GTU Department of Computer Engineering
CSE 411 – Spring 2023
Homework #1

Hasan Mutlu
1801042673

User Requirements

Employee Management: The system should have roles for doctors, nurses and secretaries and manage information about these employees such as which department do they work in.

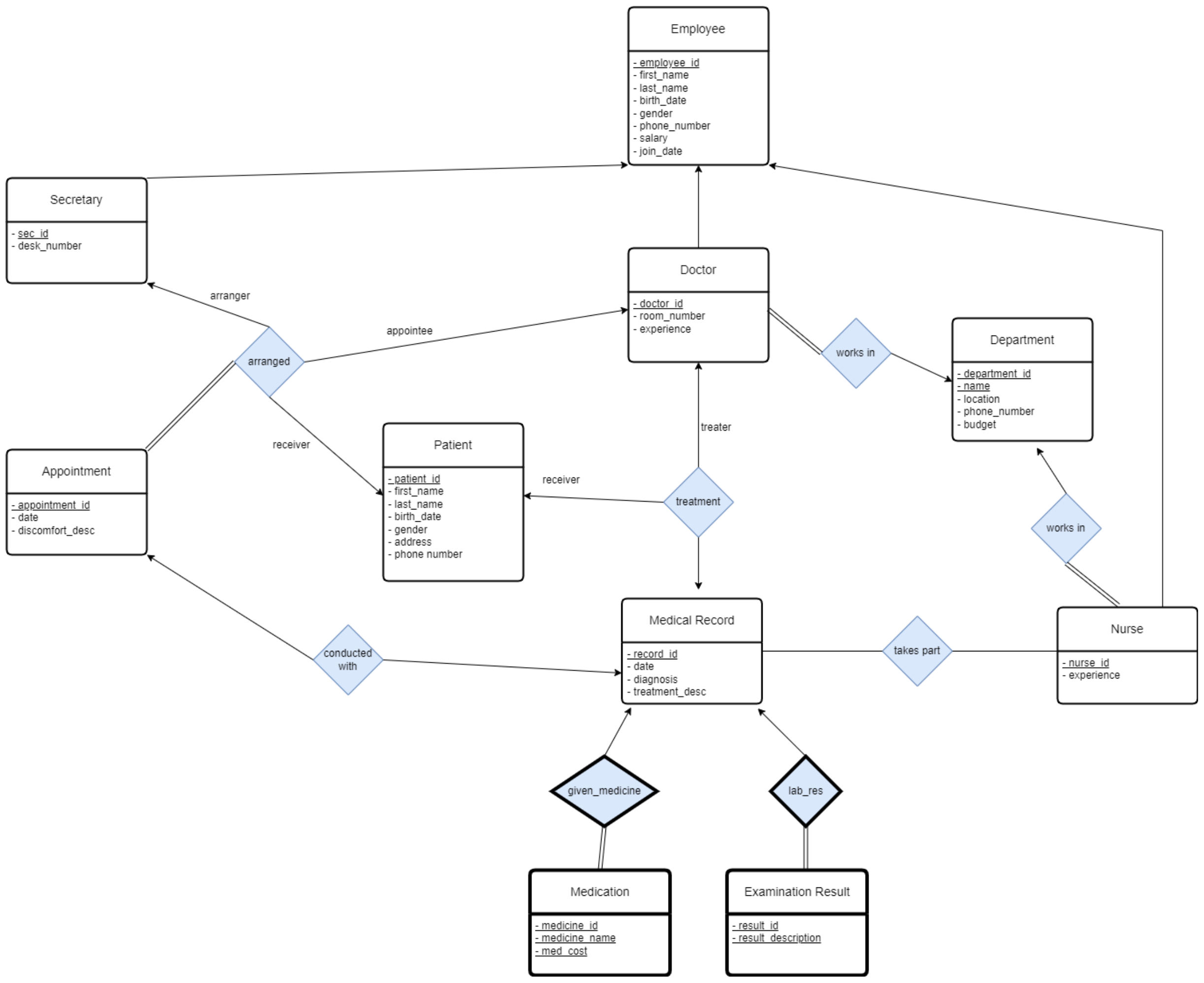
Patient Management: The system should be able to store patient information.

Appointment Management: The system should allow patients to get appointment from a doctor with the help of secretary. (It is assumed that only secretaries can arrange appointments to the patients.)

