St.Mungo's Hospital for Magical Maladies and Injuries

Phase 2

Aashish Mahajan
amahaja@asu.edu
1207662725

Ajay Kumar
achirum1@asu.edu
1205862797

Kruthika Tanikella
Kruthika.Tanikella@asu.edu
1207764060

Momin Zahed
zqmomin@asu.edu
1207656043

Srivikas Vishnubhotla
Srivikas.Vishnubhotla@asu.edu
1206616849

Contents

Part I - ER to Relational

```
CREATE DATABASE IF NOT EXISTS hospital;
```

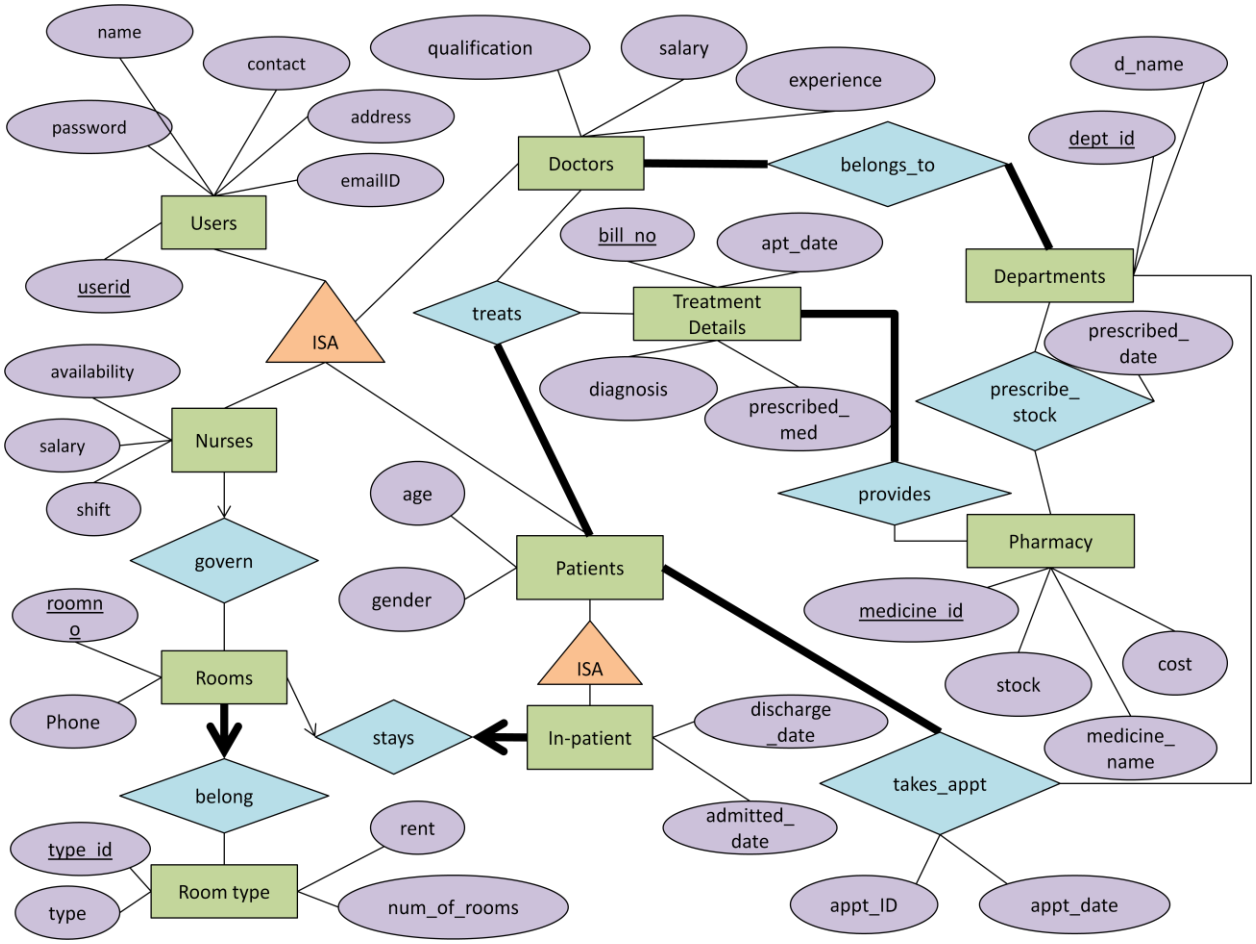


Figure 1 - ER Diagram

The following entities and relationships are converted into tables in the database (The names in the brackets correspond to the table names in the database). Not all relationships are converted into tables, the features are incorporated into the existing tables:

Users (users)	Nurses (nurses)	Rooms (rooms)
Room Type (roomtype)	Patients (patients)	In - patient (in_patients)
Doctors (doctors)	Pharmacy (pharmacy)	Treatment Details (treatments)
Departments (departments)	treats (treats)	belongs_to (belongs_to)
takes_appt (appointments)		

```
DROP TABLE IF EXISTS appointments;
DROP TABLE IF EXISTS treats;
DROP TABLE IF EXISTS treatments;
DROP TABLE IF EXISTS pharmacy;
DROP TABLE IF EXISTS belongs_to;
DROP TABLE IF EXISTS departments;
DROP TABLE IF EXISTS doctors;
DROP TABLE IF EXISTS in_patients;
DROP TABLE IF EXISTS patients;
DROP TABLE IF EXISTS nurses;
DROP TABLE IF EXISTS rooms;
DROP TABLE IF EXISTS roomtype;
DROP TABLE IF EXISTS users;

CREATE TABLE users (
userid     VARCHAR(10) NOT NULL,
password   VARCHAR(50) NOT NULL,
name       VARCHAR(25),
contact    INT,
addr       VARCHAR(50),
emailid    VARCHAR(40),
type       INT NOT NULL,
PRIMARY KEY(userid))
ENGINE=InnoDB;
```

`users.type` can take the following integer values, each integer value representing a different type of user:

