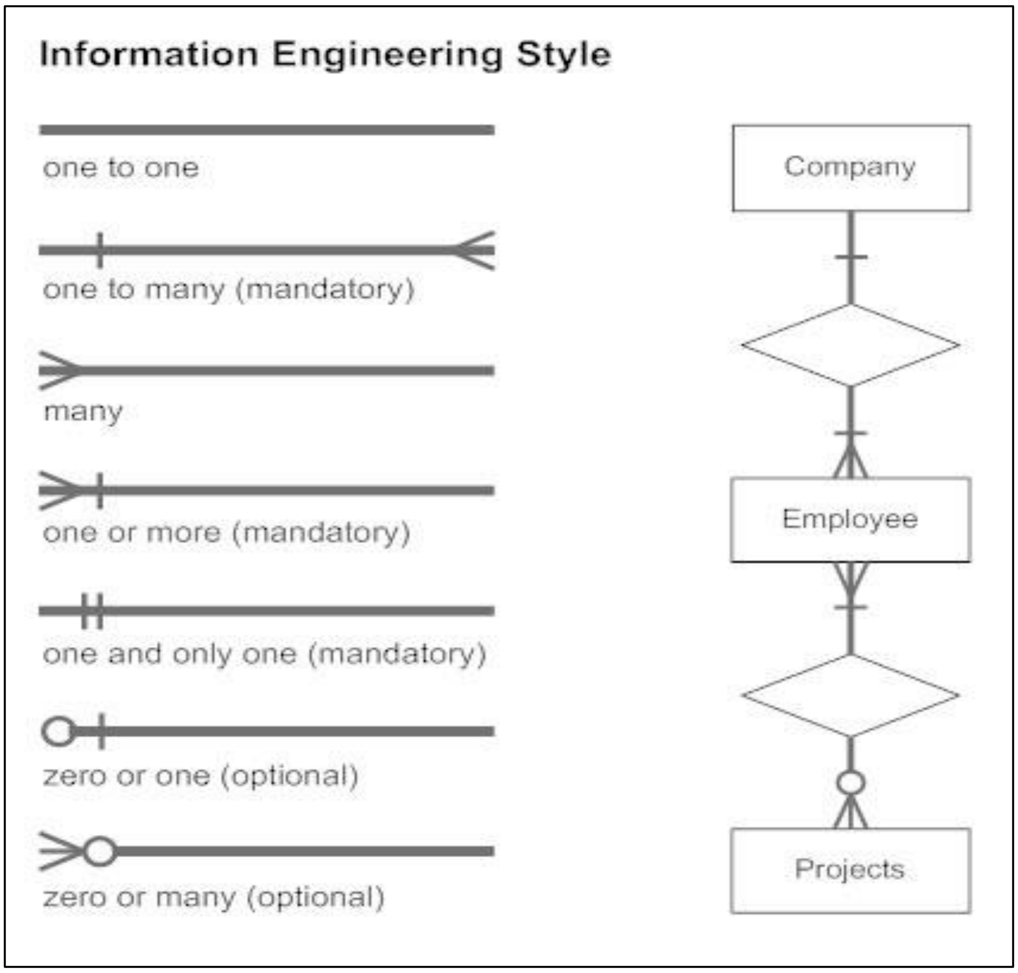


Fig.2.2 – Different relations used in ER diagram

2.2 ER to RELATIONAL MAPPING

<u>ER-to-Relational Mapping Algorithm</u>

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued attributes.

Step 7: Mapping of N-ary Relationship Types.

<u>Step 1: Mapping of Regular Entity Types.</u>

1.1: For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.

1.2 : Choose one of the key attributes of E as the primary key for R.

1.3 : If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

Step 2: Mapping of Weak Entity Types

2.1: For each weak entity type W in the ER schema with owner entity type E, create a relation R & include all simple attributes (or simple components of composite attributes) of W as attributes of R.

2.2: Also, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).

2.3: The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

Step 3: Mapping of Binary 1:1 Relation Types

For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.

There are three possible approaches:

3.1: Foreign Key approach: Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S.

3.2: Merged relation option: An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.

3.3: Cross-reference or relationship relation option: The third alternative is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

Step 4: Mapping of Binary 1:N Relationship Types.

4.1: For each regular binary 1:N relationship type R, identify the relation S that . represent the participating entity type at the N-side of the relationship type

4.2: Include as foreign key in S the primary key of the relation T that represents the . other entity type participating in R.

4.3: Include any simple attributes of the 1:N relation type as attributes of S.

Step 5: Mapping of Binary M:N Relationship Types.

5.1: For each regular binary M:N relationship type R, create a new relation S to represent R.

5.2: Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.

5.3: Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

Step 6: Mapping of Multivalued attributes.

6.1: For each multivalued attribute A, create a new relation R.

6.2: This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.

6.3: The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

Step 7: Mapping of N-ary Relationship Types.

7.1: For each n-ary relationship type R, where n>2, create a new relationship S to represent R.

7.2: Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.

7.3: Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.

Correspondence between ER and Relational Models

Entity type “Entity” relation

1:1 or 1:N relationship type Foreign key (or “relationship” relation)

M:N relationship type “Relationship” relation and two foreign keys

n-ary relationship type “Relationship” relation and n foreign keys

Simple attribute

Composite attribute Set of simple component attributes

Multi-valued attribute Relation and foreign key

Value set Domain

Key attribute Primary (or secondary) key

2.3 RELATIONAL MODEL

A relational database schema is the tables, columns and relationships that make up a relational database.

A relational database schema helps you to organize and understand the structure of a database. This is particularly useful when designing a new database, modifying an existing database to support more functionality, or building integration between databases.

The Relational model contains all entities and their relations with other entities. It also contains all relations having (m: n) cardinality.

The entities used are:

1: PHARMACY {PHID, NAME, CITY, FAX, PHONE} where PHID is primary key.
2: CUSTOMER {PID, NAME, SEX, CITY, PHONE, AGE, DID} where PID is primary key and DID is foreign key.
3: DOCTOR {DID, DNAME, SPECIALITY, AGE, MOBILE, GENDER} where DID is primary key.
4: MANUFACTURER {CID, NAME, EMAIL, MOBILE, CITY, PHARID} where CID is primary key and PHARID is foreign key.
5: MEDIQUIPMENT {CODE, TRADE_NAME, PRODUCT_TYPE, MFG_DATE, EXP_DATE, PRICE, CID} where CODE is primary key and CID is foreign key.
6: EMPLOYEE {NAME, CITY, DOJ, MOBILE, SALARY, AGE, SEX, PHARID} where MOBILE is primary key and PHARID is foreign key.
7: HOSPITAL {HID, NAME, EMAIL, PHONE, CITY, PHARID} where HID is primary key and PHARID is foreign key.
8: BILL {BID, DOB, AGE, PNAME, MOBILE, CITY, PRODUCT, AMOUNT, PHARID} where BID is primary key.
9: SUPPLIER {NAME, CITY, MOBILE, EMAIL, CID, PHARID} where MOBILE is primary key and CID, PHARID is foreign key.
10: PRESCRIBE {DATE, MEDICINE, DID, PID} where DID, PID is foreign key.
11: SELL {PHID, CODE} is a relation that uses PHID, CODE as foreign key.
All Primary keys are used to apply key constraint on relational model.
All foreign keys are used to apply referential integrity constraint on relational model.
Domain constraint are applied for each attribute by defining them with suitable data type.

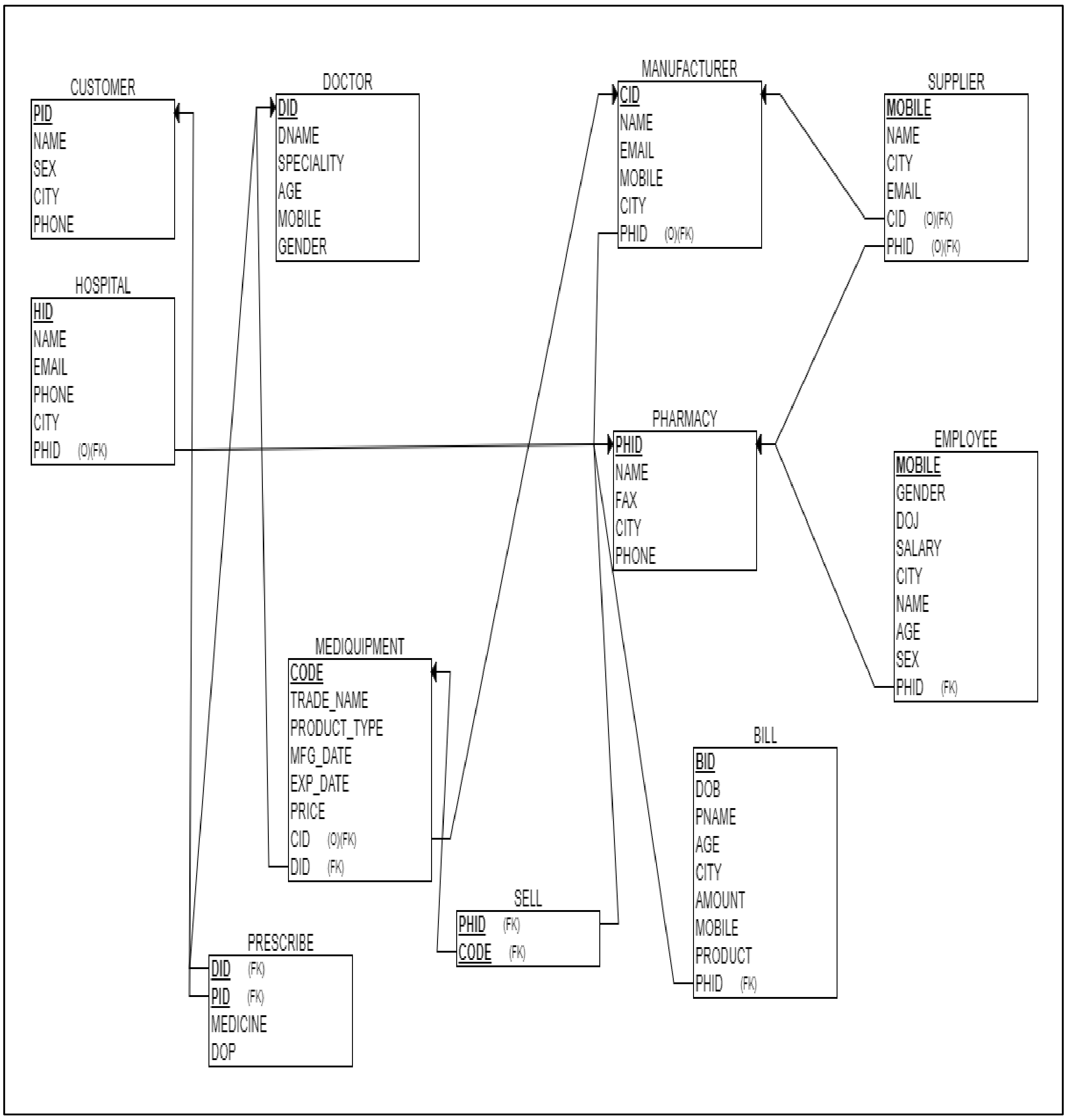


Fig.2.3 - Relational Model for Pharmacy Management System

2.4 FUNCTIONAL DEPENDENCIES

Functional dependency is a relationship that exists when one attribute uniquely determines another attribute. If R is a relation with attributes X and Y, a functional dependency between the attributes is represented as X->Y, which specifies Y is functionally dependent on X.

