Author: Marco A. Tinoco Sosa

Final Report - Hospital Project

Section One: Assumptions

1) Each room can belong to multiple patients, but one patient should always be in one room per stay in the hospital.

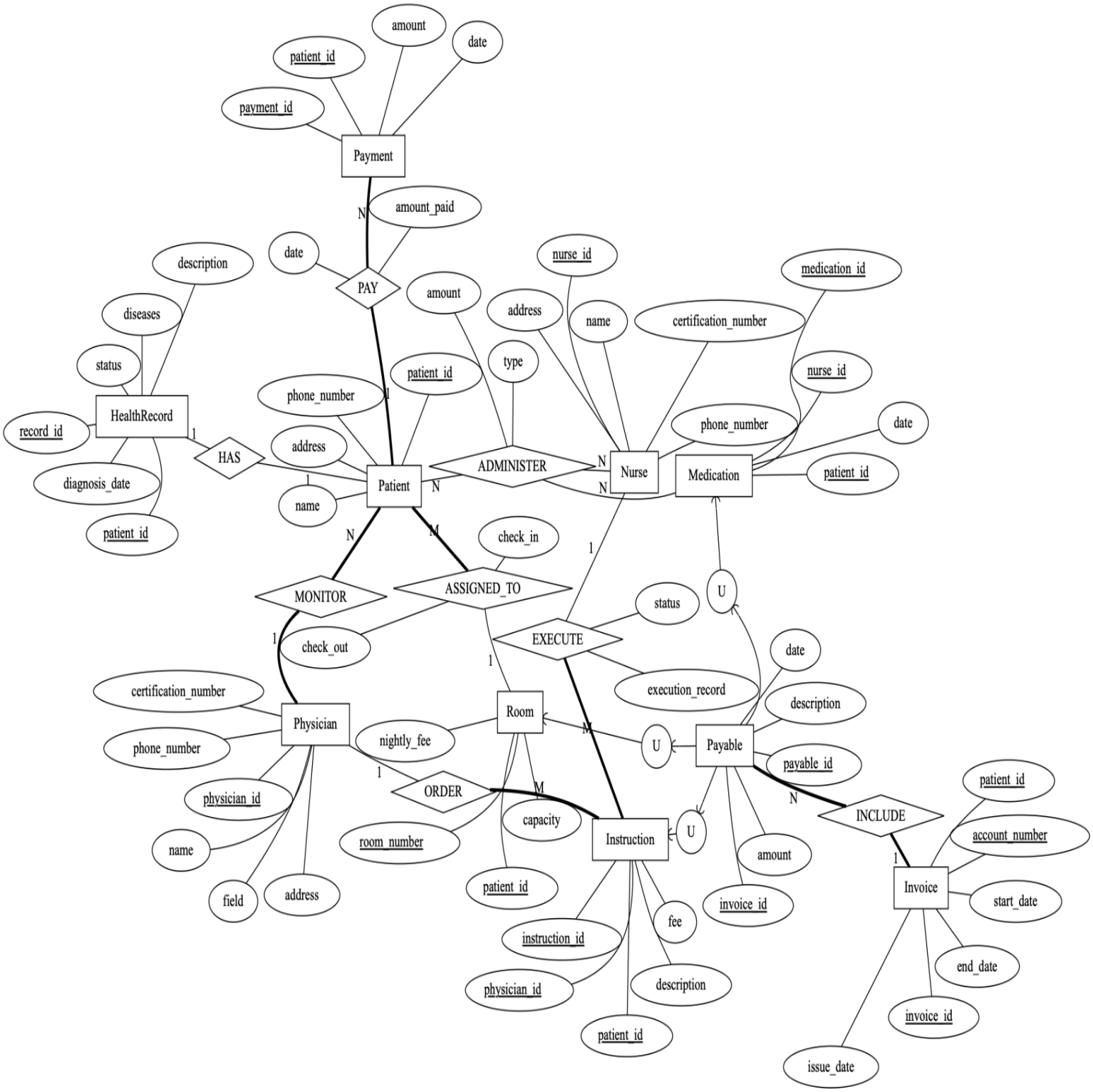
2) Each instruction must be executed by one nurse.

3) Each physician at the hospital must have at least one patient at all times.

4) Room assignments, medications, and instructions are all listed under Payables as a union.

5) Every patient must have a health record.

