16.05.2021

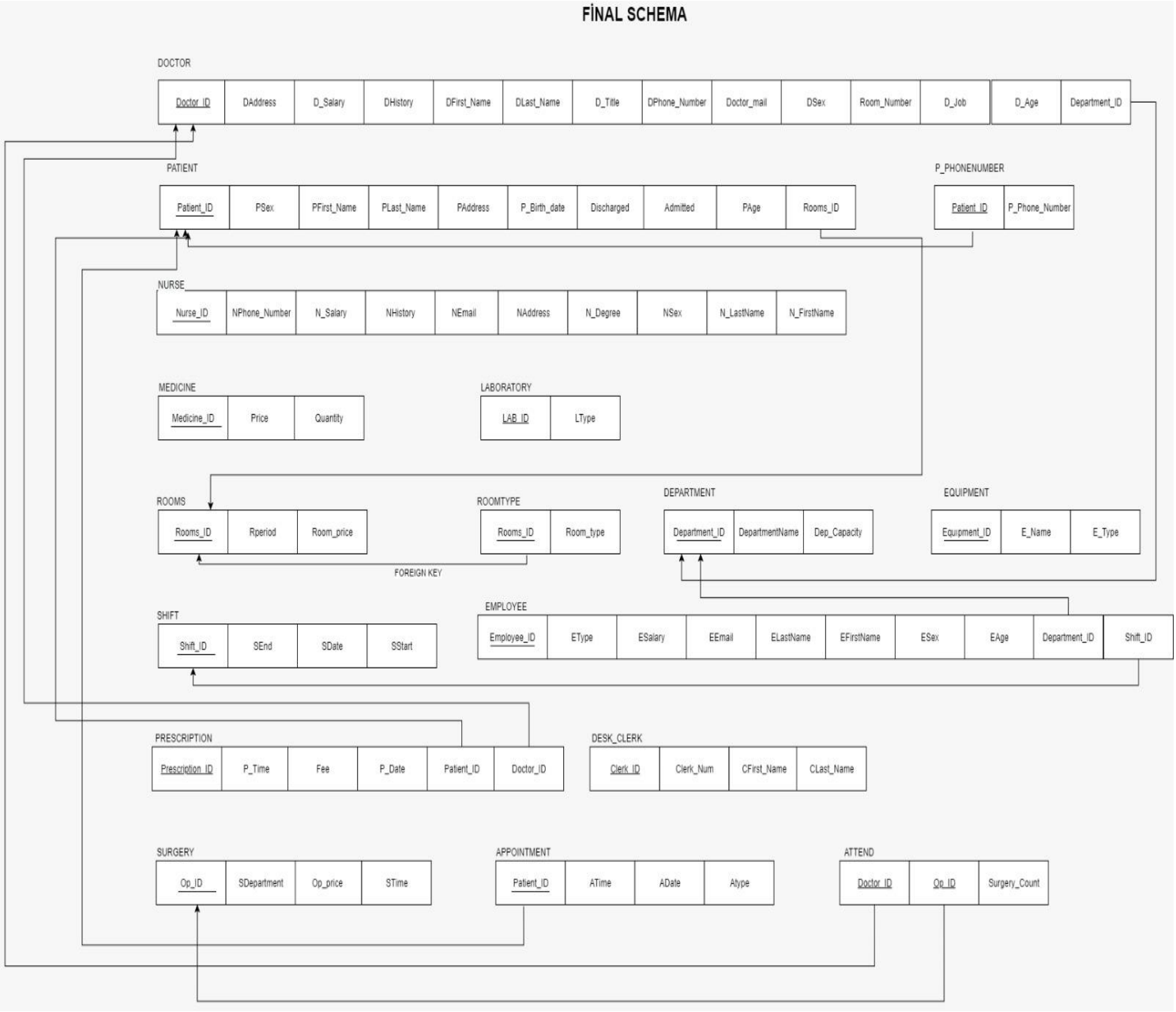
TED UNIVERSITY

CMPE 232 Relational Databases
Spring 2021
Project Phase 2
Group 7
Hospital Management System

Ayfer Feyza Şahin 30932496534

Zeynep Ayca Tanışlı 33326013720

Zülal Karın 12622989076



THE CHANGES IN PHASE 1

Firstly, we entered data in the tables we created in Phase 1 for our Hospital Management System. For this, we divided the tables and paid attention to the foreign keys. In addition to Phase 1, we also added the Equipment table and we added relevant data to our table. In phase 1 we forget to create equipment table according to our ER diagram. Unlike phase 1, we added an equipment table. And there was a foreign key that we forgot to add in phase 1. As a result, we added foreign key in addition to the doctor table. The foreign key we added is the department ID.