1. PHARMACY {PHID, NAME, CITY, FAX, PHONE}

   PHID-> {NAME, CITY, FAX, PHONE}

2. DOCTOR {DID, DNAME, SPECIALITY, AGE, MOBILE, GENDER}

   DID-> {DNAME, SPECIALITY, AGE, MOBILE, GENDER}

3. CUSTOMER {PID, NAME, SEX, CITY, PHONE, AGE, DID}

   PID-> {NAME, SEX, CITY, PHONE, AGE}

4. MANUFACTURER {CID, NAME, EMAIL, MOBILE, CITY, PHARID}

 CID-> {NAME, EMAIL, MOBILE, CITY}

5. MEDIQUIPMENT {CODE, TRADE_NAME, PRODUCT_TYPE, MFG_DATE, EXP_DATE, PRICE, CID}

 CODE-> {TRADE_NAME, PRODUCT_TYPE, MFG_DATE, EXP_DATE, PRICE}

6. SUPPLIER {NAME, CITY, MOBILE, EMAIL, CID, PHARID}

 MOBILE-> {NAME, CITY, EMAIL}

7. EMPLOYEE {NAME, CITY, DOJ, MOBILE, SALARY, AGE, SEX, PHARID}

 MOBILE-> {NAME, CITY, DOJ, SALARY, AGE, SEX}

8. HOSPITAL {HID, NAME, EMAIL, PHONE, CITY, PHARID}

 HID-> {NAME, EMAIL, PHONE, CITY}

9. BILL {BID, DOB, AGE, PNAME, MOBILE, CITY, PRODUCT, AMOUNT, PHARID}

 BID-> {DOB, AGE, PNAME, MOBILE, CITY, PRODUCT, AMOUNT}

10. WORKS {PHARID, START_DATE, END_DATE}

11. CONTRACT {PHARID, CID, START_DATE, END_DATE}

 PHARID-> {CID}

12. PRESCRIBE {DATE, MEDICINE, DID, PID}

 DID-> {PID}

2.5 NORMALISED RELATIONAL SCHEMA

Normalization involves arranging attributes in relations based on dependencies between attributes, ensuring that the dependencies are properly enforced by database integrity constraints. Normalization is accomplished by applying some formal rules either by a process of synthesis or decomposition. Synthesis creates a normalized database design based on a known set of dependencies. Decomposition takes an existing (insufficiently normalized) database design and improves it based on the known set of dependencies.

1NF:

First normal form (1NF) is a property of a relation in a relational database. A relation is in first normal form if and only if the domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain.

First normal form enforces these criteria:

- Eliminate repeating groups in individual tables.
- Create a separate table for each set of related data.
- Identify each set of related data with a primary key

Pharmacy: PHID, NAME, CITY, FAX, PHONE have atomic values. Thus the relation is in 1NF.

Doctor: DID, DNAME, SPECIALITY, AGE, MOBILE, GENDER have atomic values. Thus the relation is in 1NF.

Customer: PID, NAME, SEX, CITY, PHONE, AGE, DID have atomic values. Thus the relation is in 1NF.

Hospital: HID, NAME, EMAIL, PHONE, CITY, PHARID have atomic values. Thus the relation is in 1NF.

Employee: NAME, CITY, DOJ, MOBILE, SALARY, AGE, SEX, PHARID have atomic values. Thus the relation is in 1NF.

Manufacturer: CID, NAME, EMAIL, MOBILE, CITY, PHARID have atomic values. Thus the relation is in 1NF.

Mediquipment: CODE, TRADE_NAME, PRODUCT_TYPE, MFG_DATE, EXP_DATE, PRICE, CID have atomic values. Thus the relation is in 1NF.

Supplier: NAME, CITY, MOBILE, EMAIL, CID, PHARID have atomic values. Thus the relation is in 1NF.

Bill: BID, DOB, AGE, PNAME, MOBILE, CITY, PRODUCT, AMOUNT have atomic values. Thus the relation is in 1NF.

Works: PHARID, START_DATE, END_DATE have atomic values. Thus the relation is in 1NF.

Contract: PHARID, START_DATE, END_DATE, CID have atomic values. Thus the relation is in 1NF.

Prescribe: DATE, MEDICINE, DID, PID have atomic values. Thus the relation is in 1NF.

2NF:

Second normal form (2NF) is a normal form used in database normalization. A relation that is in first normal form (1NF) must meet additional criteria if it is to qualify for second normal form.

Specifically: a relation is in 2NF if it is in 1NF and no non-prime attribute is dependent on any proper subset of any candidate key of the relation. A non-prime attribute of a relation is an attribute that is not a part of any candidate key of the relation.

Pharmacy: PHID, NAME, CITY, FAX, PHONE. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF too.

Doctor: DID, DNAME, SPECIALITY, AGE, MOBILE, GENDER. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF too.

Customer: PID, NAME, SEX, CITY, PHONE, AGE, DID. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF too.

Hospital: HID, NAME, EMAIL, PHONE, CITY, PHARID. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF too.

Employee: NAME, CITY, DOJ, MOBILE, SALARY, AGE, SEX, PHARID. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF too.

Manufacturer: CID, NAME, EMAIL, MOBILE, CITY, PHARID. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF too.

Mediquipment: CODE, TRADE_NAME, PRODUCT_TYPE, MFG_DATE, EXP_DATE, PRICE, CID. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF too.

Supplier: NAME, CITY, MOBILE, EMAIL, CID, PHARID. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF too.

Bill: BID, DOB, AGE, PNAME, MOBILE, CITY, PRODUCT, AMOUNT. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF too.

Works: PHARID, START_DATE, END_DATE. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF too.

Contract: PHARID, START_DATE, END_DATE, CID. Since, non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is not in 2NF.

So it is divided into two tables to convert it in 2NF.

Contract1: PHARID, CID, START_DATE

Contract2: PHARID, CID, END_DATE

Prescribe: DATE, MEDICINE, DID, PID. Since non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is not in 2NF.

So it is divided into two tables to convert it in 2NF.

Prescribe1: PID, DID, DATE

Prescribe2: PID, DID, MEDICINE

3NF:

Third normal form is a normal form that is used in normalizing a database design to reduce the duplication of data and ensure referential integrity by ensuring that (1) the entity is in second normal form, and (2) all the attributes in a table are determined only by the candidate keys of that relation and not by any non-prime attributes. 3NF was designed to improve database processing while minimizing storage costs. 3NF data modelling was ideal for online transaction processing (OLTP) applications with heavy order entry type of needs.

Pharmacy: PHID, NAME, CITY, FAX, PHONE. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF and 3NF too.

Doctor: DID, DNAME, SPECIALITY, AGE, MOBILE, GENDER. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in2NF and 3NF too.

Customer: PID, NAME, SEX, CITY, PHONE, AGE, DID. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF and3NF too.

Hospital: HID, NAME, EMAIL, PHONE, CITY, PHARID. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF and3NF too.

Employee: NAME, CITY, DOJ, MOBILE, SALARY, AGE, SEX, PHARID. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF and3NF too.

Manufacturer: CID, NAME, EMAIL, MOBILE, CITY, PHARID. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF and3NF too.

Mediquipment: CODE, TRADE_NAME, PRODUCT_TYPE, MFG_DATE, EXP_DATE, PRICE, CID. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF and3NF too.

Supplier: NAME, CITY, MOBILE, EMAIL, CID, PHARID. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF and3NF too.

Bill: BID, DOB, AGE, PNAME, MOBILE, CITY, PRODUCT, AMOUNT. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF and3NF too.

Works: PHARID, START_DATE, END_DATE. Since no non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is in 2NF and3NF too.

Contract: PHARID, START_DATE, END_DATE, CID. Since, non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is not in 2NF.

So it is divided into two tables to convert it in 2NF.

Contract1: PHARID, CID, START_DATE

Contract2: PHARID, CID, END_DATE.

It is in 3NF too.

Prescribe: DATE, MEDICINE, DID, PID. Since non-prime attribute is dependent on the candidate key and the relation is in 1NF, it is not in 2NF.

So it is divided into two tables to convert it in 2NF.

Prescribe1: PID, DID, DATE

Prescribe2: PID, DID, MEDICINE

It is in 3NF too.

2.6 KEY ATTRIBUTES

ENTITY	PRIMARY KEY	FOREIGN KEY
PHARMACY	PHID	
DOCTOR	DID	
CUSTOMER	PID	DID
MANUFACTURER	CID	PHARID
MEDIQUIPMENT	CODE	CID
SUPPLIER	MOBILE	CID, PHARID
EMPLOYEE	MOBILE	PHARID
HOSPITAL	HID	PHARID
BILL	BID	PHARID
WORKS	PHARID	PHARID
CONTRACT	PHARID, CID	PHARID, CID
PRESCRIBE	PID, DID	DID,PID

Table 2.1 – Key Attributes used in Pharmacy Management System

PRIMARY KEY

Primary key is a set of one or more fields/columns of a table that uniquely identify a record in database table. It cannot accept null, duplicate values. Only one Candidate Key can be Primary Key.

