Database Application for Operating Theatres

The aim of this project is to create a database application specifically for the operating theatre for hospitals. Since the operating theatre is considered as a different and more specialized section of a hospital, it requires a different database only relevant to itself, without the unnecessary information such as the manager information, salaries, visitor information and address information of the patients.

In order to satisfy the needs of the operating theatre, our goal is to design a more exclusive database system. The main entity sets are:

1. PATIENT: Each patient has a unique protocol number (which is the primary key of the entity), name, age, gender, and diagnosis information stored. Since this is a database for operating theatres, each patient must undergo an operation in an available operating room at a certain date.

2. OPERATION: The names of the operations that are processed in the hospital are stored here with a unique operation id. Also, each operation has a score, used to calculate the total points of the employees. Some operations might need equipment.

3. EQUIPMENT: There might me more than one device with the same name, so each device has a unique id.

4. STAFF: Each staff has a unique id and their names are stored in the database. Each staff is either a doctor, a nurse or a technician. Every patient must be attended by a staff.

5. MEDICAL RECORD: Patients have a medical record with a unique record id, the medical record can be identified uniquely only by considering the protocol number of the patient, thus it is a week entity set.

```
CREATE TABLE PATIENT(
protocolNO INTEGER,
patient_name CHAR (50),
diagnosis VARCHAR (200),
age INTIGER,
gender CHAR (20),
PRIMARY KEY(protocolNO)
);

CREATE TABLE OPERATION(
opID INTEGER,
op_name CHAR(50),
op_point INTIGER,
```

```
PRIMARY KEY(opID)
);

CREATE TABLE STAFF(
staffID INTEGER,
staff_name CHAR (50),
PRIMARY KEY(staffID)
);

CREATE TABLE DOCTOR(
staffID INTEGER,
PRIMARY KEY(staffID),
FOREIGN KEY(staffID) REFERENCES
Staff(staffID)
ON DELETE CASCADE
);

CREATE TABLE NURSE(
staffID INTEGER,
PRIMARY KEY(staffID),
FOREIGN KEY(staffID) REFERENCES
Staff(staffID)
ON DELETE CASCADE
);

CREATE TABLE TECHNICIANS(
staffID INTEGER,
PRIMARY KEY(staffID),
FOREIGN KEY(staffID) REFERENCES
Staff(staffID)
ON DELETE CASCADE
```

```
);

CREATE TABLE EQUIPMENT(
equipID INTEGER,
equip_name VARCHAR(50),
PRIMARY KEY(equipID)
);

CREATE TABLE UNDERGOES(
protocolNO INTEGER,
opID INTEGER NOT NULL,
op_room INTEGER,
op_date DATE,
PRIMARY KEY(protocolNO, opID),
FOREIGN KEY(protocolNO) REFERENCES Patient(protocolNO) ON DELETE CASCADE,
FOREIGN KEY(opID) REFERENCES Operation(opID) ON DELETE CASCADE
);

CREATE TABLE ATTENDS(
protocolNO INTEGER,
staffID INTEGER NOT NULL,
PRIMARY KEY(protocolNO, staffID),
FOREIGN KEY(protocolNO) REFERENCES Patient(protocolNO) ON DELETE CASCADE,
FOREIGN KEY(staffID) REFERENCES Staff(staffID) ON DELETE CASCADE
);

CREATE TABLE NEEDS(
equipID INTEGER,
opID INTEGER,
date DATE,
PRIMARY KEY(equipID, opID),
```

```
FOREIGN KEY(equipID) REFERENCES Equipment(equipID) ON DELETE CASCADE,
FOREIGN KEY(opID) REFERENCES Operation(opID) ON DELETE CASCADE
);

CREATE TABLE  MEDICALRECORD_HAS (
recordID  INTEGER,
date_of_examination DATE,
explanation VARCHAR (200),
protocolNO  INTEGER NOT NULL,
PRIMARY KEY  (recordID, protocolNO),
FOREIGN KEY  (protocolNO) REFERENCES Patient (protocolNO) ON DELETE CASCADE
);
```

• relations, and the set of functional dependencies corresponding to each relation

- PATIENT (protocolNo, patient_name, diagnosis, age, gender)
 - F = { protocolNo -> patient_name diagnosis age gender }
- OPERATION (opID, op_name, op_point)
 - F = { opID -> op_name, op_name -> op_point} .
- EQUIPMENT (equipID, equip_name)
 - F = { equipID -> equip_name }
- MEDICALRECORD (recordID, date_of_examination, explanation)
- F ={ recordID -> date_of_examination explanation }
- STAFF (staffID, staff_name)
- F = { staffID -> staff_name }
- DOCTOR (staffID)
- NURSE (staffID)
- TECHNICIAN (staffID)
- HAS (record_id, protocolNo)
 - F = { protocolNo -> record_id}

- ATTENDS (staffID, protocolNO)

- F = { staffID -> protocolNO }

- UNDERGOES (protocolNO, opID, op_room, op_date)

- F = { protocolNO opID -> op_room op_date}

- NEEDS (opID, equipID , date)

- F = { opID equipID -> date }

Our schema is BCNF except one relation which is OPERATION relation. The reason is that our ER Model is well defined and the nature of designed operation theatre itself lead to this situation.

OPERATION relation is not in the form of BCNF because it has an FD between op_name and op_point. Since op_name is not key for our relation it must be decomposed for the sake of normalization.

• Steps of the lossless-join dependency preserving decomposition into BCNF (or 3NF if dependency preserving decomposition into BCNF is not possible)

The decomposition is as follows:

FROM:

- OPERATION (opID, op_name, op_point)

- F = { opID -> op_name, op_name -> op_point} .

TO:

- OPERATION (opID, op_name) - OPERATIONLOOKUP (op_name, op_point)