Section Two: E(ERD)



Section Three: Relations and Keys

red = primary key, blue = foreign key

Patient (patient_id, name, address, phone_number)

Physician (physician_id, name, certification_number, field, address, phone_number)

Nurse (nurse_id, name, certification_number, address, phone_number)

Room (room_number, capacity, nightly_fee)

HealthRecord (record_id, diseases, diagnosis_date, status, description, patient_id)
*patient_id references Patient(patient_id)

Invoice (invoice_id, issue_date, start_date, end_date, account_number, patient_id)
*patient_id references Patient(patient_id)

Payable (payable_id, amount, date, description, invoice_id)
*invoice_id references Invoice(invoice_id)

Payment (payment_id, amount, date, patient_id)
*patient_id references Patient(patient_id)

Instruction (instruction_id, description, fee, physician_id, patient_id, payable_id)
*physician_id references Physician(physician_id)
*patient_id references Patient(patient_id)
*payable_id references Payable(payable_id)

Medication (medication_id, type, amount, date, payable_id, nurse_id, patient_id)
*payable_id references Payable(payable_id)
*nurse_id references Nurse(nurse_id)
*patient_id references Patient(patient_id)

PatientRoomAssignment (assignment_id, check_in, check_out, patient_id, room_number)
*patient_id references Patient(patient_id)
*room_number references Room(room_number)

PatientMonitoring (monitoring_id, start_date, end_date, physician_id, patient_id)
*physician_id references Physician(physician_id)
*patient_id references Patient(patient_id)

InstructionExecution (execution_id, status, execution_record, nurse_id, instruction_id)
*nurse_id references Nurse(nurse_id)
*instruction_id references Instruction(instruction_id)

Section Four: Views and Descriptions

View One

Body:

```
CREATE VIEW Patient_Health_summary AS
SELECT Patient.name, diseases, diagnosis_date, status, description
FROM Patient
JOIN HealthRecord ON Patient.patient_id = HealthRecord.patient_id;
```

Description: Combines patient name with their health conditions, status, and diagnosis details.
Discussion: Summarizes health conditions in a readable format for dashboards or reports. It enables quick access to health trends and ongoing cases.

View Two

Body:

```
CREATE VIEW Nurse_Instructions AS
SELECT Nurse.name, Instruction.instruction_id, status, execution_record
FROM Nurse
JOIN InstructionExecution ON Nurse.nurse_id = InstructionExecution.nurse_id
JOIN Instruction ON InstructionExecution.instruction_id = Instruction.instruction_id;
```

Description: Display each nurse's involvement in executing physician instructions, including status and execution records.
Discussion: The reason why this is useful is because it can help evaluate the nursing staff and ensure compliance with physician orders. Also useful for audits or follow-ups.

View Three

Body:

```
CREATE VIEW Medication_Administered AS
SELECT Nurse.name AS Nurse, type AS Medication, Patient.name AS Patient, date
FROM Medication
JOIN Nurse ON Medication.nurse_id = Nurse.nurse_id
JOIN Patient ON Medication.patient_id = Patient.Patient_id;
```

Description: Lists which nurse administered what medication to which patient, including dates.
Discussion: The reason why this is useful is because it facilitates medication tracking, improving patient safety and ensuring proper billing and inventory control.

Section Five: Triggers and Descriptions

Trigger One

Body:

```
DELIMITER //
CREATE TRIGGER default_room_fee
BEFORE INSERT
ON Room
FOR EACH ROW
BEGIN
	IF NEW.nightly_fee IS NULL THEN
		SET NEW.nightly_fee = 800.00;
	END IF;
END;
//
DELIMITER ;
```

Description: Sets a default nightly fee of $800 if none is provided when inserting a new room.
Discussion: The reason why this trigger is useful is because it prevents incomplete data entry and enforces business rules by assigning a value for pricing.

Trigger Two

Body:

```
DELIMITER //
CREATE TRIGGER verify_room_availability
BEFORE INSERT
ON PatientRoomAssignment
FOR EACH ROW
BEGIN
	IF EXISTS (
   SELECT *
   FROM PatientRoomAssignment
   WHERE  room_number = NEW.room_number AND NEW.check_in <= check_out AND
New.check_out >= check_in
   ) Then
		SET NEW.room_number = (SELECT MAX(room_number) + 1 FROM Room);

      INSERT INTO Room VALUES (NEW.room_number,2, 500.00);
	END IF;
END;
//
DELIMITER ;
```

Description: Automatically assigns a new room if the specified room is unavailable for the requested dates, and adds it to the room table.
Discussion: The reason why this trigger is useful is because it ensures no room is double-booked and expands room inventory when needed, which enhances data integrity and availability.

