DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By PES1201800269

PES1201800269 **MADDI SIDDART**

CLINIC MANAGEMENT SYSTEM

SUMMARY

The Project aims to implement a database infrastructure for clinic management systems . The database stores information of the patients , manufacturer of the medicines, doctor's information , payment details of the patients etc. It stores the information of all the appointments taken by the patients. Each entity has a primary key to be used as an identifier .

The project has a trigger implementation to enforce some constraints in the schema as well as referential integrity constraints between all tables .We have the check of lossless decomposition and we have also implemented the creation of tables using check constraints. In the end we retrieve some data using complex sql queries and try to analyse the data .

The project is implemented in Postgresql running on Mac OS.

INTRODUCTION

A **Clinic Management System** is an integrated information **system** for managing all aspects of a medical **clinic's** operations such as medical, administrative.
Here we deal with Patients details ,Health professional's details,manufacturer's details etc.

We begin with making a Conceptual model and by drawing ER DIAGRAM we can understand the relations between the entities present in the database. We can look at the
primary keys and other constraints like type of relation such as 1:N ,M:N etc.

PATIENT :

This relation has the data about of the patient such as his name , phno email etc.

APPOINTMENT :

This relations deals with the appointments of the patients and tells us the information about the data of appointment and which doctor the
is appointment to a specific patient and what is the room his appointment is in .

HEALTH_PROFESSIONAL:

This relation talks about the details of the doctors such as their name, title etc .

PRESCRIPTION :

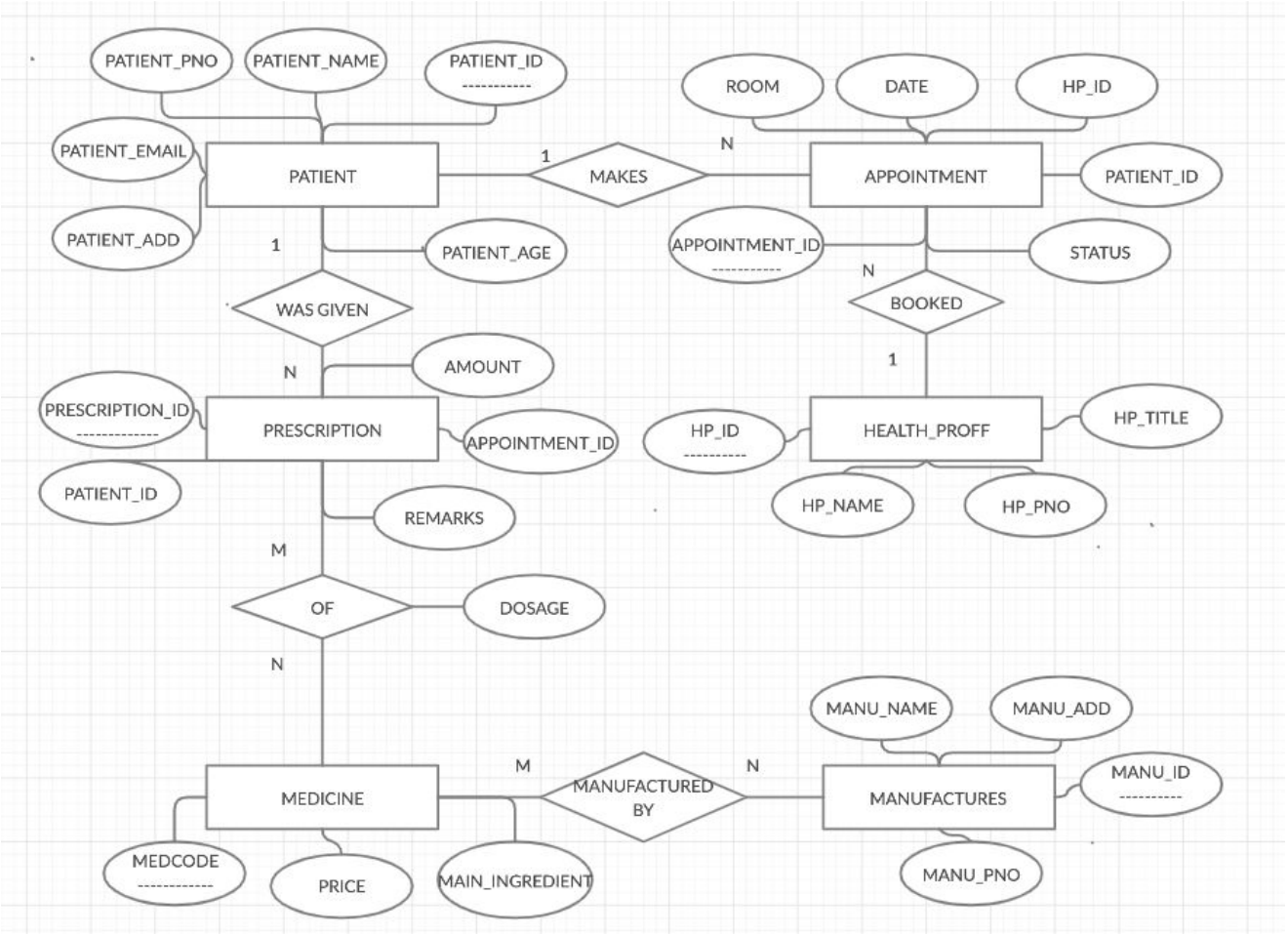
This relation is mainly about the amount paid by the patient that is mentioned in the prescription and talks about the remarks by the doctors.