FOREIGN KEY

Foreign Key is a field in database table that is Primary key in another table. It can accept multiple null, duplicate values. For more help refer the article Difference between primary key and foreign key.

UNIQUE KEY

Unique key is a set of one or more fields/columns of a table that uniquely identify a record in database table. It is like Primary key but it can accept only one null value and it cannot have duplicate values. For more help refer the article Difference between primary key and unique key.

NOT NULL

The NOT NULL constraint enforces a column to NOT accept NULL values. This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

INDEX

A field which has unique values is, essentially, a key. However, a key is used to uniquely identify a row in a table, while an index is used to sort, or group, the rows in the table. A key should not change once it has been initially set, as it might be referenced to elsewhere in your database.

THE DEFAULT SQL CONSTRAINT

At times, you might want to automatically insert a value into a column when no other value is provided. That's where the DEFAULT SQL constraint comes in. You can use the constraint to define a column's default value. This can be a handy way to add the current date and time to a column or to avoid having to use NULL values.

CHECK CONSTRAINT

A check constraint is a type of integrity constraint in SQL which specifies a requirement that must be met by each row in a database table. The constraint must be a predicate. It can refer to a single column, or multiple columns of the table.

2.6 RELATIONAL DATABASE SCHEMA

It formulates all the constraints that are to be applied on the data. A database schema defines its entities and the relationship among them.

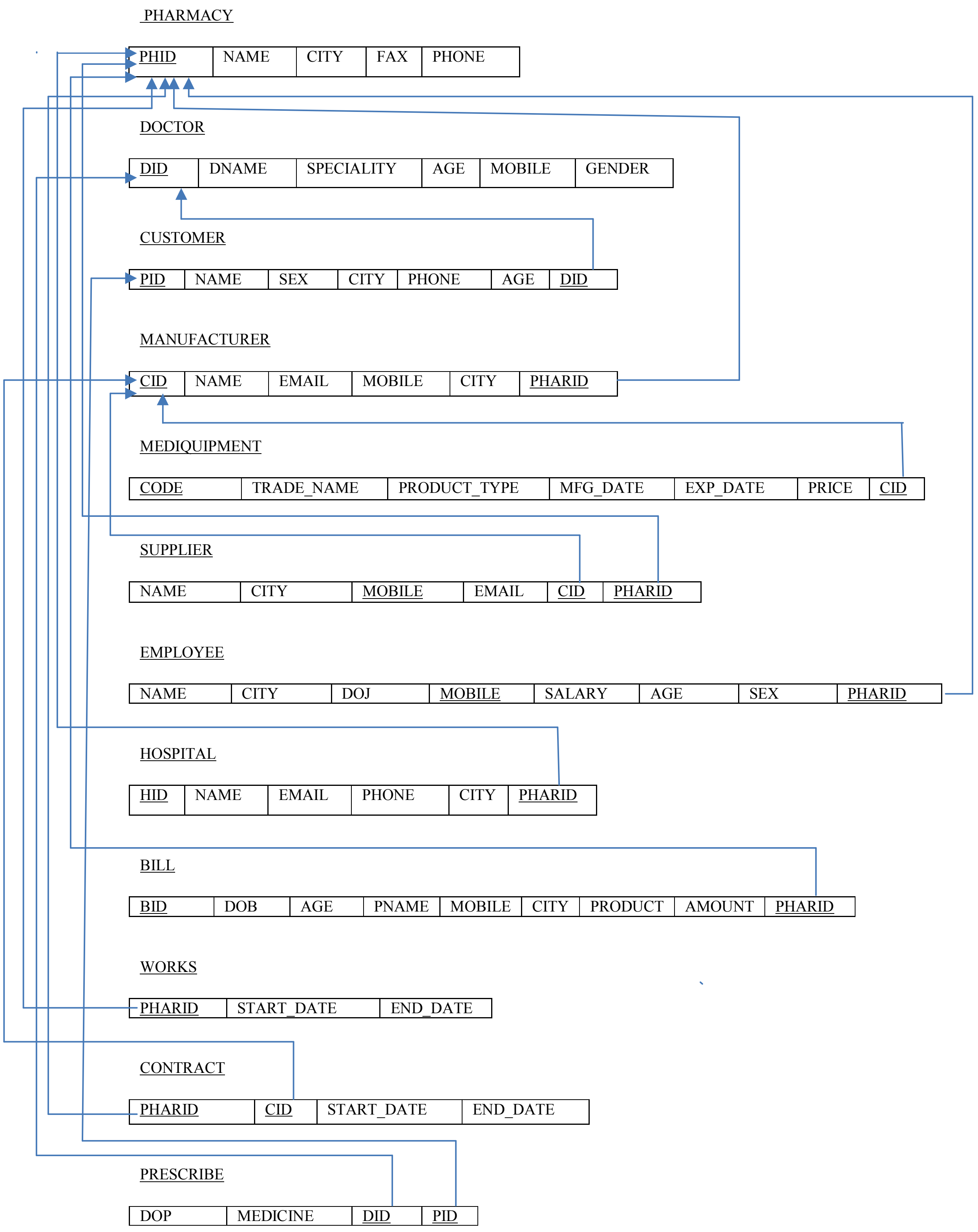

PHARMACY
PHID | NAME | CITY | FAX | PHONE
DOCTOR
DID | DNAME | SPECIALITY | AGE | MOBILE | GENDER
CUSTOMER
PID | NAME | SEX | CITY | PHONE | AGE | DID
MANUFACTURER
CID | NAME | EMAIL | MOBILE | CITY | PHARID
MEDIQUIPMENT
CODE | TRADE_NAME | PRODUCT_TYPE | MFG_DATE | EXP_DATE | PRICE | CID
SUPPLIER
NAME | CITY | MOBILE | EMAIL | CID | PHARID
EMPLOYEE
NAME | CITY | DOJ | MOBILE | SALARY | AGE | SEX | PHARID
HOSPITAL
HID | NAME | EMAIL | PHONE | CITY | PHARID
BILL
BID | DOB | AGE | PNAME | MOBILE | CITY | PRODUCT | AMOUNT | PHARID
WORKS
PHARID | START_DATE | END_DATE
CONTRACT
PHARID | CID | START_DATE | END_DATE
PRESCRIBE
DOP | MEDICINE | DID | PID

Fig.2.4 - Relational Database Schema for Pharmacy Management System

CHAPTER 3

SYSTEM REQUIREMENTS

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time.

3.1 Hardware and Software requirements

Hardware Requirements :-

- Processor :- Intel Core i3 CPU @2.10GHz or above
- RAM :- 2GB or above
- Hard Disk :- Minimum 1GB
- CD Drive

Software Requirements :-

- Operating System :- Windows 7 or higher
- Database Management System :- Oracle 10g or higher
- Netbeans IDE (Complete Package)
- Java Database Connectivity (JDBC) Driver

User Interfaces :-

- Front end :- Netbeans IDE
- Back end :- Oracle 10g

Design Tools :-

- ERD Plus

3.2 Tools

NETBEANS IDE

Netbeans is a software development platform written in Java. The Netbeans Platform allows application to be developed from a set of modular software components called modules. Applications based on the Netbeans Platform, including the Netbeans integrated development environment (IDE), can be extended by third party developers.

The Netbeans IDE is primarily intended for development in Java, but also supports other languages, in particular PHP, C/C++ and HTML5.

Netbeans is cross-platform and runs on Microsoft Windows, mac OS, Linux, Solaris and other platforms supporting a compatible JVM.

The editor supports many languages from Java, C/C++, XML and HTML, to PHP, Groovy, Javadoc, JavaScript and JSP. Because the editor is extensible, you can plug in support for many other languages.

A new version was released 8.2 on October 3, 2016.NetBeans IDE is the official IDE for Java 8. With its editors, code analyzers, and converters, you can quickly and smoothly upgrade your applications to use new Java 8 language constructs, such as lambdas, functional operations, and method references.

Netbeans IDE is an open-source integrated development environment. Netbeans IDE supports development of all Java application types (Java SE (including JavaFX), Java ME, web, EJB and mobile applications) out of the box. Among other features are an Ant-based project system, Maven support, refactoring, and version control (supporting CVS, Subversion, Git, Mercurial and Clearcase).

JDBC DRIVER

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is Java based data access technology and used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database, and is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand.

Commercial and free drivers fall into one of the following types:

- Type 1 that calls native code of the locally available ODBC driver. (Note: In JDBC 4.2, JDBC-ODBC bridge has been removed)
- Type 2 that calls database vendor native library on a client side. This code then talks to database over the network.

- Type 3, the pure-java driver that talks with the server-side middleware that then talks to the database.
- Type 4, the pure-java driver that uses database native protocol.

Note also a type called an internal JDBC driver - a driver embedded with JRE in Java-enabled SQL databases. It is used for Java stored procedures. This does not fit into the classification scheme above, although it would likely resemble either a type 2 or type 4 driver (depending on whether the database itself is implemented in Java or not). However, in the case of an internal JDBC driver, the JDBC client actually runs as part of the database being accessed, and so can access data directly rather than through network protocols.

ORACLE 10G DATABASE MANAGEMENT SYSTEM

Oracle Database (commonly referred to as Oracle RDBMS or simply as Oracle) is an object-relational database management system produced and marketed by Oracle Corporation.

Larry Ellison and his two friends and former co-workers, Bob Miner and Ed Oates, started a consultancy called Software Development Laboratories (SDL) in 1977. SDL developed the original version of the Oracle software. The name Oracle comes from the code-name of a CIA-funded project Ellison had worked on while formerly employed by Ampex.

The Oracle RDBMS has had a reputation among novice users as difficult to install on Linux systems. Oracle Corporation has packaged recent versions for several popular Linux distributions in an attempt to minimize installation challenges beyond the level of technical expertise required to preinstall a database server.