DESCRIPTIONS OF THE TABLES

JOINS: With 'join' we have combined our two join and three join tables. Firstly, we combined the Room and RoomType tables to create a table with the prices, types, and IDs of our rooms. Then we created a table that gives the average prices of the inner join and the rooms where RoomType is not X-Ray. Again, from the Room and RoomType tables, where we combined with 'join', we listed the room price in descending order. Likewise, by combining the worker and shift tables, we sorted the workers with shifts in alphabetical order. By combining the Employee and department tables, we found the identities and names of the female employees in the Cardiology department with higher salaries than some of the female employees. We used outer join to find the number of surgeries of doctors because we also brought the data in both tables connected to each other. Then we combined the attend and surgery table, it lists the operations with an operation fee greater than 3700. We chose the Surgery, Attend and Doctor tables to combine the three tables with ‘Left Outer Join’, so we collected the Op_ID, Op_price, Doctor_ID, DFirst_Name, Dlast_Name, Surgery_Count and SDepartment data in one table.

SUBQUERY: When we move on to subqueries, we wrote a query showing nurses with over 3900 salaries. In this way, we created a subquery that returns multiple rows. For From Clause Subquery, we created a table that lists all unique d_titles and averages their salaries between 12000 and 18000. For the Scalar Subquery, we have listed all departments along with the number of employees in each department.

In our sql project, we used at least one WHERE, AND/OR, AVG, COUNT, SUM, ORDER BY, ASC/DESC, GROUP BY, HAVING, EXISTS/NOT EXISTS, LIKE, DISTINCT, IS NULL, IN/NOT IN, BETWEEN AND, MIN/MAX, SOME, ALL,ANY, String operations, set operations (UNION/EXCEPT/INTERSECT) operators in all our tables. In this way, we created tables such as Lists the empty rooms which patient does not exist, List the drugs that start with A in order to be able to edit the hospital records, lists of employees over 30 years of age and with salaries between 3000 and 5000, list of the ID's and names of female employees with salary greater than that of some female employee in the Cardiology department etc. and enriched our project.

SPECIAL QUERIES:

1)We made our special queries with the division of labour. For this first of all we added Department_ID as foreign key to our doctor table. In this way, we were able to list the number of operations the doctor entered and the departments he/she was in. Our aim was listing the patients who stays in hospital and their departments where the patient has MR appointment today and grouped by department. For this, we used the operators inner join appointment, inner join department. With the query were appointment. atype = 'MR' and appointment. adate = '2021-04-20', we identified patients who had their MR during the day and listed our outputs according to their departments.

2)In our second special query, our aim was to list the doctor's information and the number of surgeries the doctor worked on in the ‘general surgery’ department. We combined Doctor table with the join operators with the Department and Attend tables to list this query. Finally, we specified the specific department with the query WHERE ATTEND.Doctor_ID is not null and department. departmantname = 'GeneralSurgery'.

SPECIFICATION OF THE TABLES

```
CREATE TABLE PATIENT(
Patient_ID int not null,
PSex varchar(255),
PFirst_Name varchar(255),
PLast_Name varchar(255),
PAddress varchar(255),
P_Birth_date date,
Discharged varchar(255),
Admitted varchar(255),
PAge int not null,
Rooms_ID int not null,
primary KEY(Patient_ID),
foreign KEY(Rooms_ID)references ROOMS(Rooms_ID)
);
```