MEDICINE: Details of the medicines

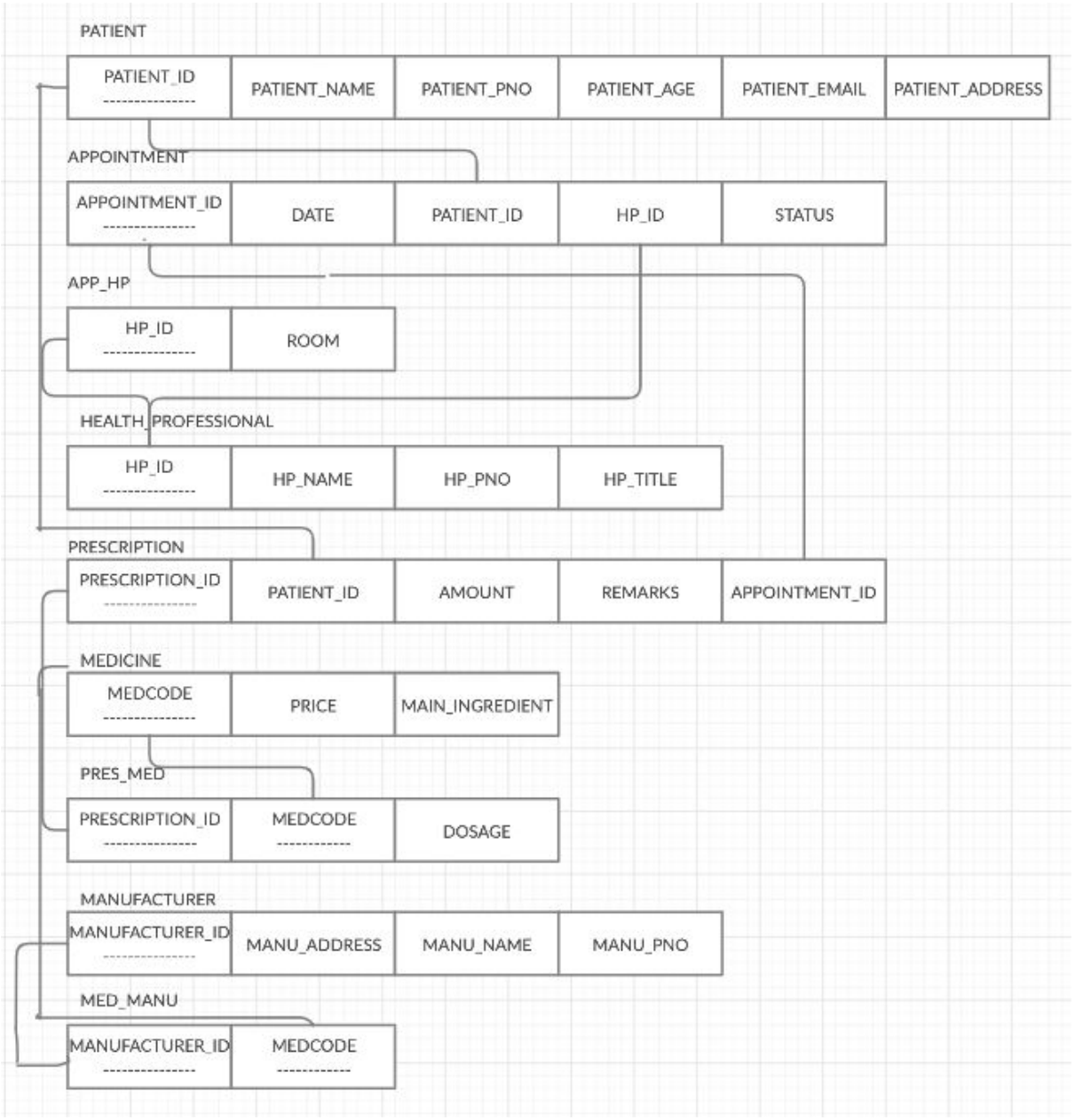
MANUFACTURER: Details of the medicine manufacturers.