Treatment Management: The system should store treatment information such as patient, doctor, nurses involved and treatment details.

Reports Management: The system should manage external information about each treatment such as given medicine or examination results (e.g. blood analysis).

ER Diagram



Weak Entities: Medication and Examination Result are weak entities. (There is no double line option in the tool I used for diagram so they are covered with bolder borders.)

Roles:

- ➔ Secretary is arranger in “arranged” relationship.
- ➔ Doctor is appointee in “arranged” relationship.
- ➔ Patient is receiver in “arranged” relationship.
- ➔ Doctor is treater in “treatment” relationship.
- ➔ Patient is receiver in “treatment relationship.

Ternary Relationship: “treatment” is ternary relationship.

Inheritances:

- ➔ Doctor inherits Employee
- ➔ Nurse inherits Employee
- ➔ Secretary inherits Employee

Relationship definitions:

- ➔ works_in: Each doctor works in one department.
- ➔ works_in: Each nurse works in one department.
- ➔ arranged: Each appointment is arranged by one secretary, to one patient, appointed to one doctor.
- ➔ conducted: One treatment and medical record is conducted after one appointment.
- ➔ treatment: One medical record is made after one treatment of a patient by one doctor.
- ➔ takes_part: One or more nurse(s) take part in medical record of a treatment.
- ➔ given_medicine: Medication is given after one medical record.
- ➔ lab_res: Examination Results (e.g. blood analysis) are made after one medical record.

Functional Dependencies

Employee:

- ➔ employee_id -> first_name, last_name, birth_date, gender, phone_number, salary, join_date
- ➔ join_date -> salary

Secretary:

- ➔ sec_id -> desk_number

Nurse:

- ➔ nurse_id -> experience

Doctor:

- ➔ doctor_id -> room_number, experience

Department:

- ➔ department_id -> name, location, phone_number, budget
- ➔ name -> location, budget

Patient:

- ➔ patient_id -> first_name, last_name, birth_date, gender, address, phone_number

Appointment:

- ➔ appointment_id -> date, secretary_id, patient_id, doctor_id

Medical Record:

- ➔ record_id -> date, diagnosis, treatment_desc, appointment_id
- ➔ diagnosis -> treatment_desc

Medication:

- ➔ medicine_id -> medicine_name, med_cost

Examination Results:

- ➔ result_id -> result_description

BCNF or 3NF Tables

Example Schema: Employee

Employee(employee_id, first_name, last_name, birth_date, gender, phone_number, salary, join_date)

Functional Dependencies:

- ➔ employee_id -> first_name, last_name, birth_date, gender, phone_number, salary, join_date
- ➔ salary -> join_date

The table is in 3NF because there are no partial or transitive dependencies.

Example Schema: Department

Department(department_id, dept_name, location, phone_number, budget)

Functional Dependencies:

- ➔ department_id -> name, location, phone_number, budget
- ➔ name -> location, budget

The table is in 3NF because there are no partial or transitive dependencies.

Example Schema: Medical Record

Medical Record(record_id, date, diagnosis, treatment_desc, appointment_id)

Functional Dependencies:

- ➔ record_id -> date, diagnosis, treatment_desc, appointment_id
- ➔ diagnosis -> treatment_desc

The table is in 3NF because there are no partial or transitive dependencies.

SQL Functions

Table SQL Function:

```
CREATE FUNCTION get_appointment_patients(date_input DATE)
RETURNS TABLE (
    patient_id INT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    gender CHAR(1),
    appointment_id INT,
    appointment_date DATE
) AS
BEGIN
    RETURN QUERY SELECT p.patient_id, p.first_name, p.last_name, p.gender,
a.appointment_id, a.date
    FROM patient p
    JOIN appointment a ON p.patient_id = a.patient_id
    WHERE a.date = date_input;
END;
```

Function with For Loop:

```
CREATE FUNCTION get_department_avg_salary(dept_id INT)
RETURNS NUMERIC AS
DECLARE
    emp_salary NUMERIC;
    emp_count INT := 0;
    total_salary NUMERIC := 0;
    avg_salary NUMERIC := 0;
BEGIN
    FOR emp_salary IN SELECT salary FROM employee WHERE employee_id IN (SELECT
doctor_id FROM doctor WHERE department_id = dept_id)
    LOOP
        emp_count := emp_count + 1;
        total_salary := total_salary + emp_salary;
    END LOOP;
    IF emp_count > 0 THEN
        avg_salary := total_salary / emp_count;
    END IF;
    RETURN avg_salary;
END;
```