More and more licenses of the Oracle Database are choosing third party support options, such as those provided by RiminiStreet.com, where they can save an immediate 50 percent or more of the cost of vendor-provided support, get a wider range of services options, and redeploy the saved funds into new IT and infrastructure investments.

Oracle Corporation also endorses certain practices and conventions as enhancing the use of its database products. These include:

- Oracle Maximum Availability Architecture (MAA) guidelines on developing high-availability systems

- Optimal Flexible Architecture (OFA), blueprints for mapping Oracle-database objects to file-systems.

ERD PLUS

ERD Plus is a web-based database modeling tool that lets you quickly and easily create

- Entity Relationship Diagrams (ERDs)
- Relational Schemas (Relational Diagrams)
- Star Schemas (Dimensional Models)

More features of these tools are

- Automatically convert ER Diagrams into Relational Schemas
- Export SQL
- Export diagrams as a PNG
- Save diagrams safely on our server

ERD Plus enables drawing standard ERD components

- Entities
- Attributes
- Relationships

The notation supports drawing regular and weak entities, various types of attributes (regular, unique, multi-valued, derived, composite, and optional), and all possible cardinality constraints of relationships (mandatory-many, optional-many, mandatory-one and optional-one).

ERD Plus enables drawing standard Relational Schema components

- Tables (Relations)
- Table Columns (including Primary and Foreign Keys)
- Referential Integrity Constraint Lines (pointing from a Foreign Key to the Primary Key it refers to)

The tool supports quick creation of foreign keys and referential integrity lines by simple click-point-connect actions. This simplifies and quickens the process of creating relational schemas.

CHAPTER 4

IMPLEMENTATION

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy.

FRONT END :-

JFrame form is used to create the front end of the project. A frame, implemented as an instance of the JFrame class, is a window that has decorations such as a border, a title, and supports button components that close or iconify the window. Applications with a GUI usually include at least one frame. Applets sometimes use frames, as well.

The description for each page is listed below in the same order as in the project :-

1) Introduction Form :-

   This form gives a brief overview of the system in hand and helps the user to go to the login page by clicking the user login button in the form, so that the user can login into the system and manage the database.

2) Login Form :-

   After pressing the user login button in introduction page, it opens the login page of the application where we have to enter the login details of the user and then press the login button to enter the application for further processes.

3) Options Form :-

   After the login is complete the user is directed to the options page where there are options to choose the task that the user wants to perform ,i.e., make a bill, check records or logout, and based on the choice, the user are being directed to the respective pages where they can perform the corresponding work. Buttons are used to direct the user to the corresponding page.

4) Bill Form :-

   In this form, enter the information of the customer and the product details that he is buying , then click on save to save the bill for later reviewing and then click on generate invoice to see the final bill layout before printing it.

5) Category Form :-

   In this form, there are categories to choose from different entities that are present in the application, so that the user can insert, delete or view the information stored in the database. So, there are buttons for all the entities, so that it can lead to the corresponding page in the application.

6) Pharmacy Form :-

   In this form, there are options to insert, delete and view details of the pharmacy table that is created in the back end by connecting it to the front end by the help of the a simple code written in java. All the processes in the code is carried out, after typing the details asked and then pressing the respective buttons assigned for each operations.

7) Doctor Form :-

   In this form, there are options to insert, delete and view details of the doctor table that is created in the back end by connecting it to the front end by the help of the same code that is used for the pharmacy form to carry out the process, just the difference is that for this form the table name and the attribute's name will change according to the database that is created in the back end. All the processes in the code is carried out, after pressing the respective buttons assigned for each operations.

8) Customer Form :-

   In this form, there are options to insert, delete and view details of the customer table that is created in the back end by connecting it to the front end by the help of the same code that is used for the pharmacy form to carry out the process, just the difference is that for this form the table name and the attribute's name will change according to the database that is created in the back end. All the processes in the code is carried out, after pressing the respective buttons assigned for each operations.

9) Manufacturer Form :-

   In this form, there are options to insert, delete and view details of the manufacturer table that is created in the back end by connecting it to the front end by the help of the same code that is used for the pharmacy form to carry out the process, just the difference is that for this form the table name and the attribute's name will change according to the database that is created in the back end. All the processes in the code is carried out, after pressing the respective buttons assigned for each operations.

10) Mediquipment Form :-

   In this form, there are options to insert, delete and view details of the mediquipment table that is created in the back end by connecting it to the front end by the help of the same code that is used for the pharmacy form to carry out the process, just the difference is that for this form the table name and the attribute's name will change according to the database that is created in the back end. All the processes in the code is carried out, after pressing the respective buttons assigned for each operations.

11) Supplier Form :-

In this form, there are options to insert, delete and view details of the supplier table that is created in the back end by connecting it to the front end by the help of the same code that is used for the pharmacy form to carry out the process, just the difference is that for this form the table name and the attribute's name will change according to the database that is created in the back end. All the processes in the code is carried out, after pressing the respective buttons assigned for each operations.

12) Employee Form :-

In this form, there are options to insert, delete and view details of the employee table that is created in the back end by connecting it to the front end by the help of the same code that is used for the pharmacy form to carry out the process, just the difference is that for this form the table name and the attribute's name will change according to the database that is created in the back end. All the processes in the code is carried out, after pressing the respective buttons assigned for each operations.

13) Hospital Form :-

In this form, there are options to insert, delete and view details of the hospital table that is created in the back end by connecting it to the front end by the help of the same code that is used for the pharmacy form to carry out the process, just the difference is that for this form the table name and the attribute's name will change according to the database that is created in the back end. All the processes in the code is carried out, after pressing the respective buttons assigned for each operations.

BACK END :-

Back end is used to create the table and insert the values in the respective tables in the database management system.

The various entities created in the back end are as follows :-

1) Pharmacy
2) Doctor
3) Customer
4) Manufacturer
5) Mediquipment
6) Supplier
7) Employee
8) Hospital
9) Bill
10) Works

11) Contract
12) Prescribe

The create table and the insert table statement for the respective entities is listed below :-

1) Pharmacy :-

```
create table pharmacy
(
phid integer primary key,
name varchar2(20),
city varchar2(20),
fax varchar2(20),
phone number(10));
```

PHID	NAME	CITY	FAX	PHONE
1	ABBOTT	BANGALORE	212-9765433	9001002003
2	ALLERGAN	MUMBAI	222-9838212	9001002004
3	JANSSEN	BANGALORE	233-9812113	9001002001
4	SOLVAY	DELHI	235-9862263	9010126387
5	ROCHE	CHENNAI	248-9818181	9661268268
6	SCHWARZ	BANGALORE	253-9866547	9662312312
7	BUSK	BANGALORE	258-9001536	9234234110
8	NOVARTIS	MYSORE	287-9112543	9235231510
9	BOOTS	CHENNAI	284-9112541	9878685412

Table 4.1 – Pharmacy Table

2) Doctor :-

```
create table doctor
(
did number(2) primary key,
dname varchar2(20),
speciality varchar2(20),
age number(2) not null,
mobile integer,
gender varchar2(6));
```

DID	DNAME	SPECIALITY	AGE	MOBILE	GENDER
10	DPGUPTA	CARDIOLOGISTS	45	9831312512	M
11	AKSAHAY	DIABETOLOGISTS	50	9826263264	M
12	LALAASHOK	GYNAECOLOGISTS	30	9006001002	M
13	CNMISHRA	HYGIENISTS	32	9875400500	M
14	MKMISHRA	LEPROLOGISTS	28	9861621312	M
15	VINEETA	NEUROLOGISTS	28	9811254112	F
16	RAGINI KHANNA	NEUROLOGISTS	24	9831054112	F
17	AJAY SOLANKI	PATHOLOGISTS	32	9811211311	M
18	MD ARIF	PSYCHIATRIST	31	9006112534	M
19	SALMA AHMED	SEXOLOGISTS	32	9110110221	F

Table 4.2 – Doctor Table

3) Customer :-

```
create table customer
(
pid number(3) primary key,
name varchar2(20),
sex varchar2(6),
city varchar2(20),
phone integer,
age number(2),
did references doctor(did) on delete set null);
```

PID	NAME	SEX	CITY	PHONE	AGE	DID
100	ANSHU	M	DELHI	9002001004	21	10
101	IQBAL	M	RAMPUR	9821330400	8	11
102	PRIYA	F	CHENNAI	9840050022	7	12
103	KAUSHIK	M	PATNA	9840050023	14	13
104	NITU	F	BANGALORE	9840050024	13	14
105	RESHMA	F	BANGALORE	9840050028	22	14
106	REKHA	F	ILLINOIS	9840050026	35	15
107	ALISHA	F	PUNE	9840050027	50	16
108	SONIA	F	PEHLGAON	9840050021	57	14
109	VISHAK SEN	M	ARWAL	9821246112	41	17

Table 4.3 – Customer Table

4) Manufacturer :-

```
create table manufacturer
(
cid number(4) primary key,
name varchar2(20),
email varchar2(20),
mobile integer,
city varchar2(20),
pharid references pharmacy(phid) on delete set null);
```

CID	NAME	EMAIL	MOBILE	CITY	PHID
1000	HOIVO	HOIVO1@GMAIL	9112113114	BANGALORE	1
1001	MEDICYL	MED2@GMAIL	9212123134	MYSORE	2
1002	VITADE	VIT3@GMAIL	9812121368	CHENNAI	2
1003	ANNUVA	ANN4@GMAIL	9001002115	DELHI	3
1004	MANDAVUS	MAN5@GMAIL	9122113114	HYDERABAD	1
1005	INOVINE	INO5@GMAIL	9881828112	BANGALORE	5
1006	NATURALS	NAT6@GMAIL	9111111111	DELHI	6
1007	VERSURE	VER7@GMAIL	9865112111	DELHI	7
1008	BIOLEAP	BIO8@GMAIL	9999511012	MUMBAI	8
1009	BORAX	BOR2@GMAIL	9006012018	PUNE	2

Table 4.4 – Manufacturer Table

5) Mediquipment :-

```
create table mediquipment
(
code number(5) primary key,
Trade_name varchar2(20),
Product_Type varchar2(20),
Mfg_date date,
Exp_date date,
price number(10,2),
cid references manufacturer(cid) on delete set null);
```