1) We identified each patient with Patient_ID. We made sure that each row is different by using a Primary Key. Each patient has name, address, sex, phone number birth date, age, and details (for discharge procedures). Patients get appointment and take prescription. We defined the age of the patient as the derived attribute because we keep it together with the date of birth. Patients Takes_treatments in departments and Get_Treatments from doctor. In addition, samples go to the laboratory from the patients, patients can be assigned to rooms and receive treatment from doctors. Thanks to these connections and relationships, we can clarify our first query. The patient can receive an appointment, choose MR as the appointment type, so we can list the patients who have an MR appointment. Thanks to Patient_Detailes, which we define as a composite attribute, we can reach hospitalized patients.

```
CREATE TABLE PRESCRIPTION(
Prescription_ID int not null,
Patient_ID int not null,
Doctor_ID int not null,
P_Time time,
Fee int,
P_Date date,
primary KEY(Prescription_ID),
foreign KEY(Patient_ID)references PATIENT(Patient_ID),
foreign KEY(Doctor_ID)references DOCTOR(Doctor_ID)
);
```

2) We defined prescriptions as Prescription_ID. Each prescription has its time, hour, and cost. Besides, they contain medicine.

```
CREATE TABLE MEDICINE(
Medicine_ID int not null,
Price int,
Quantity varchar(255),
primary KEY(Medicine_ID)
);
```

3) We defined the medicine as Medicine_ID. Each medicine has price and quantity.

```
CREATE TABLE APPOINTMENT(
Patient_ID int not null,
ATime time,
ADate date,
Atype varchar(255),
foreign KEY(Patient_ID)references PATIENT(Patient_ID)
);
```

4) We defined the appointment system of the hospital as a weak entity and each appointment has a time, date, and type.

```
CREATE TABLE DESK_CLERK(
Clerk_ID int not null,
Clerk_Num char(11),
CFirst_Name varchar(255),
CLast_Name varchar(255),
primary KEY(Clerk_ID)
);
```

5) We designed a Desk Clerk that delivers these appointments. We have identified each receptionist with their ID numbers (Clerk_ID) and the receptionists have their names and numbers.

```
CREATE TABLE LABORATORY(
LAB_ID int not null,
LType varchar(255),
primary KEY(LAB_ID)
);
```

6) We defined the laboratories with LAB_ID. Each laboratory has a type. (Such as Clinical Biochemistry, Clinical Microbiology, Clinical Haematology and Pathology)

```
CREATE TABLE ROOMS(
Rooms_ID int not null,
Rperiod date,
Room_price int,
primary KEY(Rooms_ID)
);
```

7) We identified the rooms where the patients are assigned with Rooms_IDs. There are periods, types, and fees to determine when rooms are available. Since more than one patient can stay in a

room, the type of the rooms is multivalued attribute. Also, all rooms include equipment and nurse who governs rooms.

```
CREATE table EQUIPMENT(
Equipment_ID int not null,
E_Name varchar(255),
E_Type varchar(255),
primary key(Equipment_ID)
);
```

8) We identified the equipment in the hospital with Equipment_ID. Equipment has its type and name. Such as Sterilizer, Operating Table, Traction, Ceiling Lights, Air Blender, Gynaecological Table, Anaesthesia Device, Bedside Monitor ...

```
CREATE TABLE NURSE(
Nurse_ID int not null,
NPhone_Number char(11),
N_Salary int not null,
NHistory varchar(255),
NEmail varchar(255),
NAddress varchar(255),
N_Degree varchar(255),
NSex varchar(255),
N_LastName varchar(255),
N_FirstName varchar(255),
primary KEY(Nurse_ID)
);
```

9) There are nurses assigned to rooms in our hospital and we identified these nurses with Nurse_ID. Each nurse has a name, surname, gender, number, salary, degree, address, and email. In addition, we added NHistory to assign nurses to cases related to it in the future.

```
CREATE TABLE Doctor (
    Doctor_ID int not NULL,
    DLast_Name varchar(255),
    DFirst_Name varchar(255),
    DAddress varchar(255),
    D_Salary int,
    D_Job   varchar(255),
    DPhone_Number  char(11),
    Doctor_mail  varchar(255),
    DSex      varchar(15),
    Room_Number int,
    D_Title    varchar(255),
    D_Age   int not null,
    Department_ID  int not null,
    foreign KEY(Department_ID)references Department(Department_ID),
   primary KEY(Doctor_ID)
);
```

10) A doctor identified by Doctor_ID each doctor has age, department, salary, address, history, name, surname, phone number, mail, title, sex, and room number. Also, a doctor write prescription and attend surgery. We put the Surgery_Count attribute for the Attend relation. In this way, we can count the surgeries requested from us in the second query from us.

```
CREATE TABLE SURGERY(
Op_ID int not null,
SDepartment varchar(255),
Op_price int,
STime time,
primary KEY(Op_ID)
);
```