Trigger Three
Body:

```
DELIMITER //
CREATE TRIGGER update_instructionExecution
BEFORE UPDATE
ON InstructionExecution
FOR EACH ROW
BEGIN
        IF NEW.execution_record = 'Done' THEN
   SET NEW.status = 'Completed';
        END IF;
END;
//
DELIMITER ;
```

Description: If the execution_record is set to ‘Done’, the status is automatically marked as ‘Completed’.
Discussion: The reason why this trigger is useful is because it automates status tracking and reduces human error by syncing execution records and statuses.

Section Six: Queries, Descriptions, and Results

Query One

Description: Get the name of a patient with a specific ID

Query: SELECT name FROM patient WHERE patient_id = 1;

Screenshot:

Result Grid

name
Aashika Lilaramani

Query Two

Description: Find a patient by phone number

Query: SELECT name FROM patient WHERE phone_number = '312-555-1234';

Screenshot:

Result Grid

name
Sarah Diaz

Query Three

Description: Retrieve health details for specific patient

Query: SELECT name, diseases, diagnosis_date, description FROM patient JOIN HealthRecord ON patient.patient_id = HealthRecord.patient_id WHERE patient.patient_id = 2;

Screenshot:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	diseases	diagnosis_date	description
Alexis Pelaez Alvarez	Diabetes Type 2	2025-02-20	Completed insulin therapy, maintaining with diet

Query Four

Description: List all completed instruction executions

Query: SELECT * FROM InstructionExecution WHERE status = 'Completed';

Screenshot:

Result Grid | Filter Rows: | Edit: | Export/Import:

execution_id	nurse_id	instruction_id	status	execution_record
12001	201	5001	Completed	BP recorded twice daily as ordered
12002	202	5002	Completed	Insulin administered before each meal
12004	204	5004	Completed	MRI completed at Rush, results normal
12005	205	5005	Completed	Done
NULL	NULL	NULL	NULL	NULL

Query Five

Description: List names of patients with ongoing health issues, sorted in reverse order

Query: SELECT name FROM patient JOIN HealthRecord ON patient.patient_id = healthRecord.patient_id WHERE status = 'Ongoing' ORDER BY patient.name DESC;

Screenshot:

Result Grid | Filter

name
Sarah Diaz
Marco Tinoco Sosa
Gonzalo Smith
Aashika Lilaramani

Query Six

Description: See which nurses executed which instructions for which patient

Query: SELECT Nurse.nurse_id, Nurse.name AS Nurse, Patient.name AS Patient, description FROM Nurse JOIN InstructionExecution ON Nurse.nurse_id = InstructionExecution.nurse_id JOIN Instruction ON InstructionExecution.instruction_id = Instruction.instruction_id JOIN Patient ON Instruction.patient_id = Patient.patient_id;

Screenshot:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

nurse_id	Nurse	Patient	description
201	Nurse Emily Davis	Aashika Lilaramani	Monitor blood pressure twice daily
202	Nurse Carlos Lopez	Alexis Pelaez Alvarez	Administer insulin shots before meals
203	Nurse Amy Lee	Marco Tinoco Sosa	Check oxygen levels during asthma attacks
204	Nurse Daniel Pink	Sarah Diaz	Perform MRI scan at Rush University
205	Nurse Jessica White	David de la O	Physical therapy sessions at Northwestern

Query Seven

Description: Get the total amount of all payments

Query: SELECT SUM(amount) FROM payment;

Screenshot:

Result Grid

SUM(amount)
2000.00

Query Eight

Description: Get the average amount of medication administered, rounded to two decimals

Query: SELECT ROUND(AVG(amount), 2) FROM Medication;

Screenshot:

Result Grid | Filter Row

	ROUND(AVG(amount), 2)
▶	12.60

Query Nine

Description: List patients and their assigned physician with monitoring dates

Query: SELECT Patient.name, Physician.name, PatientMonitoring.start_date, PatientMonitoring.end_date FROM Patient JOIN PatientMonitoring ON Patient.patient_id = PatientMonitoring.patient_id JOIN Physician ON PatientMonitoring.physician_id = Physician.physician_id WHERE field = "Surgery";

Screenshot:

Result Grid | Filter Rows: | Export: | Wr

	name	name	start_date	end_date
▶	David de la O	Dr. Thomas Guzman	2025-06-05	2025-06-14

Query Ten

Description: Show patients checked into rooms on a specific date, with room details

Query: SELECT DISTINCT Patient.name, room.capacity, room.nightly_fee FROM Patient JOIN PatientRoomAssignment ON PatientRoomAssignment.patient_id = patient.patient_id JOIN Room ON PatientRoomAssignment.room_number = Room.room_number WHERE check_in = '2025-06-03';