CODE	TRADE_NAME	PRODUCT_TYPE	MFG_DATE	EXP_DATE	PRICE	CID
12122	PARACETAMOL	CAPSULE	06-AUG-16	20-AUG-20	38	1001
12100	QUINAPRIL	TABLETS	07-AUG-16	19-AUG-21	56	1003
12115	ZAFIRLUKAST	TABLETS	08-SEP-16	17-SEP-19	100	1004
12116	CROCIN	SYRUP	09-MAY-16	16-MAY-18	110	1005
12117	INJECTION	OPERATION	18-JUN-16	23-APR-23	112	1006
12109	BANDAGE	CLOTH	19-SEP-16	18-AUG-19	112.07	1006
12121	OXYGEN MASK	OPERATION	25-JUL-16	28-DEC-17	212.12	1007
12120	DALACIN T	GEL	28-OCT-16	21-JAN-21	303.08	1008
12112	PARAMAX	TABLETS	29-APR-16	20-FEB-20	308.16	1009
12104	VINYL	SYRUP	13-SEP-16	13-MAR-18	113.18	1003

Table 4.5 – Mediquipment Table

6) Supplier :-

```
create table supplier
(
name varchar2(20),
city varchar2(20),
mobile integer primary key,
email varchar2(20),
cid references manufacturer(cid) on delete set null,
pharid references pharmacy(phid) on delete set null);
```

NAME	CITY	MOB	EMAIL	CID	PHID
PRIYANK	BANGALORE	9812131212	PRI1@GMAIL	1001	1
TRISHAL	MUMBAI	9811210000	TRI2@GMAIL	1002	2
RAJ	PUNE	9800100211	RAJ3@GMAIL	1003	3
ARYAN	CHENNAI	9761610610	ARY4@GMAIL	1004	4
VISHAK	HYDERABAD	9661252213	VIS5@GMAIL	1005	5
SHUBHAM	SURAT	9012131131	SHU6@GMAIL	1006	6
ROHIT	AHMEDABAD	9002112117	ROH7@GMAIL	1007	7
RISHABH	BHOPAL	9811011021	RIS8@GMAIL	1008	8
ANSHUMAN	JAIPUR	9811300200	ANS4@GMAIL	1009	9

Table 4.6 – Supplier Table

7) Employee :-

```
create table employee
(
name varchar2(20),
city varchar2(20),
doj date,
mobile integer primary key,
salary number(10),
age number(2),
sex varchar2(1),
pharid references pharmacy(phid) on delete set null);
```

NAME	CITY	DOJ	MOB	SALARY	AGE	SEX	PHID
ARJUN	BANGALORE	02-AUG-14	9110220330	15000	23	M	1
ANKITA	MUMBAI	03-MAR-13	9112221312	20000	24	F	2
GANESH	CHENNAI	03-APR-14	9112222331	24000	25	M	3
MAYUR	DELHI	02-MAY-14	9113114118	25000	26	M	4
MAYANK	BANGALORE	05-JUN-15	9116100200	26000	25	M	5
SATYAM	HYDERABAD	06-DEC-16	9081812812	27000	26	M	6
ABHISHEK	SURAT	07-SEP-15	9081813813	18000	27	M	7
DANIEL	JAIPUR	08-JUL-14	9112134168	19000	21	M	8
RON	DISPUR	09-SEP-13	901812113	25000	28	M	9

Table 4.7 – Employee Table

8) Hospital :-

```
create table hospital
(
hid number(2) primary key,
name varchar2(20),
email varchar2(40),
phone integer,
city varchar2(20),
pharid references pharmacy(phid) on delete set null);
```

HID	NAME	EMAIL	MOB	CITY	PHID
50	KRISHNA	KRI6@GMAIL	9771700700	DELHI	1
51	JAYADEVA	JAY5@GMAIL	9772800800	BANGALORE	2
52	APOLLO	APO7@GMAIL	9812900912	MYSORE	3
53	PARAS	PAR8@GMAIL	9310315315	DELHI	3
54	GETWELLO	GET9@GMAIL	9410055000	CHENNAI	4
55	ARUNODAYA	ARU8@GMAIL	9610066100	MUMBAI	5
56	AASTHA	AAS1@GMAIL	9780078001	PUNE	6
57	DEVAGIRI	DEV7@GMAIL	9871001002	SURAT	7
58	SHANTI	SHA8@GMAIL	9860010081	BHOPAL	8
59	LAKSHMI	LAK1@GMAIL	9861001012	KOLKATA	9

Table 4.8 –Hospital Table

9) Bill :-

```
create table bill
(
bid number(3) primary key,
DOB date,
age number(3),
pname varchar2(20),
mobile integer,
city varchar2(20),
product varchar2(20),
amount number(10,2),
pharid references pharmacy(phid) on delete set null);
```

BID	DOB	AGE	PNAME	MOBILE	CITY	PRODUCT	AMOUNT	PHAR ID
300	01-SEP-17	20	ANSHU	9854575613	BANGALORE	QUINAPRIL	56	1
301	02-SEP-17	25	IQBAL	8755425417	BANGALORE	OXYGEN MASK	212.12	1
302	03-AUG-17	35	PRIYA	7611413161	MUMBAI	CROCIN	110	1
303	01-JAN-17	45	KAUSHIK	9153484335	DELHI	INJECTION	112	1
304	02-FEB-17	65	NITU	7535413431	CHENNAI	DALACIN T	303.08	1
305	08-FEB-17	75	RESHMA	8351354134	MUMBAI	CROCIN	110	1
306	09-FEB-17	85	REKHA	9354533131	MYSORE	BANDAGE	112.07	1
307	07-MAR-17	95	ALISHA	8453431211	DELHI	PARAMAX	308.16	1
308	08-APR-17	44	SONIA	7153131323	MYSORE	VINYL	113.18	1
309	08-APR-17	26	SONIA	9534351743	DELHI	CROCIN	110	1

Table 4.9 – Bill Table

10) Works :-

```
create table works
(
pharid references pharmacy(phid) on delete cascade,
start_date date,
end_date date,
primary key(pharid));
```

PHID	START_DATE	END_DATE
1	01-AUG-16	19-MAR-19
2	02-AUG-16	20-MAR-19
3	03-AUG-16	21-MAR-19
4	04-MAY-16	22-MAR-19
5	05-MAY-16	23-MAR-19
6	06-JUN-16	26-OCT-19
7	21-JUL-15	28-OCT-19
8	21-SEP-16	29-OCT-19
9	22-SEP-16	23-MAR-19

Table 4.10 – Works Table

11) Contract :-

```
create table contract
(
pharid references pharmacy(phid) on delete cascade,
cid references manufacturer(cid) on delete cascade,
start_date date,
end_date date,
primary key(pharid,cid));
```

PHID	CID	START_DATE	END_DATE
1	1000	01-JAN-15	11-SEP-19
2	1001	01-JAN-15	21-SEP-19
3	1002	02-JAN-16	01-OCT-19
4	1003	03-MAR-16	02-OCT-19
5	1004	03-MAR-16	03-OCT-19
6	1005	04-APR-15	04-DEC-19
7	1006	05-APR-14	08-DEC-20
8	1004	06-APR-15	09-NOV-20
9	1002	10-JUN-15	08-NOV-19

Table 4.11 – Contract Table

12) Prescribe :-

```
create table prescribe
(
DOP date,
        medicine varchar2(20),
        did references doctor(did) on delete cascade,
        pid references customer(pid) on delete cascade,
        primary key(pid,did));
```

GIVEN_DATE	MEDICINE	DID	PID
01-SEP-17	PARACETAMOL	10	101
02-SEP-17	CROCIN	11	102
03-AUG-17	CROCIN	12	103
01-JAN-17	BANDAGE	13	105
02-FEB-17	OXYGEN MASK	14	106
08-FEB-17	VINYL	15	107
07-MAR-17	PARAMAX	16	108
21-NOV-17	DALACIN T	17	109
22-NOV-17	QUINAPRIL	18	104

Table 4.12 – Prescribe Table

QUERY'S :-

```
1) Select name,salary
   From employee
   Group by name,salary
   Having salary > (select avg(salary)
               from employee
               where city='bangalore')
   Order by 2 Asc;
2) Update mediquipment set price=500
   Where cid in (Select cid
                   From manufacturer
                   Where city='pune');
3) Select p.name,pharid,count(distinct cid) as count1
   From pharmacy p, manufacturer
   Where phid = pharid
   Group by p.name, pharid
   Having count(*) > 1;
4) Select p.name, e.name, e.salary
   From employee e, pharmacy p
```

```
 Where phid=pharid
 Minus
 Select p.name,e.name, e.salary
 From employee e, pharmacy p
 Where e.sex='f';
```

TRIGGER'S :-

```
1) Create trigger price_check
   Before insert or update on mediquipment
   For each row
   When (new.price>500)
   Begin
   :new.price:= 499;
   End;
   /
2) Create trigger age_check1
  Before insert or update on doctor
 For each row
 When (new.age>80)
 Begin
 Dbms_output.put_line ('DOCTOR AGE CANNOT BE MORE THAN 80!!!');
 End;
 /
```

PROCEDURE'S :-

```
1)Create procedure update_medi as
 Begin
 Update mediquipment set price = price+100 where trade_name = 'paramax';
 End;
 /
```

CONNECTIVITY OF FRONT END TO BACK END :-

The fundamental steps involved in the process of connecting to a database and executing a query consist of the following:

- Import JDBC packages.
- Load and register the JDBC driver.

- Open a connection to the database.
- Create a statement object to perform a query.
- Execute the statement object and return a query resultset.
- Process the result set.
- Close the result set and statement objects.
- Close the connection.

These steps are described in detail in the sections that follow.

Import JDBC Packages

This is for making the JDBC API classes immediately available to the application program. The following import statement should be included in the program irrespective of the JDBC driver being used:

```
import.java.sql.*;
```

Load and Register the JDBC Driver

This is for establishing a communication between the JDBC program and the Oracle database. This is done by using the static registerDriver() method of the DriverManager class of the JDBC API. The following line of code does this job:

```
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
```

JDBC Driver Registration

For the entire Java application, the JDBC driver is registered only once per each database that needs to be accessed. This is true even when there are multiple database connections to the same data server.