0 : Admin
1 : Nurse
2 : Patient
3 : Doctor

Table 1 - users

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**userid**	varchar(10)	latin1_swedish_ci		No	*None*	
2	**password**	varchar(50)	latin1_swedish_ci		No	*None*	
3	**name**	varchar(25)	latin1_swedish_ci		Yes	*NULL*	
4	**contact**	int(11)			Yes	*NULL*	
5	**addr**	varchar(50)	latin1_swedish_ci		Yes	*NULL*	
6	**emailid**	varchar(40)	latin1_swedish_ci		Yes	*NULL*	
7	**type**	int(11)			No	*None*	

```
CREATE TABLE roomtype (
typeid           INT NOT NULL AUTO_INCREMENT,
rtype            VARCHAR(15),
rent             FLOAT,
number_of_rooms  INT,
PRIMARY KEY(typeid))
ENGINE=InnoDB;
```

Table 2 - roomtype

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**typeid**	int(11)			No	*None*	AUTO_INCREMENT
2	**rtype**	varchar(15)	latin1_swedish_ci		Yes	*NULL*	
3	**rent**	float			Yes	*NULL*	
4	**number_of_rooms**	int(11)			Yes	*NULL*	

```
CREATE TABLE rooms (
roomno      INT NOT NULL AUTO_INCREMENT,
phone       INT,
typeid      INT NOT NULL,
PRIMARY KEY(roomno),
FOREIGN KEY(typeid) REFERENCES roomtype (typeid)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE=InnoDB;
```

Table 3 - rooms

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**roomno**	int(11)			No	*None*	AUTO_INCREMENT
2	**phone**	int(11)			Yes	*NULL*	
3	**typeid**	int(11)			No	*None*	

```
CREATE TABLE nurses (
userid     VARCHAR(10) NOT NULL,
workshift  BIT NOT NULL,
salary     FLOAT,
available  BIT,
roomno     INT,
PRIMARY KEY(userid),
FOREIGN KEY(userid) REFERENCES users (userid)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY(roomno) REFERENCES rooms (roomno)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE=InnoDB;
```

`nurses.workshift` is a `bit` value which can take either 0 or 1 indicating in which shift a particular nurse works (day/night):
1 : day shift
0 : night shift

`nurses.available` is a `bit` value which can take either 0 or 1 indicating if a nurse is currently available to be assigned to a room:
1 : available
0 : unavailable

Table 4 - nurses

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**userid**	varchar(10)	latin1_swedish_ci		No	*None*	
2	**workshift**	bit(1)			No	*None*	
3	**salary**	float			Yes	*NULL*	
4	**available**	bit(1)			Yes	*NULL*	
5	**roomno**	int(11)			Yes	*NULL*	

```
CREATE TABLE patients (
userid     VARCHAR(10) NOT NULL UNIQUE,
age        TINYINT,
gender     BIT,
PRIMARY KEY(userid),
FOREIGN KEY(userid) REFERENCES users (userid)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE=InnoDB;
```

`patients.gender` is a `bit` value which can take either 0 or 1 indicating the gender of the patient:
1 : female
0 : male

Table 5 - patients

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**userid**	varchar(10)	latin1_swedish_ci		No	*None*	
2	**age**	tinyint(4)			Yes	*NULL*	
3	**gender**	bit(1)			Yes	*NULL*	

```
CREATE TABLE in_patients (
userid           VARCHAR(10) NOT NULL UNIQUE,
admit_date       DATE,
discharge_date   DATE,
roomno           INT NOT NULL UNIQUE,
PRIMARY KEY(userid, admit_date),
FOREIGN KEY(userid) REFERENCES patients (userid)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY(roomno) REFERENCES rooms (roomno)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE=InnoDB;
```

Table 6 - in_patients

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**userid**	varchar(10)	latin1_swedish_ci		No	*None*	
2	**admit_date**	date			No	0000-00-00	
3	**discharge_date**	date			Yes	*NULL*	
4	**roomno**	int(11)			No	*None*	

```
CREATE TABLE doctors (
userid          VARCHAR(10) NOT NULL UNIQUE,
qualification   VARCHAR(20),
salary          FLOAT,
experience      DECIMAL,
PRIMARY KEY (userid),
FOREIGN KEY(userid) REFERENCES users (userid)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE=InnoDB;
```

Table 7 - doctors

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**userid**	varchar(10)	latin1_swedish_ci		No	*None*	
2	**qualification**	varchar(20)	latin1_swedish_ci		Yes	*NULL*	
3	**salary**	float			Yes	*NULL*	
4	**experience**	decimal(10,0)			Yes	*NULL*	

```
CREATE TABLE departments (
dept_id    INT NOT NULL,
dept_name  VARCHAR(25),
PRIMARY KEY (dept_id))
ENGINE=InnoDB;
```

Table 8 - departments

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**dept_id**	int(11)			No	*None*	
2	**dept_name**	varchar(25)	latin1_swedish_ci		Yes	*NULL*	

```
CREATE TABLE belongs_to (
userid     VARCHAR(10) NOT NULL,
dept_id    INT NOT NULL,
PRIMARY KEY (userid, dept_id),
FOREIGN KEY (userid) REFERENCES doctors (userid)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY (dept_id) REFERENCES departments (dept_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE=InnoDB;
```

Table 9 - belongs_to

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**userid**	varchar(10)	latin1_swedish_ci		No	*None*	
2	**dept_id**	int(11)			No	*None*	

```
CREATE TABLE pharmacy (
medicine_id      INT NOT NULL AUTO_INCREMENT,
medicine_name    VARCHAR(50) NOT NULL,
cost             FLOAT NOT NULL,
stock            INT NOT NULL,
dept_id          INT NOT NULL, prescribed_date DATE,
PRIMARY KEY (medicine_id),
FOREIGN KEY (dept_id) REFERENCES departments (dept_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE=InnoDB;
```

Table 10 - pharmacy

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**medicine_id**	int(11)			No	*None*	AUTO_INCREMENT
2	**medicine_name**	varchar(50)	latin1_swedish_ci		No	*None*	
3	**cost**	float			No	*None*	
4	**stock**	int(11)			No	*None*	
5	**dept_id**	int(11)			No	*None*	
6	**prescribed_date**	date			Yes	*NULL*	

```
CREATE TABLE treatments (
bill_no          INT NOT NULL AUTO_INCREMENT,
apt_date         DATE,
diagnosis        VARCHAR(100),
prescribed_med   VARCHAR(25),
PRIMARY KEY (bill_no))
ENGINE=InnoDB;
```

