Matthew Carroll

Prof. Sara Riazi

ITCS 3160

4/19/22

Final Report

Abstract

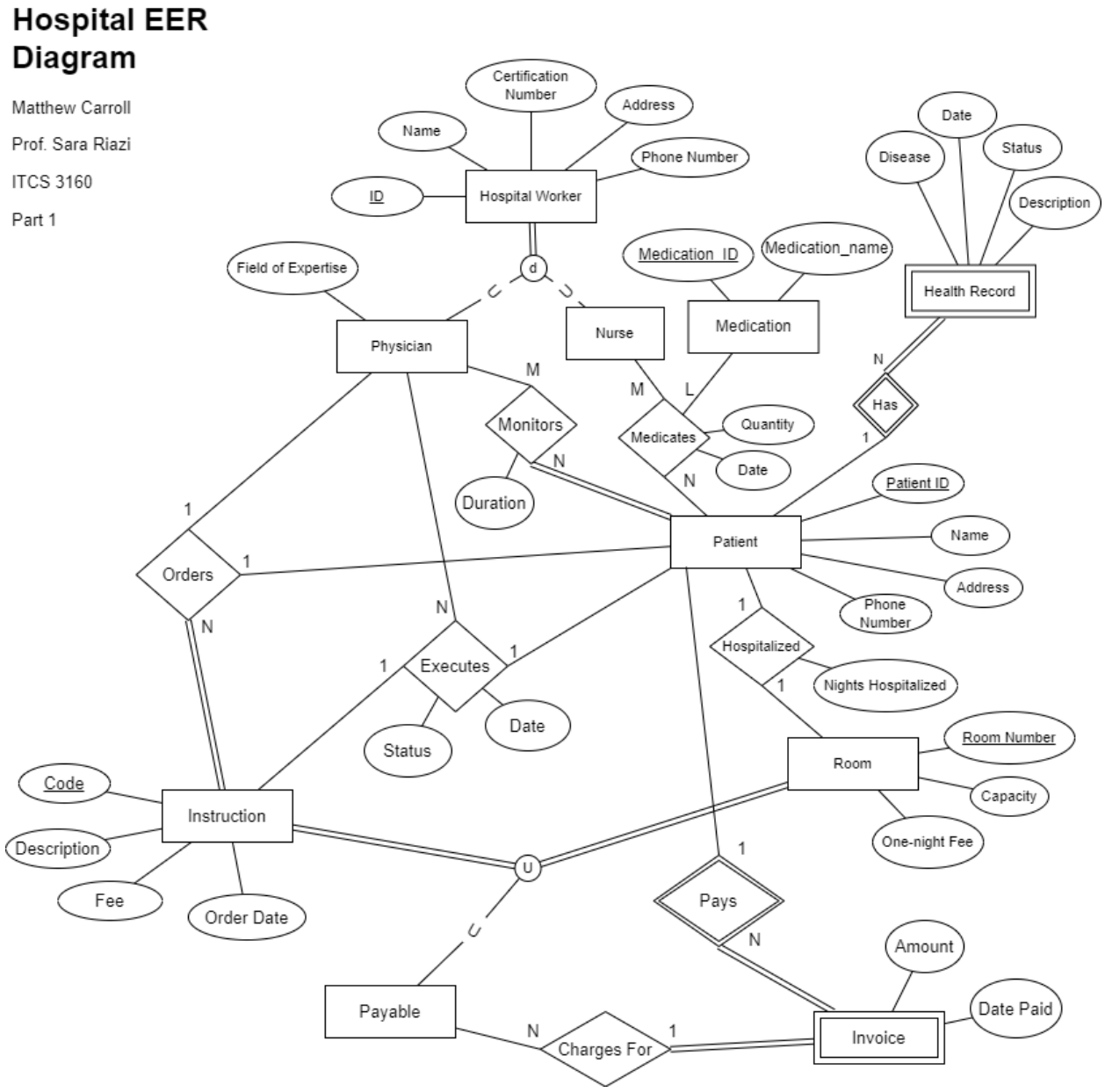
Since this is the third part to the whole Database Design and Implementation Final Project, this report will be a collection of new and older pieces of work from the semester from previous parts of the project (parts 1 and 2). Thus, the assumptions, diagram, and relations and keys will be taken directly (unless specifically modified) from the other portions of the project. All other parts, such as the views, triggers, and queries will be new as of the creation of this third and final part of the project.