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

Connecting to a Database

Once the required packages have been imported and the Oracle JDBC driver has been loaded and registered, a database connection must be established. This is done by using

the getConnection()method of the DriverManager class. A call to this method creates an object instance of thejava.sql.Connection class. The getConnection() requires three input parameters, namely, a connect string, a username, and a password. The connect string should specify the JDBC driver to be yes and the database instance to connect to.
The getConnection() method is an overloaded method that takes

- Three parameters, one each for the URL, username, and password.
- Only one parameter for the database URL. In this case, the URL contains the username and password.

The following lines of code illustrate using the getConnection() method:
Connection conn = DriverManager.getConnection(URL, username, passwd);
where URL, username, and passwd are of String data types.

Querying the Database

Querying the database involves two steps: first, creating a statement object to perform a query, and second, executing the query and returning a resultset.

Creating a Statement Object

This is to instantiate objects that run the query against the database connected to. This is done by the createStatement() method of the conn Connection object created above. A call to this method creates an object instance of the Statement class. The following line of code illustrates this:
Statement sql_stmt = conn.createStatement();

Executing the Query and Returning a ResultSet

Once a Statement object has been constructed, the next step is to execute the query. This is done by using the executeQuery() method of the Statement object. A call to this method takes as parameter a SQL SELECT statement and returns a JDBC ResultSet object. The following line of code illustrates this using the sql_stmt object created above:

ResultSet rset = sql_stmt.executeQuery("SELECT empno, ename, sal, deptno FROM emp ORDER BY ename");

Processing the Results of a Database Query That Returns Multiple Rows

Once the query has been executed, there are two steps to be carried out:

- Processing the output result set to fetch the rows

- Retrieving the column values of the current row

The first step is done using the next() method of the ResultSet object. A call to next() is executed in a loop to fetch the rows one row at a time, with each call to next() advancing the control to the next available row. The next() method returns the Boolean value true while rows are still available for fetching and returns false when all the rows have been fetched.
The second step is done by using the getXXX() methods of the JDBC resultset object. Here getXXX()corresponds to the getInt(), getString() etc with XXX being replaced by a Java datatype.
Closing the ResultSet and Statement
Once the ResultSet and Statement objects have been used, they must be closed explicitly. This is done by calls to the close() method of the ResultSet and Statement classes.
If not closed explicitly, there are two disadvantages:

- Memory leaks can occur
- Maximum Open cursors can be exceeded

Closing the ResultSet and Statement objects frees the corresponding cursor in the database.
Closing the Connection
The last step is to close the database connection opened in the beginning after importing the packages and loading the JDBC drivers. This is done by a call to the close() method of the Connection class.
The following line of code does this:

```
conn.close();
```

CHAPTER 5

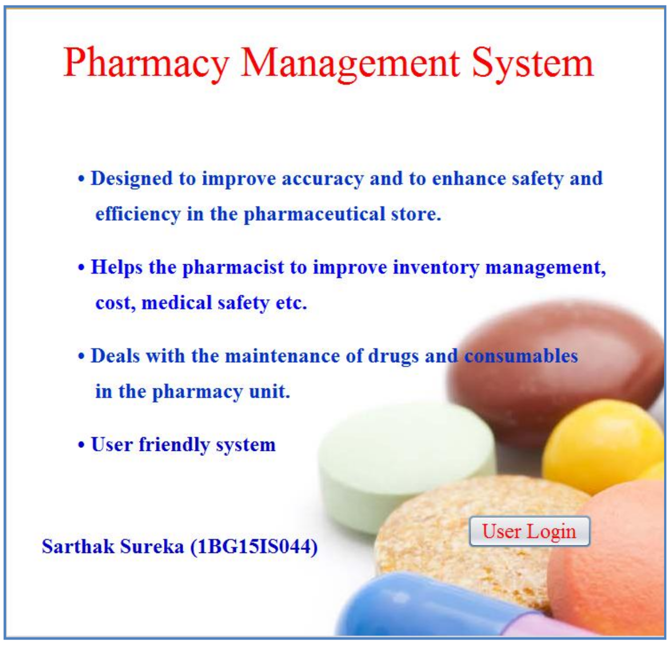
SNAPSHOTS

In computer systems, a snapshot is the state of a system at a particular point in time. The term was coined as an analogy to that in photography. It can refer to an actual copy of the state of a system or to a capability provided by certain systems.

FRONT END :-

1) Introduction Page :-

   This page gives a brief idea of the project and how it can be used, there is a user login button that lets the user sign into the system, so that the data can be modified.



Pharmacy Management System

- Designed to improve accuracy and to enhance safety and efficiency in the pharmaceutical store.
- Helps the pharmacist to improve inventory management, cost, medical safety etc.
- Deals with the maintenance of drugs and consumables in the pharmacy unit.
- User friendly system

Sarthak Sureka (1BG15IS044)

Fig.5.1 – Introduction Page

2) Login Page :-

   This page asks the user to enter the username and password to login into the system.

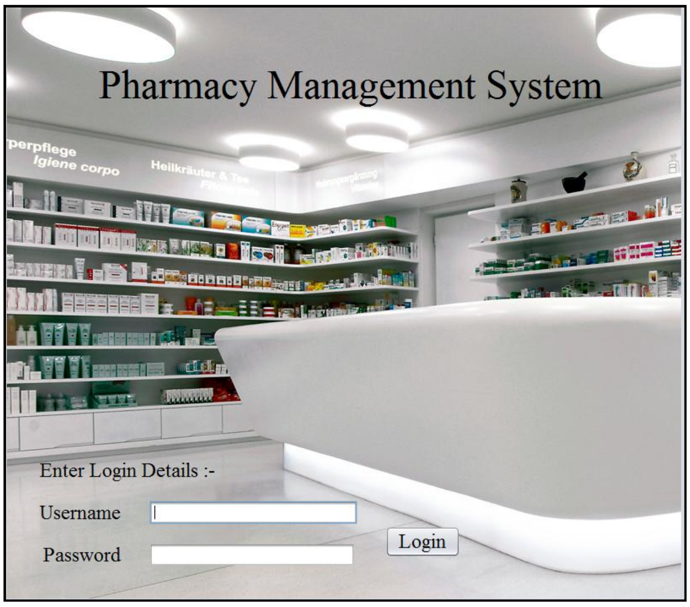


Fig.5.2 – Login Page

3) Option Page :-

This page lets the user select from the different options available in the form.

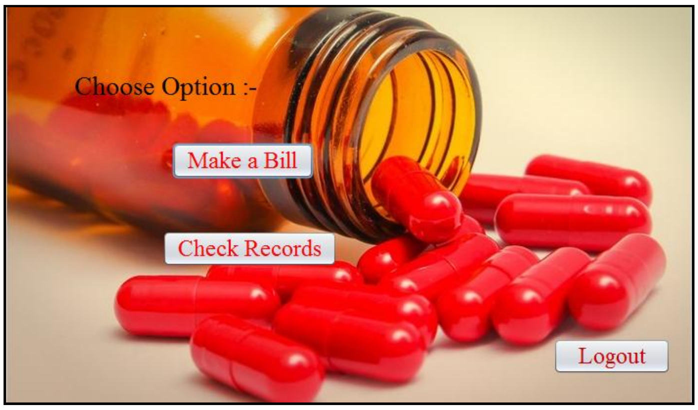


Fig.5.3 – Status Page

4) Billing Page :-

This page allows the user to generate a bill and save the bill, once all the data that are asked in the form are inserted.

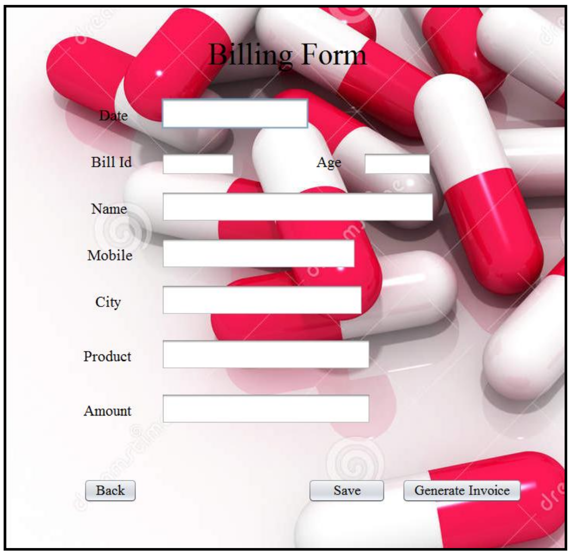
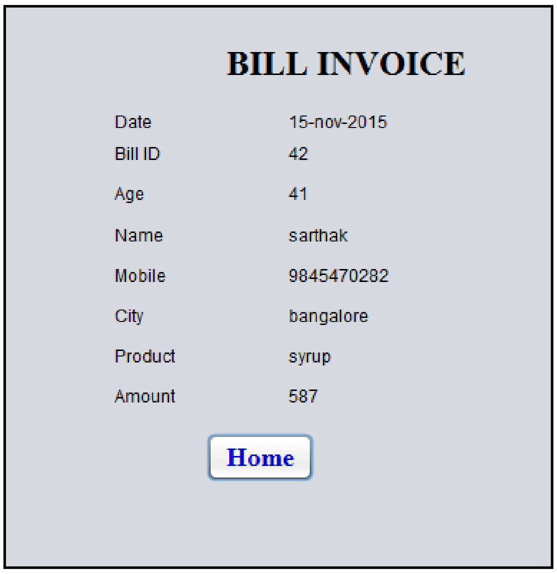


Fig.5.4 – Billing Page

5) Bill Invoice Page :-

This page displays all the data that the user has entered in the billing page.



BILL INVOICE

Date	15-nov-2015
Bill ID	42
Age	41
Name	sarthak
Mobile	9845470282
City	bangalore
Product	syrup
Amount	587

Fig.5.5 – Bill Invoice

6) Category Page :-

This page displays the different categories that are available in the system and are connected to the database.