Table 11 - treatments

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**bill_no**	int(11)			No	*None*	AUTO_INCREMENT
2	**apt_date**	date			Yes	*NULL*	
3	**diagnosis**	varchar(100)	latin1_swedish_ci		Yes	*NULL*	
4	**prescribed_med**	varchar(25)	latin1_swedish_ci		Yes	*NULL*	

```
CREATE TABLE treats (
patient_id       VARCHAR(10) NOT NULL,
doctor_id        VARCHAR(10) NOT NULL,
bill_no          INT NOT NULL,
PRIMARY KEY (patient_id, bill_no),
FOREIGN KEY (patient_id) REFERENCES patients (userid)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY (doctor_id) REFERENCES doctors (userid)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY (bill_no) REFERENCES treatments (bill_no)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE=InnoDB;
```

Table 12 - treats

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**patient_id**	varchar(10)	latin1_swedish_ci		No	*None*	
2	**doctor_id**	varchar(10)	latin1_swedish_ci		No	*None*	
3	**bill_no**	int(11)			No	*None*	

```
CREATE TABLE appointments (
apt_id     INT NOT NULL AUTO_INCREMENT,
apt_date   DATE,
patient_id VARCHAR(10) NOT NULL,
dept_id    INT NOT NULL,
PRIMARY KEY (apt_id),
FOREIGN KEY (patient_id) REFERENCES patients (userid)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
FOREIGN KEY (dept_id) REFERENCES departments (dept_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE=InnoDB;
```

Table 13 - appointments

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**apt_id**	int(11)			No	*None*	AUTO_INCREMENT
2	**apt_date**	date			Yes	*NULL*	
3	**patient_id**	varchar(10)	latin1_swedish_ci		No	*None*	
4	**dept_id**	int(11)			No	*None*	

```
DROP TRIGGER IF EXISTS `insertPND`;
DROP TRIGGER IF EXISTS `updatePND`;
DROP TRIGGER IF EXISTS `check_dates_patients_insert`;
DROP TRIGGER IF EXISTS `check_dates_patients_update`;
delimiter //

CREATE TRIGGER insertPND
AFTER INSERT ON users
FOR EACH ROW
BEGIN
      IF (new.type = 2) THEN
      INSERT INTO patients( userid ) VALUES (new.userid);

      ELSEIF (new.type = 3) THEN
      INSERT INTO doctors( userid ) VALUES (new.userid);

      ELSEIF (new.type = 1) THEN
      INSERT INTO nurses( userid, workshift ) VALUES (new.userid, 1);

      ELSEIF (NEW.type > 4) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "Value of Type field
violated";

END IF ;
END//

CREATE TRIGGER updatePND
BEFORE UPDATE ON users
FOR EACH ROW
BEGIN
      IF (new.type != (SELECT u.type FROM users u WHERE u.userid =
new.userid)) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "Value of Type field
cannot be updated";
END IF ;
END//

CREATE TRIGGER check_dates_patients_insert
BEFORE INSERT ON in_patients
FOR EACH ROW
BEGIN
      IF (NEW.admit_date > NEW.discharge_date) THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "Admit date
larger than discharge date";
      END IF;
END//

CREATE TRIGGER check_dates_patients_update
BEFORE UPDATE ON in_patients
FOR EACH ROW
BEGIN
```

```
      IF (NEW.admit_date > NEW.discharge_date) THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "Admit date larger
than discharge date";
      END IF;
END//

delimiter ;
```

The triggers written above enforce some additional constraints required in our database. We have four triggers which perform each of the following actions:

1) `insertPND` trigger ensures that whenever a new record is inserted in the `users` table, a new record with that userid is inserted into `patients`, `nurses` or `doctors` tables correspondingly.
2) `updatePND` trigger ensures that once a userid is inserted into users table with a specific `type`, the value of `type` cannot be changed for that userid again. Once a doctor, always a doctor.
3) `check_dates_patients_insert` and `check_dates_patients_update` triggers ensure that whenever a new record is inserted into `in_patients` table, the discharge date value is always greater than the admit date value.

Part II - Sample Data

```
INSERT INTO users VALUES
('AjayAdmin', 'Ajay', 'Ajay', 4569564569, 'Mills dr Tempe', 'ajay@gmail.com', 0),
('ZahedAdmin', 'Zahed', 'Zahed', 7974569894, 'Mills Tempe', 'zahed@gmail.com', 0),
('VikasAdmin', 'Vikas', 'Vikas', 4469567458, 'Roten grn Tempe', 'vikas@gmail.com', 0),
('KruthikaAdmin', ' Kruthika', 'Kruthika', 4467896458, 'Apache Tempe', 'Kruth@gmail.com', 0),
('AashishAdmin', 'Aashish', 'Aashish', 4567456894, 'Mill Tempe', 'aashi@gmail.com', 0);

INSERT INTO users VALUES
('Kelly1', 'Kelly', 'Kelly', 4567456894, 'Mill Tempe', 'Kelly@gmail.com', 1),
('Tina', 'Tina', 'Tina', 4567456894, 'Mill Tempe', 'tina@gmail.com', 1),
('Sunny1', 'Sunny', 'Sunny', 4567456894, 'Amber Tempe', 'sunny@gmail.com', 1),
('Marty1', 'Marty', 'Marty', 4567456894, 'University Tempe', 'marty@gmail.com', 1),
('Sara1', 'Sara', 'Sara', 4227456894, 'Mill Tempe', 'sara@gmail.com', 1),
('Priya1', 'Priya', 'Priya', 433456894, 'Amber Tempe', 'priya@gmail.com', 1),
('Claire1', 'Claire', 'Claire', 2267456894, 'Mill Tempe', 'claire@gmail.com', 1),
('Torry1', 'Torry', 'torry', 4567756894, 'U dr Tempe', 'torry@gmail.com', 1),
('Nelly1', 'Nelly', ' Nelly', 4567456894, 'Mill Tempe', 'Nelly@gmail.com', 1);

INSERT INTO users VALUES
('Aashish1', 'Aashish', 'Aashish', 4569569894, 'Mills dr Tempe', 'aashi@gmail.com', 2),
('Mohan1', 'Mohan', 'Mohan', 4569567854, 'Mills dr tempe', 'mohan@gmail.com', 2),
('Abhinav1', 'Abhinav', 'Abhinav', 4569578965, 'Mills dr tempe', 'abhi@gmail.com', 2),
('Phani1', 'Phani', 'Phani', 4569574124, 'Mills dr tempe', 'phani@gmail.com', 2),
('Akshay1', 'Akshay', 'Akshay', 45695612365, 'Mills dr tempe', 'aksh@gmail.com', 2),
('Vikas1', 'Vikas', 'Vikas', 4469567458, 'Roten grn tempe', 'vikas@gmail.com', 2),
('Kruthika1', 'Kruthika', 'Kruthika', 4569564569, 'Mills dr tempe', 'kruthika@gmail.com', 2),
('Pooja1', 'Pooja', 'Pooja', 4569564449, 'Dre tempe', 'pooja@gmail.com', 2),
('Sue1', 'Sue', 'Sue', 4569567474, 'Mat dr tempe', 'sue@gmail.com', 2),
('Camille1', 'Camille', 'Camille', 4577564569, 'Mills tempe', 'camille@gmail.com', 2);

INSERT INTO users VALUES
('Smith1', 'smith', 'smith', 4807779878, '12 mclintock drive, tempe', 'smith@gmail.com', 3),
('Anderson1', 'Anderson', 'Anderson', 4807745698, '12 mclintock drive, tempe', 'Anderson@gmail.com', 3),
('Clark1', 'Clark', 'Clark', 4800000878, '12 University drive, tempe', 'clark@gmail.com', 3),
('James1', 'James', 'James', 4800001234, 'Hill Top Drive, tempe', 'james@gmail.com', 3),
('Kevin1', 'Kevin', 'Kevin', 7960001234, 'Apache drive, tempe', 'kevin@gmail.com', 3),
('William1', 'William', 'William', 7780001234, 'Mills tempe', 'william@gmail.com', 3),
('Perez1', 'Perez', 'Perez', 7960569834, 'Apache drive, tempe', 'perez@gmail.com', 3),
('Lee1', 'Lee', 'Lee', 7960078964, 'Amber garden, tempe', 'kevin@gmail.com', 3),
('Bing1', 'Bing', 'Bing', 7960001234, 'Apache drive, tempe', 'bing@gmail.com', 3),
('Rob1', 'Rob', 'Rob', 7789601234, 'Apache drive, tempe', 'rob@gmail.com', 3),
('Ron1', 'Ron', 'Ron', 7960001234, 'Amber drive, tempe', 'ron@gmail.com', 3),
('Zahed1', 'Zahed', 'Zahed', 7974569894, 'Mills dr tempe', 'zahed@gmail.com', 3);
```