Screenshot:

Result Grid | Filter Rows:

	name	capacity	nightly_fee
▶	Marco Tinoco Sosa	3	300.00

Query Eleven

Description: Find patients with a payable amount between $350 and $450

Query: SELECT Patient.name FROM patient JOIN Invoice ON patient.patient_id = Invoice.patient_id JOIN Payable ON Invoice.invoice_id = Payable.invoice_id WHERE amount >= 350.00 AND amount <= 450.00 ORDER BY Patient.name DESC;

Screenshot:

Result Grid | Filter Ro

name
Sarah Diaz
Alexis Pelaez Alvarez
Aashika Lilaramani

Query Twelve

Description: Find patients with the highest payable amount

Query: SELECT Patient.name, Payable.amount FROM Patient JOIN Invoice ON patient.patient_id = Invoice.patient_id JOIN Payable ON Invoice.invoice_id = Payable.invoice_id WHERE Payable.amount IN (SELECT MAX(Payable.amount) FROM Payable JOIN Invoice ON Payable.invoice_id = Invoice.invoice_id JOIN Patient ON Invoice.patient_id = Patient.patient_id);

Screenshot:

Result Grid | Filter Rows:

name	amount
Gonzalo Smith	1050.00

Query Thirteen

Description: List patients who have at least one medication above 8.00

Query: SELECT Patient.name, Medication.type FROM Medication JOIN Patient ON Medication.patient_id = Patient.patient_id WHERE EXISTS (SELECT * FROM Medication M2 WHERE M2.patient_id = Medication.patient_id AND amount > 8.00);

Screenshot:

Result Grid | Filter Rows:

name	type
Aashika Lilaramani	Lisinopril
Alexis Pelaez Alvarez	Insulin
Marco Tinoco Sosa	Albuterol
Sarah Diaz	Sumatriptan

Query Fourteen

Description: List patients who have no medication above 8.00

Query: SELECT Patient.name, Medication.type FROM Medication JOIN Patient ON Medication.patient_id = Patient.patient_id WHERE NOT EXISTS (SELECT * FROM Medication M2 WHERE M2.patient_id = Medication.patient_id AND amount > 8.00);

Screenshot:

Result Grid | Filter Rows:

	name	type
▶	David de la O	Ibuprofen

Query Fifteen

Description: Find names of patients who received medication above 15.00

Query: SELECT name FROM Patient WHERE patient_id IN (SELECT Patient_id FROM Medication WHERE amount > 15.00);

Screenshot:

Result Grid

	name
▶	Sarah Diaz

Query Sixteen

Description: List patients who received medication amounts above the average

Query: SELECT name FROM Patient JOIN Medication ON Patient.patient_id = Medication.patient_id where Medication.amount > (SELECT AVG(amount) FROM Medication);

Screenshot:

Result Grid | Filt

	name
▶	Aashika Lilaramani
	Sarah Diaz

Query Seventeen

Description: Count how many health records have a status of 'Ongoing'

Query: SELECT COUNT(*) AS Ongoing FROM HealthRecord WHERE status = 'Ongoing';

Screenshot:

Result Grid

	Ongoing
▶	4

Section Seven: Transactions and Descriptions

Transaction One

Body:

```
START TRANSACTION;
INSERT INTO Patient VALUES (6, 'Gonzalo Smith', '350 E Cermak, Chicago, IL',
'773-123-4321');
INSERT INTO PatientRoomAssignment VALUES (10008, 6, 101, '2025-07-31', '2025-08-05');
INSERT INTO Healthrecord VALUES (1006, 6, 'High Cholesterol', '2025-07-31', 'Ongoing',
'Lifestyle changes and medication');
COMMIT;
```

Description: Insert a new patient, their room assignment, and health record, all in one operation.
Discussion: The reason why this transaction is useful is because it ensures consistency, either all patient admission data is added, or none is, preventing partial/incomplete entries.

Transaction Two

Body:

```
START TRANSACTION;
INSERT INTO Invoice VALUES (7006, '2025-07-31', '2025-07-27', '2025-07-30', 'CHI7006', 6);
INSERT INTO Payable VALUES (8006, 1050.00, '2025-07-31', 'Room 101 $350x3', 7006);
COMMIT;
```

Description: Insert a new invoice and its corresponding payable item together.
Discussion: The reason why this transaction is useful is because it keeps financial records consistent and avoids mismatches between invoice entries and their associated charges.