DATA MODEL:

ER MODEL:



RELATIONAL MODEL SCHEMA:



FUNCTIONAL DEPENDENCIES AND NORMALISATION:

IN RELATION **PATIENT**:

PATIENT_ID -> { PATIENT_NAME, PATIENT_PNO, PATIENT_AGE, PATIENT_EMAIL,

PATIENT_ADDRESS }

WHICH IMPLIES PATIENT_ID IS THE PRIMARY KEY WHICH DETERMINES ALL.

IN RELATION **HEALTH PROFESSIONAL**:

HP_ID ->{ HP_NAME, HP_TITLE , HP_PNO }

HP_ID IS THE PRIMARY KEY

IN RELATION **APPOINTMENT** :

APPOINTMENT_ID -> { DATE, HP_ID,PATIENT_ID,STATUS , ROOM }

HP_ID -> { ROOM }

APPOINTMENT_ID IS THE PRIMARY KEY
IT VIOLATES 3NF
AND HERE TRANSITIVE DEPENDENCY BETWEEN {
APPOINTMENT_ID,ROOM,HP_ID}
HENCE WE HAVE TO NORMALISE(3NF)

WE FIRST SPLIT IT INTO TWO TABLES
SUCH AS ONE HAS : DATE,HP_ID,PATIENT_ID,STATUS
OTHER HAS: ROOM ,HP_ID

IN RELATION **PRESCRIPTION**:

PRESCRIPTION_ID -> { PATIENT_ID, AMOUNT , APPOINTMENT_ID, REMARKS }

CANDIDATE KEY = appointment_id

PRIMARY KEY = PRESCRIPTION_ID

IN RELATION **MEDICINE**:

MEDCODE -> { PRICE , MAIN_INGREDIENT }

HENCE MEDCODE IS THE PRIMARY KEY .

IN RELATION **MANUFACTURER**:

MANUFACTURE_ID -> { MANUFACTURER->NAME, MANUFACTURER->ADDRESS,
MANUFACTURER_PNO}

HERE MANUFACTURER_ID IS THE PRIMARY KEY.

NORMALIZATION :

1) All relations are in first normal form as there are no non atomic attribute values in any relation.

2) All relations are in second normal form as no attribute is dependent on the proper subset of a super key in a relation.

3) All relations are in third normal form as there are not transitive dependencies left in any relation.