Table 14 - users - Sample Data

userid	password	name	contact	addr	emailid	type
aashish.m	aashish123	Aashish	2147483647	Quadrangles, University Drive	aashish.m@mungo.com	0
kruthika.t	kruthika123	Kruthika	2147483647	Villas, Apache Boulevard	kruthika.t@mungo.com	0
ajay.k	ajay123	Ajay	2147483647	Quadrangles, University Drive	ajay.k@mungo.com	0
vikas.v	vikas123	Vikas	2147483647	Quadrangles, University Drive	vikas.v@mungo.com	0
zahed.m	zahed123	Zahed	2147483647	Quadrangles, University Drive	zahed.m@mungo.com	0
Torry1	Torry	torry	2147483647	U dr Tempe	torry@gmail.com	1
Tina	Tina	Tina	2147483647	Mill Tempe	tina@gmail.com	1
Sara1	Sara	Sara	2147483647	Mill Tempe	sara@gmail.com	1
Priya1	Priya	Priya	433456894	Amber Tempe	priya@gmail.com	1
Nelly1	Nelly	Nelly	2147483647	Mill Tempe	Nelly@gmail.com	1
Marty1	Marty	Marty	2147483647	University Tempe	marty@gmail.com	1
Sunny1	Sunny	Sunny	2147483647	Amber Tempe	sunny@gmail.com	1
Kelly1	Kelly	Kelly	2147483647	Mill Tempe	Kelly@gmail.com	1
Claire1	Claire	Claire	2147483647	Mill Tempe	claire@gmail.com	1
Sue1	Sue	Sue	2147483647	Mat dr tempe	sue@gmail.com	2
Camille1	Camille	Camille	2147483647	Mills tempe	camille@gmail.com	2
Akshay1	Akshay	Akshay	2147483647	Mills dr tempe	aksh@gmail.com	2
Pooja1	Pooja	Pooja	2147483647	Dre tempe	pooja@gmail.com	2

```
INSERT INTO roomtype VALUES
(1,'General',100,20),
(2,'Deluxe',200,10),
(3,'Luxury',300,8),
(4,'Premium',450,2);

INSERT INTO rooms VALUES
(101, 3101, 1),
(202, 3202, 2),
(103, 3103, 3),
(104, 3104, 2),
(105, 3105, 4),
(106, 3106, 1),
(107, 3107, 1),
(308, 3308, 3),
(109, 3109, 2);
```

Table 15 - roomtype - Sample Data

typeid	rtype	rent	number_of_rooms
1	General	100	20
2	Deluxe	200	10
3	Luxury	300	8
4	Premium	450	2

Table 16 - rooms - Sample Data

roomno	phone	typeid
101	3101	1
103	3103	3
104	3104	2
105	3105	4
106	3106	1
107	3107	1
109	3109	2
202	3202	2
308	3308	3

Since, the `userid` values have already been inserted into `patients`, `doctors` and `nurses` tables, we only update the tuples to add values to the other attributes in each table.

```
UPDATE nurses SET workshift = 1, salary = 210.00, available = 0, roomno = 101 WHERE userid = 'Kelly1';
UPDATE nurses SET workshift = 1, salary = 240.00, available = 0, roomno = 202 WHERE userid = 'Tina';
UPDATE nurses SET workshift = 1, salary = 160.00, available = 0, roomno = 103 WHERE userid = 'Sunny1';
UPDATE nurses SET workshift = 1, salary = 180.00, available = 0, roomno = 104 WHERE userid = 'Marty1';
UPDATE nurses SET workshift = 1, salary = 200.00, available = 0, roomno = 105 WHERE userid = 'Sara1';
UPDATE nurses SET workshift = 0, salary = 220.00, available = 1, roomno = NULL WHERE userid = 'Priya1';
UPDATE nurses SET workshift = 1, salary = 240.00, available = 0, roomno = 107 WHERE userid = 'Claire1';
UPDATE nurses SET workshift = 1, salary = 230.00, available = 0, roomno = 308 WHERE userid = 'Torry1';
UPDATE nurses SET workshift = 0, salary = 280.00, available = 1, roomno = NULL WHERE userid = 'Nelly1';

UPDATE patients SET age = 23, gender = 0 WHERE userid = "Aashish1";
UPDATE patients SET age = 28, gender = 0 WHERE userid = "Mohan1";
UPDATE patients SET age = 19, gender = 1 WHERE userid = "Camille1";
UPDATE patients SET age = 25, gender = 0 WHERE userid = "Abhinav1";
UPDATE patients SET age = 24, gender = 0 WHERE userid = "Akshay1";
UPDATE patients SET age = 25, gender = 0 WHERE userid = "Vikas1";
UPDATE patients SET age = 23, gender = 1 WHERE userid = "Kruthika1";
UPDATE patients SET age = 21, gender = 0 WHERE userid = "Phani1";
UPDATE patients SET age = 26, gender = 1 WHERE userid = "Pooja1";
UPDATE patients SET age = 29, gender = 1 WHERE userid = "Sue1";
```