This entire project was a process of designing, implementing, and modifying a database based on given instructions. I created a design and mapping structure, implemented the design into the MySQL environment, created and inserted data into that implementation, and then produced queries and other extra features for bonus functionality. All in all, this database can store information about patients and hospital workers, and keep track of the monitoring of patients, order execution, and payments made in the hospital environment.

Assumptions

- Invoice is defined by its relationship to Patients in specific (an invoice is assigned to a patient), so the relationship between the two is defining
- Since every Instruction must be ordered when it is created, the date ordered attribute is assigned to the Instruction itself rather than the relationship
- Rooms are assumed to only hold one Patient at any time
- Hospital Workers must be either a Nurse or Physician in the case of this database

(E)ERD



Changes:

- Medication was added as its own relation with medication_id and medication_name attributes
- The medicates relationship is now three-way between Nurse, Patient, and Medication, with L to M to N (many to many to many) cardinality.

Relations and Keys

Entity Tables:

** Since hospital workers have to be either physicians or nurses, the hospital entity is not a relation in the database **

// physicians and nurses
Physician(physician_id, full_name, certification_number, address, phone_number, field_of_expertise)
    Primary key: {physician_id}
    Foreign key: {None}
Nurse(nurse_id, full_name, certification_number, address, phone_number)
    Primary key: {nurse_id}
    Foreign key: {None}

// patients, health records, and their medications
Patient(patient_id, full_name, address, phone_number)
    Primary key: {patient_id}
    Foreign key: {None}
Health_Record(patient_id, disease, record_date, record_status, descr)
    Primary key: {patient_id}
    Foreign key: {patient_id references patient(patient_id)}
Medication(medication_id, medication_name)
    Primary key: {medication_id}
    Foreign key: {None}

// payable items like instructions and rooms as well as invoices
Instruction(instruction_code, descr, fee, order_date)
    Primary key: {instruction_code}
    Foreign key: {None}
Room(room_number, capacity, fee)
    Primary key: {room_number}
    Foreign key: {None}
Invoice(invoice_id, patient_id, total)
    Primary key: {invoice_id}
    Foreign key: {patient_id references patient(patient_id)}

Relationship Tables:
** These tables are used to represent relations, and thus all have foreign keys **
** Some 1 - N relationships are not represented by their own relation table, and instead, the attributes for the N side of the relationship are added to the corresponding table (e.g. Has_Record is represented by a patient_id in the Health_Record table) **

// monitors, medicates, and hospitalized relationships
Monitors(physician_id, patient_id, duration)
    Primary key: {physician_id, patient_id}

Foreign key: {physician_id references physician(physician_id), patient_id references patient(patient_id)}

Medicates(nurse_id, patient_id, medication_id, quantity, med_date)
Primary key: {nurse_id, patient_id, medication_id}
Foreign key: {nurse_id references nurse(nurse_id), patient_id references patient(patient_id), medication_id references medication(medication_id)}

Hospitalized(patient_id, room_number, num_nights)
Primary key: {patient_id, room_number}
Foreign key: {patient_id references patient(patient_id), room_number references room(room_number)}

// orders and executes three-way relationships

Orders(instruction_code, physician_id, patient_id)
Primary key: {instruction_code}
Foreign key: {instruction_code references instruction(instruction_code), physician_id references physician(physician_id), patient_id references patient(patient_id)}

Executes(instruction_code, nurse_id, patient_id)
Primary key: {instruction_code, nurse_id}
Foreign key: {instruction_code references instruction(instruction_code), nurse_id references nurse(nurse_id), patient_id references patient(patient_id)}

// payment relationship that records a payment made at a time for an invoice

Payment(invoice_id, payment_id, patient_id, amount, date_paid)
Primary key: {invoice_id, payment_id}
Foreign key: {invoice_id references invoice(invoice_id), patient_id references patient(patient_id)}

Views

```
CREATE VIEW physician_basic_info AS
    SELECT physician_name, certification_number, field_of_expertise
    FROM physician;
```

Selects physicians just by their name, certification number, and field of expertise.