TEST FOR LOSSLESS JOIN PROPERTY :

FOR THE DECOMPOSITION OF APPOINTMENT :

relation App_id Date room patient_id hp_id status

R1	a1	a2	**b13**	a4	**a5**	a6
R2	b21	b22	**a3**	b24	**a5**	b26

The attribute sets are {Appointment_id,date,patient_id,hp_id,status} , { room,hp_id} the union is the set of all attributes and intersection is the key of hp_id.

hence by the algorithm
we get :

R1	a1	a2	~~b13~~ a3	a4	**a5**	a6
R2	b21	b22	**a3**	b24	**a5**	b26

As all values are ‘a’ values in the first row the decomposition is lossless.

DATA DEFINITION LANGUAGE SCRIPTS

```
CREATE TABLE IF NOT EXISTS PATIENT(
    PATIENT_ID INTEGER PRIMARY KEY,
    PATIENT_NAME VARCHAR(80) NOT NULL,
    PATIENT_PNO BIGINT NOT NULL CHECK (PATIENT_PNO>=1000000000 AND PATIENT_PNO<=9999999999),
    PATIENT_ADDRESS VARCHAR(255) NOT NULL,
    PATIENT_EMAIL VARCHAR(80) NOT NULL,
    PATIENT_AGE INT NOT NULL
);
CREATE TABLE IF NOT EXISTS HEALTH_PROFFESIONAL(
    HP_ID INTEGER PRIMARY KEY,
    HP_NAME VARCHAR(80) NOT NULL,
    HP_PNO BIGINT NOT NULL CHECK (HP_PNO>=1000000000 AND HP_PNO<=9999999999),
    TITLE VARCHAR(80) NOT NULL
);

CREATE TABLE IF NOT EXISTS APPOINTMENT(
    APPOINTMENT_ID INTEGER PRIMARY KEY ,
    APPOINTMENT_DATE DATE NOT NULL,
    PATIENT_ID INTEGER REFERENCES PATIENT(PATIENT_ID) ON DELETE CASCADE,
    HP_ID INTEGER REFERENCES HEALTH_PROFFESIONAL(HP_ID) ON DELETE CASCADE,
    STATUS INTEGER DEFAULT 0
);
CREATE TABLE IF NOT EXISTS APP_HP (
    HP_ID INTEGER PRIMARY KEY,
    ROOM INTEGER NOT NULL
);
CREATE TABLE IF NOT EXISTS PRESCRIPTION(
    PRESCRIPTION_ID INTEGER PRIMARY KEY,
    PATIENT_ID INTEGER REFERENCES PATIENT(PATIENT_ID) ON DELETE CASCADE,
    APPOINTMENT_ID INTEGER REFERENCES APPOINTMENT(APPOINTMENT_ID) ON DELETE CASCADE,
    AMOUNT INTEGER NOT NULL,
    REMARKS VARCHAR(255) NOT NULL
);
```

```
CREATE TABLE IF NOT EXISTS MEDICINE(
    MEDCODE INTEGER PRIMARY KEY,
    PRICE INTEGER NOT NULL,
    MAIN_INGREDIENT VARCHAR(80) NOT NULL
);

CREATE TABLE IF NOT EXISTS PRES_MED(
    PRESCRIPTION_ID INTEGER REFERENCES PRESCRIPTION(PRESCRIPTION_ID) ON DELETE CASCADE,
    MEDCODE INTEGER REFERENCES MEDICINE(MEDCODE) ON DELETE CASCADE,
    DOSAGE VARCHAR(80) NOT NULL,
    PRIMARY KEY(PRESCRIPTION_ID,MEDCODE)
);
CREATE TABLE IF NOT EXISTS MANUFACTURER(
    MANUFACTURER_ID INTEGER PRIMARY KEY,
    MANUFACTURER_NAME VARCHAR(80) NOT NULL,
    MANUFACTURER_ADDRESS VARCHAR(255) NOT NULL,
    MANUFACTURER_PNO BIGINT NOT NULL CHECK (MANUFACTURER_PNO >=1000000000 AND MANUFACTURER_PNO<=9999999999)
);
CREATE TABLE IF NOT EXISTS MED_MANU(
    MEDCODE INTEGER REFERENCES MEDICINE(MEDCODE) ON DELETE CASCADE,
    MANUFACTURER_ID INTEGER REFERENCES MANUFACTURER(MANUFACTURER_ID) ON DELETE CASCADE,
    PRIMARY KEY(MEDCODE,MANUFACTURER_ID)
);
```