Table 17 - nurses - Sample Data

userid	workshift	salary	available	roomno
Claire1	1	240	0	107
Kelly1	1	210	0	101
Marty1	1	180	0	104
Nelly1	0	280	1	*NULL*
Priya1	0	220	1	*NULL*
Sara1	1	200	0	105
Sunny1	1	160	0	103
Tina	1	240	0	202
Torry1	1	230	0	308

Table 18 - patients - Sample Data

userid	age	gender
Aashish1	23	0
Abhinav1	25	0
Akshay1	24	0
Camille1	19	1
Kruthika1	23	1
Mohan1	28	0
Phani1	21	0
Pooja1	26	1
Sue1	29	1
Vikas1	25	0

```
INSERT INTO in_patients VALUES
("Mohan1","2014-09-17","2014-10-17",103),
("Abhinav1","2014-09-19","2014-10-09",104),
("Akshay1","2014-10-09","2014-10-24",107),
("Sue1","2014-10-12","2014-10-14",109);
```

Table 19 - in_patients - Sample Data

userid	admit_date	discharge_date	roomno
Abhinav1	2014-10-17	2014-10-19	104
Akshay1	2014-10-07	2014-10-17	107
Camille1	2014-10-17	2014-11-10	308
Mohan1	2014-09-17	2014-10-17	103
Sue1	2014-10-17	2014-11-17	109

```
UPDATE doctors SET qualification = "Surgeon", salary = 900, experience = 10.5 WHERE userid = "Smith1";
UPDATE doctors SET qualification = "Neurologist", salary = 1000, experience = 12.5 WHERE userid = "Anderson1";
UPDATE doctors SET qualification = "Paediatrician", salary = 1230, experience = 5.5 WHERE userid = "Clark1";
UPDATE doctors SET qualification = "Psychology", salary = 1200, experience = 8 WHERE userid = "James1";
UPDATE doctors SET qualification = "Dentist", salary = 1100, experience = 2 WHERE userid = "Kevin1";
UPDATE doctors SET qualification = "Ophthalmologist", salary = 1520, experience = 6.2 WHERE userid = "William1";
UPDATE doctors SET qualification = "Dermatologist", salary = 2100, experience = 7.3 WHERE userid = "Perez1";
UPDATE doctors SET qualification = "Anaesthesiologist", salary = 1800, experience = 16 WHERE userid = "Lee1";
UPDATE doctors SET qualification = "Physiotherapy", salary = 1100, experience = 9 WHERE userid = "Bing1";
UPDATE doctors SET qualification = "Gynaecologist", salary = 3200, experience = 12 WHERE userid = "Rob1";
UPDATE doctors SET qualification = "Ophthalmologist", salary = 3100, experience = 2 WHERE userid = "Ron1";
UPDATE doctors SET qualification = "Dentist", salary = 2200, experience = 12 WHERE userid = "Zahed1";
```

Table 20 - doctors - Sample Data

userid	qualification	salary	experience
Anderson1	Neurologist	1000	13
Bing1	Physiotherapy	1100	9
Clark1	Paediatrician	1230	6
James1	Psychology	1200	8
Kevin1	Dentist	1100	2
Lee1	Anaesthesiologist	1800	16
Perez1	Dermatologist	2100	7
Rob1	Gynaecologist	3200	12
Ron1	Ophthalmologist	3100	2
Smith1	Surgeon	900	11
William1	Ophthalmologist	1520	6
Zahed1	Dentist	2200	12

```
INSERT INTO departments VALUES
(1, "General"),
(2, "Neurology"),
(3, "Children"),
(4, "Psychology"),
(5, "Dental"),
(6, "ENT"),
(7, "Skin"),
(8, "Drugs"),
(9, "Physiotherapy"),
(10, "Gynaecology");
```

Table 21 - departments - Sample Data

dept_id	dept_name
1	General
2	Neurology
3	Children
4	Psychology
5	Dental
6	ENT
7	Skin
8	Drugs
9	Physiotherapy
10	Gynaecology

```
INSERT INTO belongs_to VALUES
("Smith1", 1),
("Anderson1", 2),
("Clark1", 3),
("James1", 4),
("Kevin1", 5),
("William1", 6),
("Perez1", 7),
("Lee1", 8),
("Bing1", 9),
("Rob1", 10);
```

```
INSERT INTO pharmacy VALUES
(1,"m1",6,800,1,"2014-10-17"),
(2,"m2",4,500,3,"2014-01-17"),
(3,"m3",5,80,5,"2014-10-11"),
(4,"m4",9,65,2,"2014-10-19"),
(5,"m5",25,1200,2,"2014-10-16"),
(6,"m6",45,15,3,"2014-10-17"),
(7,"m7",2,346,5,"2014-10-27"),
(8,"m8",34,986,6,"2014-10-11"),
(9,"m9",18,127,9,"2014-10-13"),
(10,"m10",1,387,9,"2014-10-14");
```

Table 22 - belongs_to - Sample Data

userid	dept_id
Smith1	1
Anderson1	2
Clark1	3
James1	4
Kevin1	5
William1	6
Perez1	7
Lee1	8
Bing1	9
Rob1	10