For the sake of someone searching the different physicians, displaying the address or personal ID of the physician may be confidential information and unnecessary. Thus, the above view provides a simplified, protected set of attributes.

```
CREATE VIEW monitor_participants AS
    SELECT ph.physician_name, p.patient_name, duration
    FROM monitors m
    JOIN physician ph ON ph.physician_id = m.physician_id
    JOIN patient p ON p.patient_id = m.patient_id;
```

Displays the physician and patients' name for a specific monitoring situation, as well as the duration of said monitoring.

Within a hospital, if a patient is having issues that were unrecognized in previous screenings or monitoring, then an executive may want to find who monitored them and if the duration was adequate. This is the particular information displayed by the above view.

```
CREATE VIEW instruction_info AS
    SELECT o.instruction_code, ph.physician_name, n.nurse_name, p.patient_name
    FROM orders o
    JOIN executes e ON o.instruction_code = e.instruction_code
    JOIN physician ph ON ph.physician_id = o.physician_id
    JOIN nurse n ON n.nurse_id = e.nurse_id
    JOIN patient p ON p.patient_id = o.patient_id
```

Shows the physician that ordered an instruction for a patient, the nurses that executed the instruction, and the patient themselves.

Since the two acts of ordering and executing instructions are separated, the above view provides a quick and concise display for the connected information for each patient.

Triggers

1. ```
   delimiter //
   CREATE TRIGGER payment_limit
   BEFORE INSERT ON payment
   FOR EACH ROW
   BEGIN
       SET @total = (select total_cost from invoice i where i.invoice_id =
   NEW.invoice_id);
       IF NEW.amount > @total THEN
           SET NEW.amount = @total;
       END IF;
   END; //
   delimiter ;
   ```

   The trigger payment_limit simply limits patients to making payments that are equal to or less than the total cost of the bill for the invoice. It wouldn't make sense to pay more than you owe.

2. ```
   delimiter //
   CREATE TRIGGER correct_room_size
   AFTER INSERT ON hospitalized
   FOR EACH ROW
   BEGIN
   ```

```
        SET @curr_size = (select count(*) from hospitalized h where h.room_number =
NEW.room_number);
    SET @room_capacity = (select capacity from room r where r.room_number =
NEW.room_number);
        IF @curr_size = @room_capacity + 1 THEN
            SET @message = CONCAT('room number ', NEW.room_number, ' is
full');
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @message;
      DELETE FROM hospitalized WHERE patient_id = NEW.patient_id;
        END IF;
END; //
delimiter ;
```

The trigger correct_room_size will raise an error if you attempt to insert a patient into a room that is full.

3.
```
delimiter //
CREATE TRIGGER medicate_limit
BEFORE INSERT ON medicates
FOR EACH ROW
BEGIN
        IF NEW.quantity > 2000 THEN
            SET NEW.quantity = 2000;
        END IF;
END; //
delimiter ;
```

   If it is true that 2000 mg/mL is a large amount of medication to be given at a time, the trigger medicate_limit will ensure that no patient is recorded for being given more than 2000 mg/mL, assuming it must have been a mistake.

Queries, Descriptions, and Results

```
-- 1. Shows all physicians whose fields end with "ology"
select physician_name, field_of_expertise from physician
where field_of_expertise like "%ology";
```

	physician_name	field_of_expertise
▶	Dr. Top Hat	Epidemiology
	Dr. Anderson	Biology

```
-- 2. Shows all patients who were given morphine
select patient_name as  patients_given_morphine
from patient p
```

```
join medicates m on m.patient_id = p.patient_id
join medication med on med.medication_id = m.medication_id
where medication_name = 'Morphine';
```

patients_given_morphine
Neil Armweak
John Jordan

```
-- 3. Shows all patients given over 200 mg/ml of medication
select patient_name, medication_name, quantity from medicates m
join patient p on p.patient_id = m.patient_id
join medication med on med.medication_id = m.medication_id
where quantity > 200;
```

patient_name	medication_name	quantity
Neil Armweak	Morphine	1000.00
Payton Manning	Caffeine	230.00
John Jordan	Morphine	500.00

```
-- 4. Show all total prices for invoices with the names of the patients associated
select patient_name, invoice_id, total_cost from invoice i
join patient p on p.patient_id = i.patient_id;
```