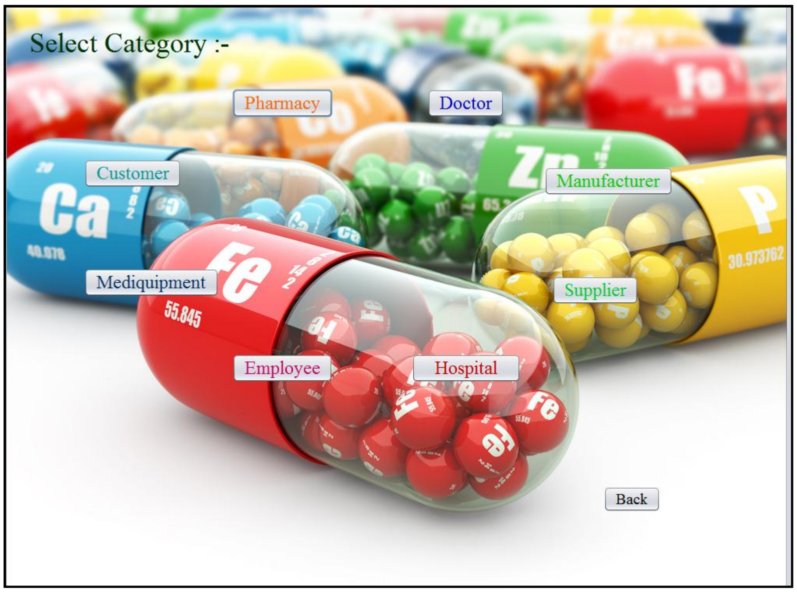
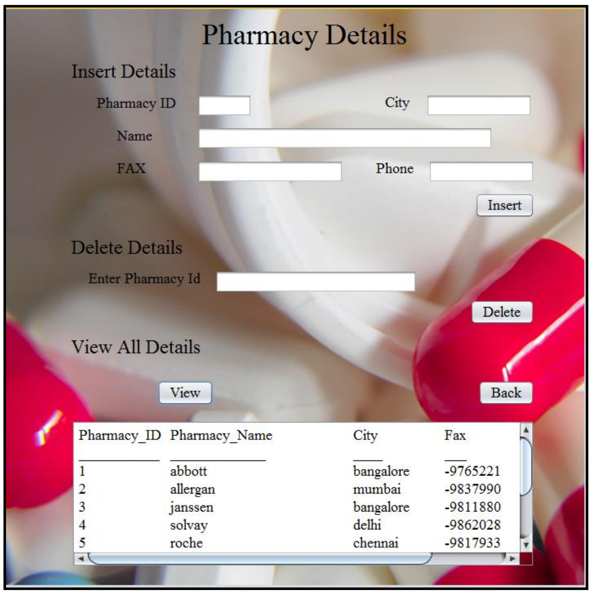


Fig.5.6 – Category Page

7) Pharmacy Page :-

   This page lets the user to insert data into the database from the front end, delete the data that the user wants to delete and view the database that is stored in the database.



```
Pharmacy_ID  Pharmacy_Name         City        Fax
___________  _____________         ____        ___
1            abbott                bangalore   -9765221
2            allergan              mumbai      -9837990
3            janssen               bangalore   -9811880
4            solvay                delhi       -9862028
5            roche                 chennai     -9817933
```

Fig.5.7 – Pharmacy Page

8) Doctor Page :-

   This page lets the user to insert data into the database from the front end, delete the data that the user wants to delete and view the database that is stored in the database.

9) Customer Page :-

This page lets the user to insert data into the database from the front end, delete the data that the user wants to delete and view the database that is stored in the database.

10) Manufacturer Page :-

This page lets the user to insert data into the database from the front end, delete the data that the user wants to delete and view the database that is stored in the database.

11) Mediquipment Page :-

This page lets the user to insert data into the database from the front end, delete the data that the user wants to delete and view the database that is stored in the database.

12) Supplier Page :-

This page lets the user to insert data into the database from the front end, delete the data that the user wants to delete and view the database that is stored in the database.

13) Employee Page :-

This page lets the user to insert data into the database from the front end, delete the data that the user wants to delete and view the database that is stored in the database.

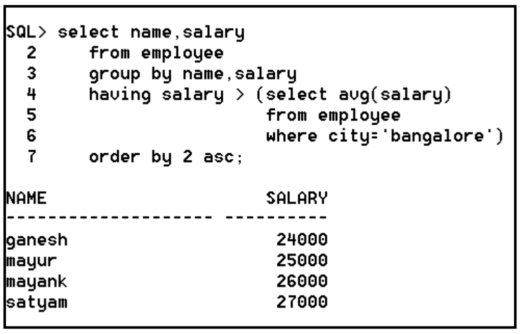
14) Hospital Page :-

This page lets the user to insert data into the database from the front end, delete the data that the user wants to delete and view the database that is stored in the database.

BACK END :-

Query :-

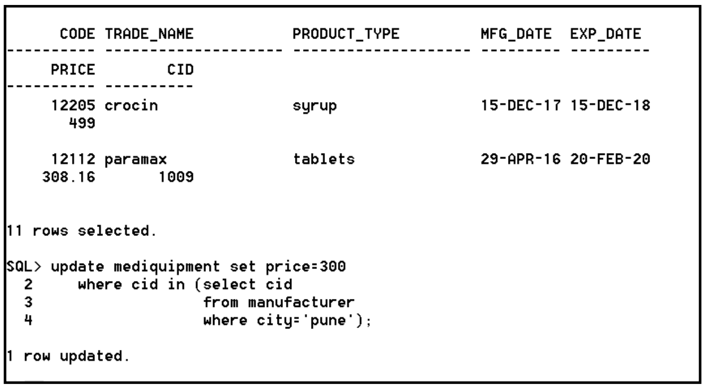
1) This query displays the name and salary of all employee's whose salary is greater than the average salary of all employee's working in Bangalore.



```
SQL> select name,salary
  2     from employee
  3     group by name,salary
  4     having salary > (select avg(salary)
  5                      from employee
  6                      where city='bangalore')
  7     order by 2 asc;

NAME                     SALARY
-------------------- ----------
ganesh                    24000
mayur                     25000
mayank                    26000
satyam                    27000
```

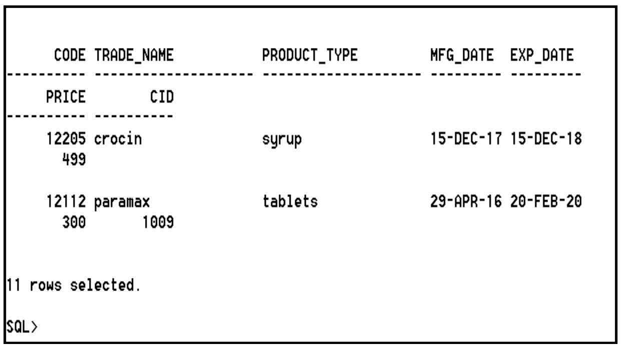
Fig.5.8 – Query 1

2) This query updates the medicine's price in mediquipment table to 300, if the manufacturer is from Pune.



```
      CODE TRADE_NAME           PRODUCT_TYPE         MFG_DATE  EXP_DATE
---------- -------------------- -------------------- --------- ---------
     PRICE        CID
---------- ----------
     12205 crocin               syrup                15-DEC-17 15-DEC-18
       499

     12112 paramax              tablets              29-APR-16 20-FEB-20
    308.16       1009


11 rows selected.

SQL> update mediquipment set price=300
  2     where cid in (select cid
  3                   from manufacturer
  4                   where city='pune');

1 row updated.
```

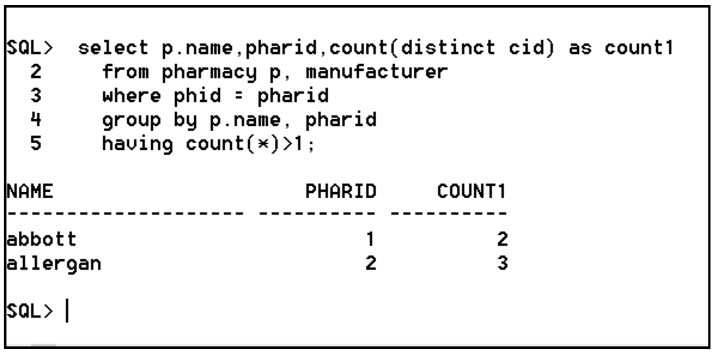
Fig.5.9(a) – Query 2 (Before execution)



```
      CODE TRADE_NAME           PRODUCT_TYPE         MFG_DATE  EXP_DATE
---------- -------------------- -------------------- --------- ---------
     PRICE        CID
---------- ----------
     12205 crocin               syrup                15-DEC-17 15-DEC-18
       499

     12112 paramax              tablets              29-APR-16 20-FEB-20
       300       1009


11 rows selected.

SQL>
```

Fig.5.9(b) – Query 2 (After execution)

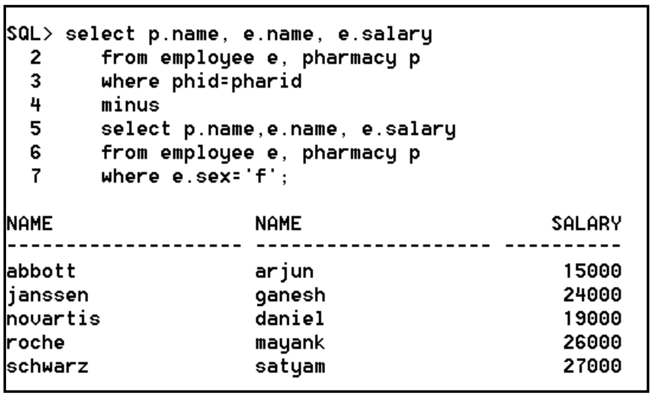
3) This query displays the name of pharmacy, pharmacy id and the count of the manufacturer id, if a pharmacy is taking medicine's from more than one manufacturer.



```
SQL>  select p.name,pharid,count(distinct cid) as count1
  2     from pharmacy p, manufacturer
  3     where phid = pharid
  4     group by p.name, pharid
  5     having count(*)>1;

NAME                     PHARID     COUNT1
-------------------- ---------- ----------
abbott                        1          2
allergan                      2          3

SQL> |
```

Fig.5.10 – Query 3

4) This query displays the pharmacy name, employee name and employee salary, who are working in a pharmacy except the female employee.