Function with Input, Temporary Variable and Output:

```
CREATE FUNCTION get_medication_cost(record_id INT)
RETURNS NUMERIC AS
DECLARE
    medication_cost NUMERIC := 100;
    total_cost NUMERIC := 0;
BEGIN
    FOR medication_cost IN SELECT m.cost FROM medication m JOIN
medical_record_medication mr ON m.medicine_id = mr.medicine_id WHERE
mr.record_id = record_id
    LOOP
        total_cost := total_cost + medication_cost;
    END LOOP;
    RETURN total_cost;
END;
```

Triggers

Trigger with "referencing old row as" and "referencing new row as" together:

A trigger that updates the department's budget when a doctor is assigned to a different department:

```
CREATE OR REPLACE FUNCTION update_budget()
RETURNS TRIGGER AS
BEGIN
    UPDATE department
    SET budget = budget - OLD.salary
    WHERE department_id = OLD.department_id;

    UPDATE department
    SET budget = budget + NEW.salary
    WHERE department_id = NEW.department_id;

    RETURN NEW;
END;

CREATE TRIGGER update_department_budget
AFTER UPDATE OF department_id ON doctor
FOR EACH ROW
WHEN (OLD.department_id != NEW.department_id)
EXECUTE FUNCTION update_budget();
```

Trigger with “when” of “if”

A trigger that updates the medical record's diagnosis if the treatment description is changed.

```
CREATE OR REPLACE FUNCTION update_diagnosis()
RETURNS TRIGGER AS
BEGIN
    UPDATE medical_record
    SET diagnosis = NEW.treatment_desc
    WHERE medical_record.appointment_id = NEW.appointment_id;

    RETURN NEW;
END;

CREATE TRIGGER update_medical_record_diagnosis
BEFORE UPDATE OF treatment_desc ON medical_record
FOR EACH ROW
WHEN (OLD.treatment_desc != NEW.treatment_desc)
EXECUTE FUNCTION update_diagnosis();
```

Trigger with “for each row”

A trigger that inserts a new medical record when a treatment is conducted.

```
CREATE OR REPLACE FUNCTION update_diagnosis()
RETURNS TRIGGER AS
BEGIN
    UPDATE medical_record
    SET diagnosis = NEW.treatment_desc
    WHERE medical_record.appointment_id = NEW.appointment_id;

    RETURN NEW;
END;

CREATE TRIGGER update_medical_record_diagnosis
BEFORE UPDATE OF treatment_desc ON medical_record
FOR EACH ROW
WHEN (OLD.treatment_desc != NEW.treatment_desc)
EXECUTE FUNCTION update_diagnosis();
```

Trigger with "for each statement"

A trigger that updates the budget of a department by subtracting the cost of medication from the department's budget when medication is given.

```
CREATE OR REPLACE FUNCTION update_department_budget()
RETURNS TRIGGER AS
DECLARE
    medication_cost INTEGER;
BEGIN
    SELECT SUM(cost)
    INTO medication_cost
    FROM medication
    WHERE medicine_name = NEW.medicine_name;

    UPDATE department
    SET budget = budget - medication_cost
    WHERE department_id = NEW.department_id;

    RETURN NEW;
END;

CREATE TRIGGER update_department_budget_trigger
AFTER INSERT ON medication
FOR EACH STATEMENT
EXECUTE FUNCTION update_department_budget();
```

Transactions

Transaction to update a patient's information and their medical record.

```
BEGIN TRANSACTION;

UPDATE patient
SET phone_number = '123-456-7890', address = 'Democracy Neihboorhood, 14th May
Street 23/2'
WHERE patient_id = 12345678901;

UPDATE medical_record
SET diagnosis = 'Flu', treatment_desc = 'It is determined that the patient has
flu. Medicine is given and resting is advised.'
WHERE record_id = 123456789;

COMMIT;
```

Transaction to update salaries of all doctors and nurses in one department

```
BEGIN TRANSACTION;

UPDATE employee
SET salary = salary * 1.1
WHERE employee_id IN (
    SELECT employee_id
    FROM doctor
    WHERE department_id = 3
) OR employee_id IN (
    SELECT employee_id
    FROM nurse
    WHERE department_id = 3
);

COMMIT;
```