Table 23 - pharmacy - Sample Data

medicine_id	medicine_name	cost	stock	dept_id	prescribed_date
1	m1	6	800	1	2014-10-17
2	m2	4	500	3	2014-01-17
3	m3	5	80	5	2014-10-11
4	m4	9	65	2	2014-10-19
5	m5	25	1200	2	2014-10-16
6	m6	45	15	3	2014-10-17
7	m7	2	346	5	2014-10-27
8	m8	34	986	6	2014-10-11
9	m9	18	127	9	2014-10-13
10	m10	1	387	9	2014-10-14

```
INSERT INTO treatments VALUES
(1234,"2014-01-17","General","m1,m2"),
(1235,"2014-11-17","Neurology","m3,m2"),
(1245,"2014-09-17","Psychology","m1,m2,m3"),
(8457,"2014-07-14","Pshysiotherapy","m5,m6,m4"),
(2895,"2014-08-17","Children","m7,m8,m9"),
(9281,"2014-05-17","Gynacology","m10,m5,m3"),
(5576,"2014-07-17","Dental","m1,m6,m3"),
(4993,"2014-10-27","Eye","m7,m2,m8"),
(5798,"2014-10-13","Skin","m9,m1,m5"),
(9859,"2014-10-07","Drugs","m3,m8,m2");

INSERT INTO treats VALUES
("Aashish1","Smith1",1234),
("Mohan1","Anderson1",1235),
("Abhinav1","Clark1",1245),
("Phani1","James1",8457),
("Akshay1","Kevin1",2895),
("Vikas1","William1",9281),
("Sue1","Perez1",5576);

INSERT INTO appointments VALUES
(321,"2014-10-18","Aashish1",3),
(322,"2014-10-13","Mohan1",6),
(323,"2014-10-15","Akshay1",1),
(324,"2014-09-16","Kruthika1",5),
(325,"2014-10-20","Vikas1",1);
```

Table 24 - treatments - Sample Data

bill_no	apt_date	diagnosis	prescribed_med
1234	2014-01-17	General	m1,m2
1235	2014-11-17	Neurology	m3,m2
1245	2014-09-17	Psychology	m1,m2,m3
2895	2014-08-17	Children	m7,m8,m9
4993	2014-10-27	Eye	m7,m2,m8
5576	2014-07-17	Dental	m1,m6,m3
5798	2014-10-13	Skin	m9,m1,m5
8457	2014-07-14	Pshysiotherapy	m5,m6,m4
9281	2014-05-17	Gynacology	m10,m5,m3
9859	2014-10-07	Drugs	m3,m8,m2

Table 25 - treats - Sample Data

patient_id	doctor_id	bill_no
Mohan1	Anderson1	1235
Abhinav1	Clark1	1245
Phani1	James1	8457
Akshay1	Kevin1	2895
Sue1	Perez1	5576
Aashish1	Smith1	1234
Vikas1	William1	9281

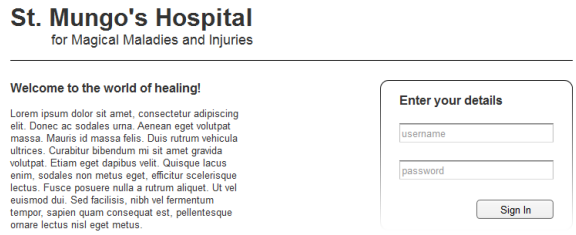
Table 26 - appointments - Sample Data

apt_id	apt_date	patient_id	dept_id
321	2014-10-18	Aashish1	3
322	2014-10-13	Mohan1	6
323	2014-10-15	Akshay1	1
324	2014-09-16	Kruthika1	5
325	2014-10-20	Vikas1	1

Part III - SQL Queries

Generic Pages

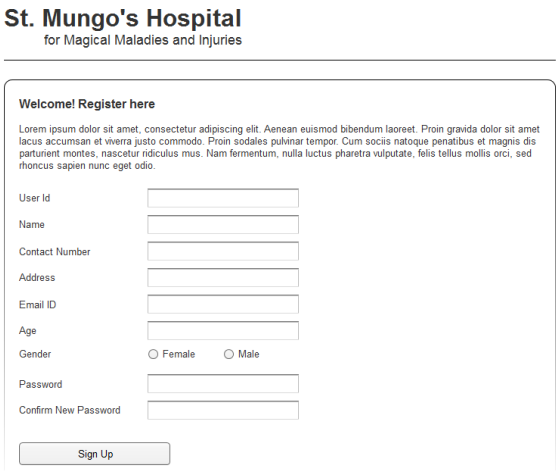
Login Page



Query 1.1:

```
SELECT      COUNT(*)
FROM        users
WHERE       userid = "Aashish1"
AND         password = "Aashish";
```

Registration Page (only for Patients)



Query 1.2:

```
INSERT INTO users
VALUES ('sam123', 'sam123','Sam
N', 123456789, 'Mesa',
'sam.n@abc.com',2);
```

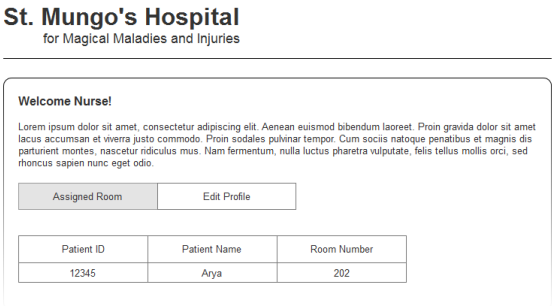
The above insert statement will execute the trigger 'insertPND' and insert the userid of the above user in to the Patients table. The other specific attributes of patients will be updated using the query:

Query 1.3:

```
UPDATE      patients
SET         age = 21,
            gender = 1
WHERE       userid = 'sam123';
```

Nurses' Pages

Nurse Page 1 - Assigned Room

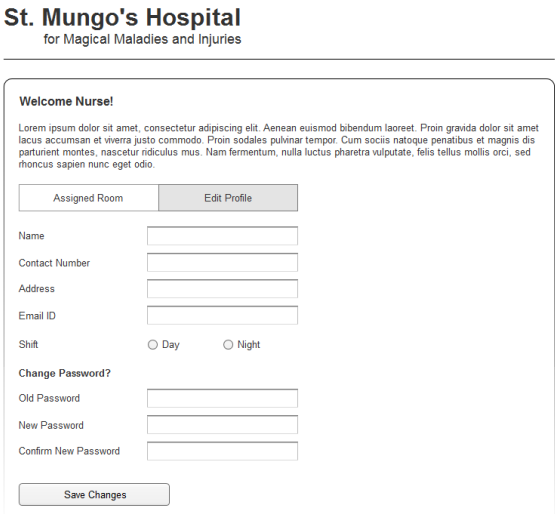


After logging in as a nurse, the nurse can view which room she is assigned to and the details of the patient admitted to that room.