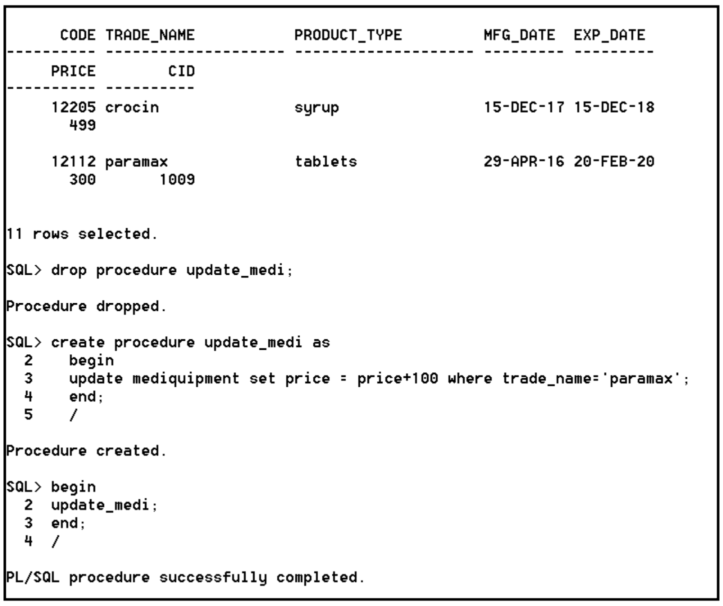


```
SQL> select p.name, e.name, e.salary
  2     from employee e, pharmacy p
  3     where phid=pharid
  4     minus
  5     select p.name,e.name, e.salary
  6     from employee e, pharmacy p
  7     where e.sex='f';

NAME                 NAME                     SALARY
-------------------- -------------------- ----------
abbott               arjun                     15000
janssen              ganesh                    24000
novartis             daniel                    19000
roche                mayank                    26000
schwarz              satyam                    27000
```

Fig.5.11 – Query 4

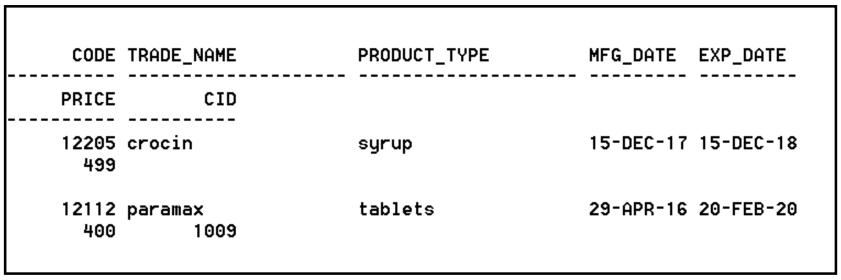
PROCEDURE :-

This procedure update's the medicine price by 100 in the mediquipment table, where the trade name is paramax.



```
      CODE TRADE_NAME           PRODUCT_TYPE         MFG_DATE  EXP_DATE
---------- -------------------- -------------------- --------- ---------
     PRICE        CID
---------- ----------
     12205 crocin               syrup                15-DEC-17 15-DEC-18
       499

     12112 paramax              tablets              29-APR-16 20-FEB-20
       300       1009


11 rows selected.

SQL> drop procedure update_medi;

Procedure dropped.

SQL> create procedure update_medi as
  2    begin
  3    update mediquipment set price = price+100 where trade_name='paramax';
  4    end;
  5    /

Procedure created.

SQL> begin
  2  update_medi;
  3  end;
  4  /

PL/SQL procedure successfully completed.
```

Fig.5.12(a) – Update_Medi (Before procedure is executed)

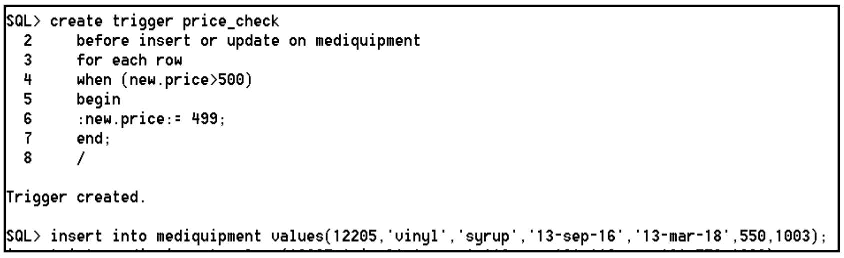


```
      CODE TRADE_NAME           PRODUCT_TYPE         MFG_DATE  EXP_DATE
---------- -------------------- -------------------- --------- ---------
     PRICE        CID
---------- ----------
     12205 crocin               syrup                15-DEC-17 15-DEC-18
       499

     12112 paramax              tablets              29-APR-16 20-FEB-20
       400       1009
```

Fig.5.12(b) – Update_Medi (After procedure is executed)

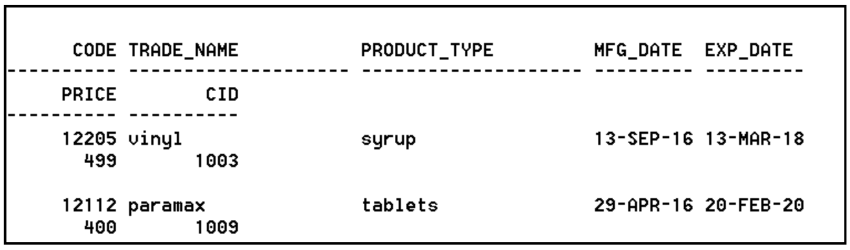
TRIGGER :-

1) This trigger will change the price of a medicine to 499, if the price is greater than 500.



```
SQL> create trigger price_check
       before insert or update on mediquipment
       for each row
       when (new.price>500)
       begin
       :new.price:= 499;
       end;
       /

Trigger created.

SQL> insert into mediquipment values(12205,'vinyl','syrup','13-sep-16','13-mar-18',550,1003);
```

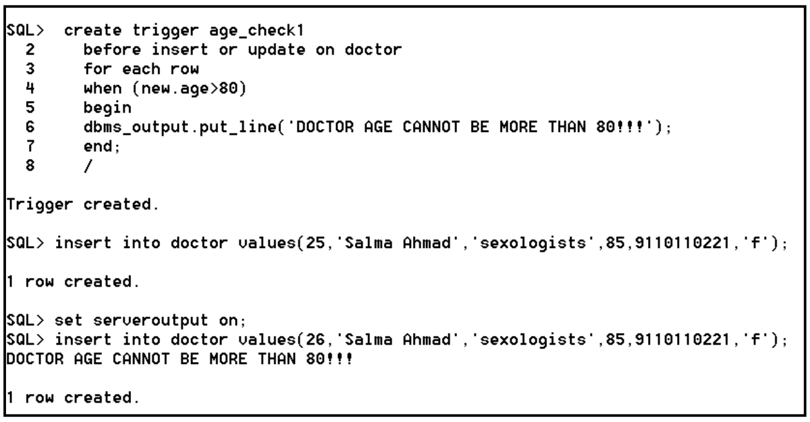
Fig.5.13(a) – Price Check Trigger (Trigger is executed and a insertion is performed)



```
      CODE TRADE_NAME           PRODUCT_TYPE         MFG_DATE  EXP_DATE
---------- -------------------- -------------------- --------- ---------
     PRICE        CID
---------- ----------
     12205 vinyl                syrup                13-SEP-16 13-MAR-18
       499       1003

     12112 paramax              tablets              29-APR-16 20-FEB-20
       400       1009
```

Fig.5.13(b) – Price Check Trigger (Inserted price is less than 500)

2) This trigger will display a message if the age of a doctor is greater than 80.



```
SQL>  create trigger age_check1
        before insert or update on doctor
        for each row
        when (new.age>80)
        begin
        dbms_output.put_line('DOCTOR AGE CANNOT BE MORE THAN 80!!!');
        end;
        /

Trigger created.

SQL> insert into doctor values(25,'Salma Ahmad','sexologists',85,9110110221,'f');

1 row created.

SQL> set serveroutput on;
SQL> insert into doctor values(26,'Salma Ahmad','sexologists',85,9110110221,'f');
DOCTOR AGE CANNOT BE MORE THAN 80!!!

1 row created.
```

Fig.5.14 – Age Check Trigger

CHAPTER 6

CONCLUSION

A Pharmacy Management Database is created to handle various operations related to different people or organizations. Different persons like Doctors, Customers, Suppliers, Manufacturers and Employees working in a pharmacy are considered for developing this project. Other important attributes related to a pharmacy like Mediquipments and Bill and organizations like Hospital are merged in one database to show the relations of these entities and to retrieve data from them whenever deemed necessary.

Advantages and Applications of this project are:

1) This Pharmacy Management system helps to make the system paper less and more efficient.
2) It is user friendly and simple.
3) It helps the manager of the pharmacy to keep a track of all mediquipments sold and purchased.
4) It helps to easily access medicines in pharmacy and calculate the income of pharmacy.
5) Manager can use this database to contact manufacturers, suppliers of a particular medicine without going through a complete register of records.
6) It is a very convenient method to retrieve information related to a medicine or any other entity related to pharmacy.

Limitations:

1) Detailed information and gathering has to be done to obtain satisfactory results.
2) Implementing the software requires change in business practices.
3) Implementation and maintenance costs run very high.

Chapter 7

REFERENCES

Text Books:

1. Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
2. Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill.
3. Herbert Schildt: JAVA the Complete Reference, 7th/9th Edition, Tata McGraw Hill, 2007.
4. Jim Keogh: J2EE-TheCompleteReference, McGraw Hill, 2007.

Websites:

1. www.erdplus.com
2. https://netbeans.org/kb/docs/java/gui-functionality.html
3. http://www.oracle-dba-online.com/sql/oracle_sql_tutorial.html
4. https://netbeans.org/kb/docs/ide/oracle-db.html