11) We identified the surgeries as Op_ID. Surgeries have price, time, and departments.

```
CREATE TABLE Employee (
Employee_ID int not null,
Department_ID int not null,
Shift_ID int not null,
EType  varchar(25),
ESalary int,
EEmail varchar(255),
ELastName varchar(255),
EFirstName varchar(255),
ESex varchar(255),
EAge int not null,
primary KEY(Employee_ID),
foreign KEY(Department_ID)references Department(Department_ID),
foreign KEY(Shift_ID)references SHIFT(Shift_ID)
);
```

12) We defined the employees by Employee_ID. Employees have their name, age, sex, email, salary, and type (cleaners, security guards, technicians ...). Employees work in various departments and serve shifts.

```
CREATE TABLE DEPARTMENT(
Department_ID int not null,
DepartmentName varchar(255),
Dep_Capacity int,
primary KEY(Department_ID)
);
```

13) We have identified the departments with the Department_IDs. Each department has a name and capacity.

```
CREATE TABLE SHIFT(
Shift_ID int not null,
SEnd time,
SDate date,
SStart time,
primary KEY(Shift_ID)
);
```

14) We defined the shifts with Shift_ID. Shifts have a start time (SStart), end time (SEnd) and date. (Since it is a start and end key word, we did not use them directly.)

```
CREATE TABLE P_PHONENUMBER(
Patient_ID int not null,
P_Phone_Number char(11),
foreign KEY(Patient_ID)references PATIENT(Patient_ID)
);
```

15) We created the P_PHONENUMBER table and assigned the Patient_ID as the foreign key and the P_Phone_Number as the primary key. In this way, we provided the user with the opportunity to reach our patients.

```
CREATE TABLE ROOMTYPE(
Rooms_ID int not null,
Room_type varchar(255),
foreign KEY(Rooms_ID)references ROOMS(Rooms_ID)
);
```

16)We created ROOMTYPE for our hospital rooms. We assigned the Rooms_ID as the foreign key Room_type foreign key. In this way, we were able to create rooms such as X-Ray, EEG, EKG, Emergency Room and DexaScan etc.

```
CREATE TABLE ATTEND(
Doctor_ID int not null,
Op_ID int not null,
Surgery_Count int,
foreign KEY(Doctor_ID)references DOCTOR(Doctor_ID),
foreign KEY(Op_ID)references SURGERY(Op_ID)
);
```

17) -We have created an ATTEND table that gives the number of doctors' participation in surgeries. We assigned Doctor_ID and Op_ID as foreign key and Surgery_Count as primary key.

EXPLANATION OF THE JAVA APPLICATION PROGRAM

In our Java code, first we connected our postgreSQL and after welcome the user, we created admin and user options . We have ensured that all selections are created and selected by using if-else statements and while loops.

```
Welcome to Our Hospital Database Management System!
1) Admin Login
2) User Login
3) Exit
```

```
1) Add a new doctor
2) Delete a nurse
3) List All equipments
4) Return to main menu
Please enter the number of your choice:
```

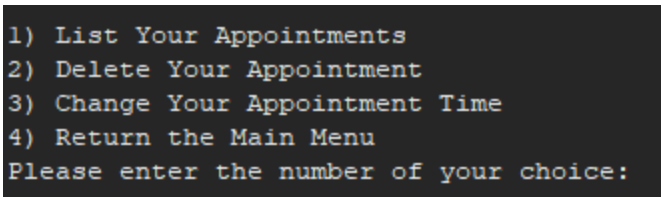
We created add, delete and list features for admin option. In this way, we can add doctors, delete nurses and list the equipment to our hospital system in SQL.

```
1) Doctor
2) Patient
3) Return to main menu
Please enter the number of your choice:
```

In the user part, We have created doctor and patient entry options. These options also contain some options in themselves.

```
1) Update Your Adress
2) List Your Surgery Count
3) Return the Main Menu
Please enter the number of your choice:
```

Doctors at the hospital can update their addresses and learn the number of surgeries.



We created three selection for patients; they can list the appointments they have made, delete their appointments and change the appointment times. In addition, we've included the option to return to the main menu in all options by using while loop. In this way, we made it possible for the user to return