Query 2.1:

```
SELECT      U.userid, U.name, I.roomno
FROM        users U, in_patients I
WHERE       U.userid = I.userid
AND         I.roomno IN
            (SELECT     N.roomno
            FROM        nurses N
            WHERE       N.userid="Claire1");
```

Nurse Page 2 - Edit Profile



Nurse can edit her/his details. These details will be updated in the database.

Assuming that all the above fields are populated. (in the web application the text fields will be auto filled from the values already existing in the database - except for the change password fields. Password will be updated only when the user chooses to change it and if the conditions are met). On clicking the button 'Save Changes', the following queries will be executed:

Query 2.2:

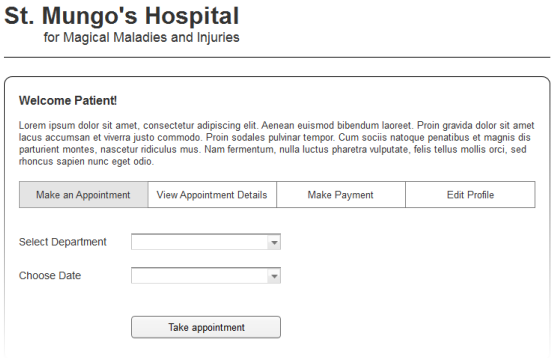
```
UPDATE      users
SET         password = "Claire123",
            name = "Claire Bennet",
            contact = 123456789,
            addr = "1203 W mill avenue",
            emailid = "claire.b@xyz.com"
WHERE       userid = "Claire1";
```

Query 2.3:

```
UPDATE      nurses
SET         workshift = 0
WHERE       userid = "Claire1";
```

Patients' Pages

Patient Page 1 - Make an appointment



Once a patient logs in, in the first screen he is given an option to make an appointment. After he selects the department and the date for the appointment, and clicks on 'Take appointment', the following queries will be executed:

Query 3.1:

```
SELECT      COUNT(*)
FROM        appointments
WHERE       apt_date = "2014-10-29"
AND         dept_id = 3;
```

If the number of appointments for the date and department chosen by the patient is greater than 10 (assuming that a department permits only 10 appointments per day), then the user will be prompted to choose a different date. If the count is less than 10, then the following query will be executed:

Query 3.2:

```
INSERT INTO appointments (apt_date, patient_id, dept_id)
VALUES ("2014-10-29","Pooja1", 3);
```

A bill number for the patient's appointment is automatically generated; say 5679. This bill number should not already exist. To check this;

Query 3.3:

```
SELECT      COUNT(*)
FROM        treatments
WHERE       bill_no = 5679;
```

If the above query results in 0, then we perform the following operations, otherwise we generate another bill number.

Also, a doctor from the patient's chosen department is randomly chosen:

Query 3.4:

```
SELECT      userid
FROM        belongs_to
WHERE       dept_id = 3;
```

One doctor from the above set is chosen, and the following insert statements are executed; eg 'Anderson1'

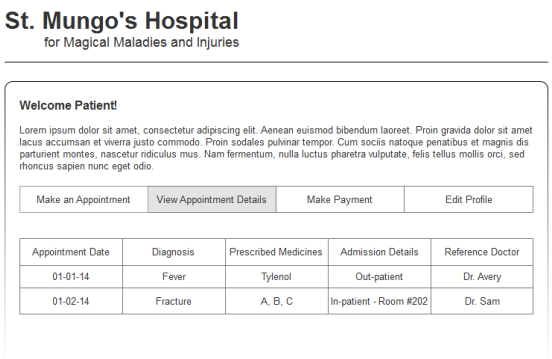
Query 3.5:

```
INSERT INTO treats
VALUES ('pooja1', 'Anderson1',5679);
```

Query 3.6:

```
INSERT INTO treatments(bill_no, apt_date)
VALUES (5679, "2014-10-29");
```

Patient Page 2 - View appointment details



Query 3.7:

```
SELECT      A.apt_date, T.diagnosis,
T.prescribed_med, D.name
FROM        appointments A,
treatments T, users D, treats T2
WHERE       A.patient_id = "Mohan1"
AND         A.patient_id =
T2.patient_id
AND         T2.doctor_id = D.userid
AND         T2.bill_no = T.bill_no;
```

If the doctor changes the status of a patient from out-patient to in-patient, the patient will be assigned to a default room temporarily, but will be given the option to choose a different type of room. Based on the type of room the user selects ('general', 'deluxe', 'luxury', 'premium'), a room of that type will be allotted to the patient. If no room of that type is available, i.e. all rooms of that type are full, then the user will be asked to choose a different type of room.

Query 3.8

```
SELECT      R.roomno
FROM        rooms R
INNER JOIN roomtype RT
ON          R.typeid = RT.typeid
WHERE       RT.rtype = 'Premium'
AND         R.roomno NOT IN
            (SELECT I.roomno
            FROM in_patients I);
```

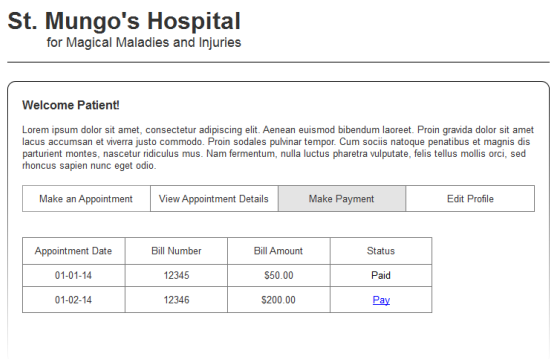
One of the rooms from the above set will be assigned to the patient.

