Group 20

Ata Seyhan - S024812

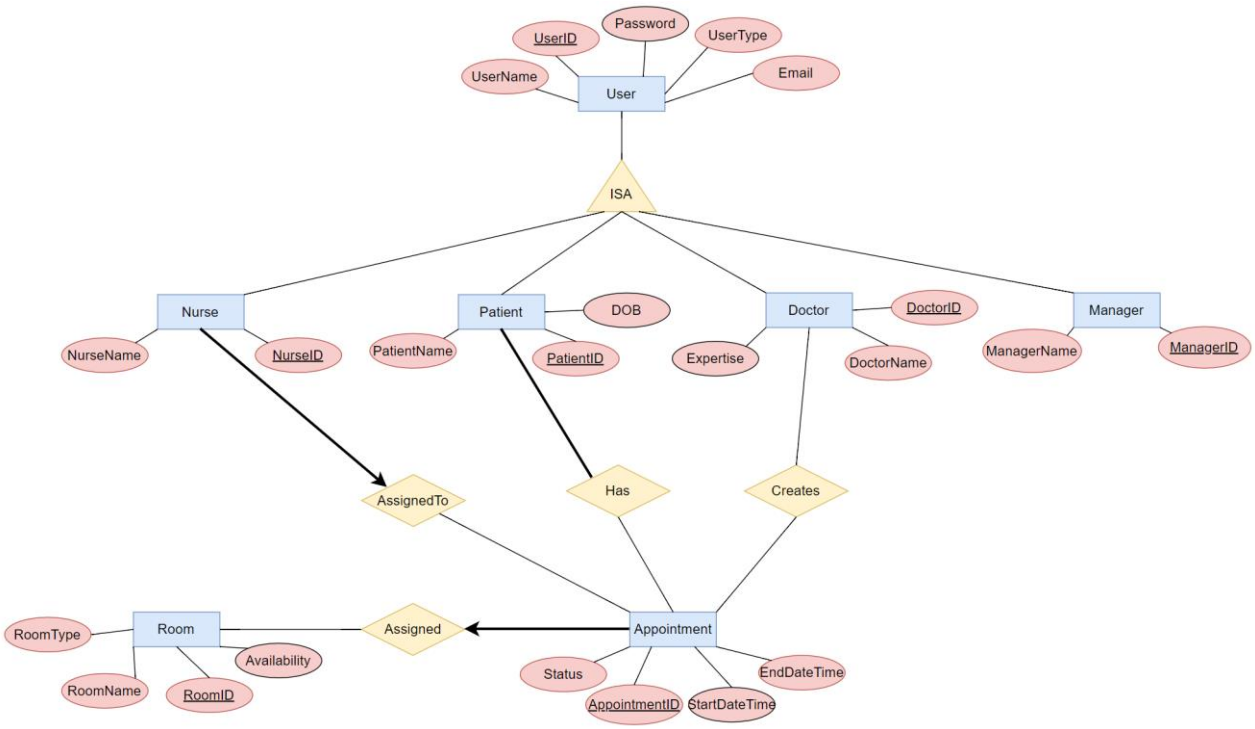
Mertcan Sağlam – S029208

CS 202 Semester Project Part 1 Report

Abstract

This study aims to explore and analyze the key aspects of SQL and Database management within the context of ER diagram, DDL code, DML code, normalization, and relational dependencies subjects. In this project we have implemented the first part of a hospital management system database model. In our model we have firstly implemented our ER diagram and then defined our DDL and DML codes according to our ER model. We have used our knowledge of normalization when defining our relational dependencies in 3NF form.

ER Diagram



When creating our ER diagram, we have defined our entities attributes and relations clearly. We have defined patient, nurse, doctor, and manager entities which have an ISA relationship with user. Patients, managers, doctors, and nurses are users in our hospital management system. Each user has a username, unique user id, password, user type and email address. Nurses, managers, doctors and patients have names and unique id's in our system. Each patient has an additional date of birth DOB information. Each doctor has their expertise information recorded in the system. Nurses are assigned to appointments, patients have appointments and doctors can create appointments. A specific appointment has its status, unique appointment id, and starting & finishing date and time records of the appointment. Appointments are assigned to rooms. Rooms have unique room id, type, name and availability attributes.