TRIGGERS :

A TRIGGER IS ATTACHED TO PRESCRIPTION TABLE SO THAT AN INSERT ON IT WILL RESULT IN A CHANGE IN STATUS ATTRIBUTE OF THE APPOINTMENT TABLE IN THE BEGINNING THE STATUS OF THE PATIENT WILL BE ‘0’ AS HE HAS TAKEN THE APPOINTMENT BUT HASN'T ATTENDED IT YET BUT ONCE HE GETS THE PRESCRIPTION HIS STATUS CHANGES TO ‘1’ WHICH DEPICTS THAT HE FINISHED HIS APPOINTMENT.

CODE :

```
CREATE OR REPLACE FUNCTION trig() RETURNS TRIGGER AS $example_table$
    BEGIN
        UPDATE APPOINTMENT SET STATUS = 1 where APPOINTMENT_ID = new.APPOINTMENT_ID;
        RETURN NEW;
    END;
$example_table$ LANGUAGE plpgsql;
 CREATE TRIGGER example_trigger AFTER INSERT ON PRESCRIPTION
FOR EACH ROW EXECUTE PROCEDURE trig();
```

SQL QUERIES :

1) AGGREGATE QUERY :

Finding the Patient that has paid the maximum amount for all his prescriptions and his total amount .

Query is as Follows :
MAX , SUM functions are used .

```
SELECT PATIENT_ID , SUM_AMT
FROM
(
    SELECT PATIENT_ID,(SUM (AMOUNT))AS SUM_AMT
    FROM PRESCRIPTION
    GROUP BY PATIENT_ID

) AS T2
INNER JOIN
(
    SELECT MAX(SUM_AMT) AS MAX_AMT
    FROM
    (
        SELECT PATIENT_ID,(SUM(AMOUNT)) AS SUM_AMT
        FROM PRESCRIPTION
        GROUP BY PATIENT_ID
    )AS T1

) AS T3
ON T2.SUM_AMT=T3.MAX_AMT;
```

2) NESTED QUERIES :

Finding the count of medicines manufactured by each manufacturer whose costs are more than 70

```
SELECT MAN.MANUFACTURER_ID,COUNT(MAN.MANUFACTURER_ID)
FROM MED_MANU AS MAN
WHERE EXISTS(
    SELECT MAN.MEDCODE
    FROM MEDICINE AS MED
    WHERE MED.PRICE >=70 AND MAN.MEDCODE=MED.MEDCODE)
GROUP BY(MAN.MANUFACTURER_ID);
```

3) OUTER JOIN QUERIES:

Medicines which are manufactured but still have never been prescribed till now.

```
SELECT M.MEDCODE,M.PRICE,M.MAIN_INGREDIENT
FROM PRES_MED AS PM
RIGHT JOIN MEDICINE AS M
ON PM.MEDCODE=M.MEDCODE
WHERE PM.MEDCODE IS NULL;
```

PATIENTS that have not taken appointments .

```
SELECT P.PATIENT_NAME
FROM PATIENT AS P
LEFT JOIN APPOINTMENT AS A
ON P.PATIENT_ID=A.PATIENT_ID
WHERE A.PATIENT_ID IS NULL;
```

CONCLUSIONS:

1) The aim of the project is to make a database that is suitable for any clinic which can be used for storage of patients data and other necessary data.
2) It has capability of implementing a trigger to track the modifications done on the database and update the changes .
3) demonstrate some data retrieval from the database and apply it to real world problems.
4) we can implement a front end web interface to it so that it can be used as a website for an clinic
5) Demonstrate normalized forms of all entities and operate on real world problems.