To get the admission details of the patient (if the patient is admitted or not - if yes, then the room number):

Query 3.9:

```
SELECT      userid, roomno
FROM        in_patients
WHERE       userid = "Mohan1";
```

Patient Page 3 - Make Payment



Assuming that the treatment costs are covered by the patient's health insurance (not in scope of this application), we calculate only the total cost of the prescribed medicines and the room rent (if the patient was admitted)

Query 3.10:

```
SELECT      A.apt_date, T.bill_no, T.prescribed_med
FROM        appointments A, treatments T, treats T2
WHERE       A.patient_id = "Mohan1"
AND         A.patient_id = T2.patient_id
AND         T2.bill_no = T.bill_no;
```

We get the prescribed medicines as a comma separated list. We programmatically split this list to get the individual names of the medicines, and for each medicine we find its cost and add it to the bill amount:

Query 3.11:

```
SELECT      cost
FROM        pharmacy
WHERE       medicine_name = "m2";
```

Query 3.12:

```
SELECT      I.admit_date, I.discharge_date, RT.rent
FROM        roomtype RT, rooms R, in_patients I
WHERE       I.userid = 'Mohan1'
AND         I.roomno = R.roomno
AND         R.typeid = RT.typeid;
```

For the room rent obtained above, we multiply it with the number of days the patient was admitted

Bill amount = [(discharge date - admit date) * room rent] + [sum of all costs of prescribed medicines]

Patient Page 4 - Edit Profile

Just like nurses, patients can also edit their profiles. The same assumptions we had for nurses hold true even for patients.

Query 3.13:

```
UPDATE      users
SET         password = "mohan123",
            name = "Mohan C",
            contact = 987654321,
            addr = "1000 S Road",
            emailid =
"mohan.c@xyz.com"
WHERE       userid = "Mohan1";
```

Query 3.14:

```
UPDATE      patients
SET         age = 25,
            gender = 0
WHERE       userid = "Mohan1";
```

Doctors' Pages

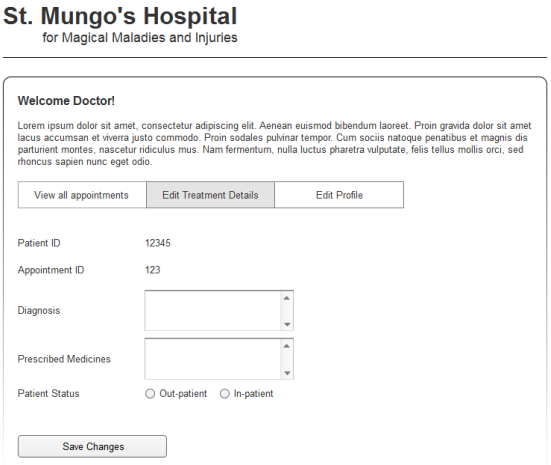
Doctors Page 1 - View all appointments

Once a doctor logs in, he can view the list of the patients he is treating or will treat and their respective treatment dates. The link 'Edit Treatment' will be active for only those patients whose appointment date is current date or before the current date (the link will be inactive for future dates).

Query 4.1:

```
SELECT      T.patient_id, U.name, T2.apt_date
FROM        treats T, users U, treatments T2
WHERE       T.doctor_id = "Anderson1"
AND         T.bill_no = T2.bill_no
AND         T.patient_id = U.userid;
```

Doctor Page 2 - Edit Treatment Details



Like mentioned above, once the doctor clicks on the edit treatment details for a particular patient, populates the fields and clicks on the 'Save changes' button, the following query will be executed:

Query 4.2:

```
UPDATE      treatments
SET         diagnosis = "Malaria",
            prescribed_med = "m1,m2,m3"
WHERE       apt_date = "2014-11-17"
AND         bill_no IN
            (SELECT    T.bill_no
            FROM       treats T, users U
            WHERE      T.doctor_id = "Anderson1"
            AND        T.patient_id ="Mohan1");
```

When the doctor changes the status of a patient from out-patient to in-patient, then the following queries will be executed:

Firstly, a general type of room is chosen:

Query 4.3:

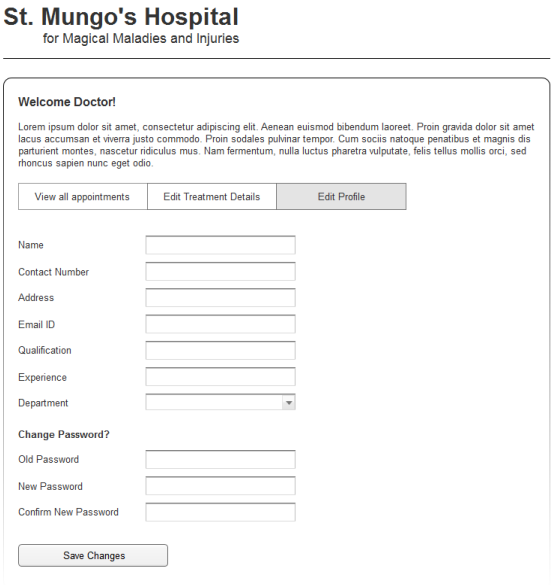
```
SELECT      R.roomno
FROM        rooms R, roomtype T
WHERE       T.rtype = "general";
```

One room from the above set is chosen say 101 and the values are inserted into in_patients table;

Query 4.4:

```
INSERT INTO in_patients
VALUES ("Mohan1", "2014-10-29", NULL, 101);
```

Doctor Page 3 - Edit Profile



Just like patients and nurses, doctors can edit their profiles. Same assumptions made for nurses and patients hold true for doctors.

Query 4.5:

```
UPDATE users
SET   password = "anderson123",
      name = "Anderson H",
      contact = 4802398472,
      addr = "1400 S Univ Rd",
      emailid = "andy.h@xyz.com"
WHERE userid = "Anderson1";
```

Query 4.6:

```
UPDATE      doctors
SET         qualification = "Neurosurgeon",
            experience = 11
WHERE       userid = "Anderson1";
```

Query 4.7:

```
UPDATE      belongs_to
SET         dept_id = 3
WHERE       userid = "Anderson1";
```

Lastly, following are some queries that would not execute because of the triggers that we have established to maintain consistency in the database.

```
UPDATE users SET TYPE = 3 WHERE userid = 'vikas.v';
```

Changing value of `type` field is not permitted.

```
INSERT INTO users VALUES ('Vijay1', 'Vijay', 'vijay', 4567456894,
'Mill Tempe', 'vijay@gmail.com', 5);
```

Violating the `type` field domain constraints (should be between 0 and 3)

```
INSERT INTO in_patients VALUES ("Pooja1","2014-09-17","2014-08-
17",103);
```

Inserting a record which has an admit date smaller than discharge date; not allowed.