patient_name	invoice_id	total_cost
Matthew Carroll	1	1300.00
Jeffrey Jeffman	2	4050.00
Payton Manning	3	1500.00
Neil Armweak	4	1200.00
George Jr.	5	9950.00
John Jordan	6	410.00

```
-- 5. Find all instructions with more than one nurse executing it
select i.instruction_code from executes e
join instruction i on i.instruction_code = e.instruction_code
join nurse n on n.nurse_id = e.nurse_id
group by i.instruction_code
having count(*) > 1;
```

instruction_code
afsd658
i3ep56a
qly6i7o

```
-- 6. Select all room numbers that have hospitalized more than one person
select room_number, count(patient_id) as number_of_patients
from hospitalized
group by room_number
```

```
having count(patient_id) > 1;
```

room_number	number_of_patients
1010	2
2010	2

```
-- 7. Show the average cost for all invoices
select avg(total_cost) as avg_invoice_cost from invoice;
```

avg_invoice_cost
3068.333333

```
-- 8. Lists all patients not medicated by Nurse Simone Biles
select patient_name, n.nurse_name from medicates m
join patient p on p.patient_id = m.patient_id
join nurse n on n.nurse_id = m.nurse_id
where n.nurse_id not in
        (select nurse_id from nurse where nurse_name = "Nurse Simone Biles");
```

patient_name	nurse_name
Neil Armweak	Nurse Nina
George Jr.	Gina Georgina Nursina
Jeffrey Jeffman	Nurse John Cena
Matthew Carroll	Nurse Evelina Ileena

```
-- 9. Select all rooms that have not hospitalized a patient given morphine
select distinct room_number as room_without_morphine_patient from room
where room_number not in
        (select room_number from hospitalized h
        join patient p on p.patient_id = h.patient_id
        join medicates m on m.patient_id = p.patient_id
        join medication med on med.medication_id = m.medication_id
        where medication_name = 'Morphine');
```

room_without_morphine_patient
1010
3005
4545

```
-- 10. Find all patients who have not made any payments
select patient_name as patients_without_payments
from patient
where patient_id not in
        (select patient_id from invoice i
    join payment pay on pay.invoice_id = i.invoice_id);
```

	patients_without_payments
▶	George Jr.
	John Jordan

-- 11. Shows all medication names that are smaller than 10 characters
select medication_name as small_medication_names from medication
where length(medication_name) < 10;

	small_medication_names
▶	Zithromax
	Caffeine
	Morphine

-- 12. Shows all physicians and nurses who have a certification number over 8
select physician_name as worker_name, certification_number
from physician
where certification_number > 8
UNION
select nurse_name as worker_name, certification_number
from nurse
where certification_number > 8;

	worker_name	certification_number
▶	Doctor Who	10
	Gina Georgina Nursina	11
	Nurse Simone Biles	77
	Nurse John Cena	19
	Nurse Evelina Ileena	12

-- 13. Select all patient id's from patients hospitalized in or after 2013
select h.patient_id, record_date from hospitalized h
join health_record hr on hr.patient_id = h.patient_id
where year(record_date) >= 2013;

	patient_id	record_date
▶	1	2019-05-01 00:00:00
	2	2013-07-29 00:00:00
	5	2022-03-31 00:00:00

-- 14. Select the highest payment that each payer has made
select patient_name, max(amount) as max_payment from payment pay
join invoice i on i.invoice_id = pay.invoice_id
join patient p on p.patient_id = i.patient_id
group by patient_name;

	patient_name	max_payment
▶	Matthew Carroll	1000.00
	Jeffrey Jeffman	4050.00
	Payton Manning	750.00
	Neil Armweak	8000.00

```
-- 15. Display the number of inactive and active patients
select record_status, count(*) as number_of_patients
from health_record
group by record_status;
```

	record_status	number_of_patients
▶	Inactive	5
	Active	1

Additional Information

The sql files will be listed below with their corresponding content:

- Hospital-schema.sql - tables, views, triggers
- Hospital-data.sql - all inserted data
- Hospital-query.sql - all 15 queries