DML Code

```sql
CREATE DATABASE IF NOT EXISTS cs202_project;
USE cs202_project;

SET SQL_SAFE_UPDATES=0;
SET foreign_key_checks = 0;

DELETE FROM UserTypeLookup;
DELETE FROM User;
DELETE FROM Patient;
DELETE FROM Manager;
DELETE FROM Nurse;
DELETE FROM Doctor;
DELETE FROM Appointment;
DELETE FROM Room;

INSERT INTO UserTypeLookup (UserTypeCode, UserType) VALUES
    (1, 'Manager'),
    (2, 'Doctor'),
    (3, 'Nurse'),
    (4, 'Patient');

INSERT INTO User (email, password_hash, password_salt, UserName, userTypeCode) VALUES
    ('admin',SHA2(CONCAT('123456', RAND()), 256), -- Hashing with salt
    RAND(), 'AdminMertcan',1),
    ('manager@example.com',  SHA2(CONCAT('user_password', RAND()), 256), -- Hashing with salt
    RAND(), 'NejatIsler', 1),
    ('doctor@example.com',  SHA2(CONCAT('user_password', RAND()), 256), -- Hashing with salt
    RAND(), 'MehmetOz', 2),
    ('nurse@example.com',  SHA2(CONCAT('u3ser_password', RAND()), 256), -- Hashing with salt
    RAND(), 'MargotRobbie', 3),
    ('patient@example.com',  SHA2(CONCAT('use22r_password', RAND()), 256), -- Hashing with salt
    RAND(), 'RamboOkan', 4),
    ('manager2@example.com',  SHA2(CONCAT('use51r_password', RAND()), 256), -- Hashing with salt
    RAND(),  'RyanGosling', 1),
    ('doctor2@example.com',  SHA2(CONCAT('u3ser_password', RAND()), 256), -- Hashing with salt
    RAND(), 'DoctorDisrespect', 2),
    ('nurse2@example.com',  SHA2(CONCAT('user_password11', RAND()), 256), -- Hashing with salt
    RAND(), 'DuaLipa', 3),
    ('patient2@example.com',  SHA2(CONCAT('user33_password', RAND()), 256), -- Hashing with salt
    RAND(), 'Mertcan', 4),
    ('manager3@example.com', SHA2(CONCAT('user_passwo12rd', RAND()), 256), -- Hashing with salt
    RAND(),  'SteveJobs', 1),
    ('doctor3@example.com',  SHA2(CONCAT('user_password', RAND()), 256), -- Hashing with salt
    RAND(), 'SaglamKafa', 2);

INSERT INTO Patient (PatientName,DOB) VALUES
    ('RamboOkan','1993-08-18'),
    ('Mertcan','1980-04-05');

INSERT INTO Manager (ManagerName) VALUES ('NejatIsler'), ('RyanGosling'), ('SteveJobs');
INSERT INTO Nurse (NurseName) VALUES ('MargotRobbie'), ('DuaLipa');
INSERT INTO Doctor (DoctorName, expertise) VALUES ('MehmetOz','Gynecology'), ('DoctorDisrespect','Dermatology'), ('SaglamKafa','Oncology');

INSERT INTO Room (RoomType, RoomName, Availability) VALUES
    ('Standard', 'Room6', 1),
    ('VIP', 'Room7', 1),
    ('Standard', 'Room8', 1),
    ('VIP', 'Room9', 1),
    ('Standard', 'Room10', 1);

INSERT INTO Appointment (StartTime, EndTime, Status_, PatientID, NurseID, DoctorID, RoomID) VALUES
    ('2023-02-05 14:00:00','2023-02-07 14:00:00', 'Scheduled', 3, 1, 3, 3),
    ('2023-02-20 10:45:00','2023-02-28 14:00:00', 'Completed', 4, 2, 1, 4),
    ('2023-03-10 11:30:00','2023-05-05 14:00:00', 'Scheduled', 5, 2, 2, 2),
    ('2023-03-25 13:15:00','2023-04-05 14:00:00', 'Scheduled', 1, 3, 1, 1),
    ('2023-04-08 16:45:00','2023-04-23 14:00:00', 'Scheduled', 2, 1, 2, 5),
    ('2023-04-23 09:30:00','2023-05-05 14:00:00','Completed', 3, 2, 3, 3),
    ('2023-05-15 12:00:00','2023-06-05 14:00:00', 'Scheduled', 4, 3, 1, 2),
    ('2023-05-30 15:15:00','2023-07-05 14:00:00', 'Scheduled', 5, 1, 2, 5),
    ('2023-06-10 08:30:00','2023-06-12 14:00:00', 'Completed', 1, 2, 3, 1),
    ('2023-06-25 11:00:00','2023-06-26 14:00:00', 'Scheduled', 2, 3, 1, 4);

SET foreign_key_checks = 1;
SET SQL_SAFE_UPDATES=1;
```

We have updated our own database for each table. We have used insert statements to try our dummy datas. We have also used Mockaroo and Ruby Script to generate realistic dummy start times, end times, status, patient id, nurse id, doctor id, room id, room type, room name, availability etc. We did not use any null values for our dataset for simplicity purposes. We have also SET SQL_SAFE_UPDATE flag to 0 and then deleted all the rows for safe insertion at the end of the code we SET SQL_SAFE_UPDATE flag back to 1. Our code is stated above.

DDL Code

```sql
CREATE DATABASE IF NOT EXISTS cs202_project;
USE cs202_project;

SET foreign_key_checks = 0;

DROP TABLE IF EXISTS RoomAssignment;
DROP TABLE IF EXISTS DoctorSchedule;
DROP TABLE IF EXISTS Nurse;
DROP TABLE IF EXISTS Manager;
DROP TABLE IF EXISTS Appointment;
DROP TABLE IF EXISTS Patient;
DROP TABLE IF EXISTS Room;
DROP TABLE IF EXISTS Doctor;
DROP TABLE IF EXISTS User;
DROP TABLE IF EXISTS UserTypeLookup;  -- Added this line

CREATE TABLE UserTypeLookup (  -- maps the usertype
    UserTypeCode INT NOT NULL PRIMARY KEY,
    UserType VARCHAR(50) NOT NULL
);
CREATE TABLE User (
    UserID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(255),
    password_hash VARCHAR(255) NOT NULL,
    password_salt VARCHAR(255) NOT NULL,
    UserName VARCHAR(255),
    userTypeCode INT,
    FOREIGN KEY (userTypeCode) REFERENCES UserTypeLookup(UserTypeCode)  -- Added this line
);
CREATE TABLE Patient (
    PatientID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    DOB VARCHAR(255) NOT NULL,
    PatientName VARCHAR(255),

    FOREIGN KEY (PatientID) REFERENCES User(UserID)
);
CREATE TABLE Manager (
    ManagerID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ManagerName VARCHAR(255),

    FOREIGN KEY (ManagerID) REFERENCES User(UserID)
);
CREATE TABLE Nurse (
    NurseID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    NurseName VARCHAR(255),

    FOREIGN KEY (NurseID) REFERENCES User(UserID)
);
CREATE TABLE Doctor (
    DoctorID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    DoctorName VARCHAR(255),
    expertise VARCHAR(100) NOT NULL,

    FOREIGN KEY (DoctorID) REFERENCES User(UserID)
);
CREATE TABLE Appointment (
    AppointmentID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    StartTime DATETIME NOT NULL,
    EndTime DATETIME,
    Status_ VARCHAR(50),
    PatientID INT,
    NurseID INT,
    DoctorID INT,
    RoomID INT,
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID)  ON DELETE CASCADE,
    FOREIGN KEY (NurseID) REFERENCES Nurse(NurseID),
    FOREIGN KEY (RoomID) REFERENCES Room(RoomID)  ON DELETE RESTRICT
);
CREATE TABLE Room (
    RoomID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    RoomType VARCHAR(50),
    RoomName VARCHAR(255),
    Availability BIT NOT NULL,  -- BOOLEAN
    FOREIGN KEY (RoomID) REFERENCES Appointment(AppointmentID)
);
SET foreign_key_checks = 1;
```

We have created our DDL code by implementing the necessary tables which reflects our ER diagram. We have used drop table statements. We have clearly defined the primary key, foreign key and indicated the references in each table. We have implemented a UserTypeLookup which enables us to map the user types to our entities with integers. We have implemented delete cascade and delete restrict to handle the foreign keys.

Relational Dependencies

```
UserTypeLookup:

UserTypeCode → UserType

User:

UserID → email
UserID → password
UserID → UserName
UserID → userTypeCode

Patient:

PatientID → DOB
PatientID → PatientName

Manager:

ManagerID → ManagerName

Nurse:

NurseID → NurseName

Doctor:

DoctorID → DoctorName
DoctorID → expertise

Appointment:

AppointmentID → datetime
AppointmentID → Status
AppointmentID → PatientID
AppointmentID → NurseID
AppointmentID → DoctorID
AppointmentID → RoomID

Room:

RoomID → RoomType
RoomID → RoomName
RoomID → Availability
```

We have clearly defined our functional dependencies. We have put them in 3NF form. By this way he have understood how to maintain our database system. much more efficiently.