ISTE 230

Instructor: Dr. Khalil Al Hussaini

HOSPITAL MANAGEMENT DATABASE PROJECT REPORT

SANIKA PANDIT

PRAISLIN PETER

AAMINA

DATE – 30TH APRIL

Group no – 3.

Manifesto

Contribution:

1. Sanika Pandit:

- Led the conceptual and logical design phase.
- Defined the business scenario and justified the need for a database.
- Identified core tasks for database users.
- Identified entities, attributes, primary keys, relationships, and relationship constraints.
- Created the E-R diagram using computer-aided tools.
- Transposed the E-R diagram into relations.

2. Praislin Peter:

- Led the physical design phase.
- Normalized relations, ensuring efficient data storage and retrieval.
- Implemented the database solution using SQL.
- Demonstrated the database's functionality during the presentation.
- Composed relational algebra notations for business operations.
- Created SQL queries demonstrating various SQL clauses and functions.
- Ensured the completeness and accuracy of SQL queries and functions.

3. Aamina:

- Incorporated feedback into the design process to ensure the database solution addressed real-world challenges effectively.
- Assisted in identifying entities, attributes, and relationships.
- Reviewed and provided input on the E-R diagram and relational design.

Equal Contribution:

- All team members actively participated in brainstorming sessions, discussions, and decision-making processes throughout the project.
- Each member contributed to the project's success by fulfilling their assigned tasks and responsibilities diligently.

This manifesto highlights the individual contributions of each team member and emphasizes the collective effort towards completing the project successfully.

Table of Contents:

PHASE 1

PHASE1_Q1_A.

Scenario: Hospital Management System

a. Scenario Overview:

In the hospital management scenario, a relational database is required to efficiently manage various aspects of hospital operations, including patient appointments, medical records, prescriptions, lab orders, and patient-doctor interactions. The hospital management system involves organizing and maintaining vast amounts of data related to patient care, administrative tasks, and medical staff activities. It encompasses patients, doctors, pharmacists, and lab technicians, all of whom interact with the database to carry out their respective tasks effectively.

PHASE1_Q1_b. Justification for a Database:

A relational database is crucial for managing the diverse data involved in hospital operations. Without a database, managing patient appointments, medical records, prescriptions, lab orders, and vitals monitoring would be chaotic and error prone. A database ensures data integrity, security, and efficient retrieval, facilitating smooth hospital management.

PHASE1_Q1_c. Core Tasks for Database Users:

1.View and Manage Appointments:

Patients can view upcoming appointments and manage them according to doctor schedules.

2.Patient Management:

Doctors can access patient profiles with medical history for diagnosis and treatment.

3.Prescriptions and Medication Management:

Doctors, and pharmacists can generate prescriptions and manage medication lists and the patients can access it as well.

4.Test and Lab Order Management:

Lab technicians conduct tests and update reports, while doctors can order and access lab reports.

PHASE1_Q2_B.

Based on the provided information, let's identify the entities, attributes, primary keys (PKs), relationships, and relationship constraints (multiplicities):

Entities:

1. LabReport:
    - Lab_report_id (PK)
    - *doctor_id* (Foreign Key)
    - *patient_id* (FK)
    - *lab_tech_id* (FK)
    - Test_type.
    - test_result.
    - report_date.
    - report_status.

    Relationships:

    i. LabReport to LabTech Relationship:
        - Type: One-to-Many
        - Cardinality: Each LabTech can be associated with zero or more LabReports, but each LabReport must be associated with one LabTech and one LabTech only.

    ii. LabReport to Doctor Relationship:
        - Type: One-to-Many
        - Cardinality: Each Doctor can be associated with zero or more LabReports, but each LabReport must be associated with one Doctor only.

    iii. LabReport to Patient Relationship:
        - Type: One-to-Many
        - Cardinality: Each Patient can be associated with zero or more LabReports, but each LabReport must be associated with one Patient only.
2. LabTech:
    - lab_tech_id (PK)
    - first_name
    - last_name
    - Gender
    - email_address.
    - phone_number.
    - *Department_id* (FK)

    Relationships:

i. LabTech to Department Relationship:
- Type: One-to-Many
- Cardinality: Each Department can be associated with zero or more LabTechs, but each LabTech must be associated with one Department only.

3. Doctor:
   - doctor_id (PK)
   - first_name
   - last_name
   - start_date.
   - Gender
   - email_address.
   - phone_number.
   - Department_id (FK)

   Relationships:

   i. Doctor to Department Relationship:
   - Type: One-to-Many
   - Cardinality: Each Department must be associated with one or more Doctors, but each doctor must be associated with one Department only.

4. Department:
   - Department_id (PK)
   - Department_name
   - Department_description
5. Prescription:
   - Prescription_id (PK)
   - appointment_id (FK)
   - medicine_name.
   - Prescription_date
   - Prescription_instruction

   Relationships:

   i. Prescription to Appointment Relationship:
   - Type: One-to-Many
   - Cardinality: Each Prescription must be linked to exactly one Appointment and an Appointment can have 0 or more prescriptions.
6. Appointment:
   - Appointment_id (PK)
   - patient_id (FK)
   - doctor_id (FK)
   - Appointment_date
   - Appointment_status

   Relationships:

   i. Appointment to Patient Relationship:

- Type: One-to-Many
- Cardinality: A Patient can have zero or more Appointments, and appointment must be linked to one patient only.

ii. Appointment to Doctor Relationship:

- Type: One-to-Many
- Cardinality: A Doctor can have zero or more Appointments, and appointment must be linked to one doctor only.

7. Service:
    - service_id (PK)
    - department_id (FK)
    - cost
    - service_type.
    - service_description.

    Relationships:

    i. Service to Department Relationship:
        - Type: One-to-Many
        - Cardinality: A Department can offer one or more Services

8. Patient:
    - patient_id (PK)
    - first_name
    - last_name
    - date_of_birth.
    - Gender
    - email_address.
    - Number

9. InvoiceService:
    - invoice_service_id (PK)
    - invoice_id (FK)
    - service_id (FK)
    - Quantity

    Relationships:

    i. InvoiceService to Invoice Relationship:
        - Type: One-to-Many
        - Cardinality: An Invoice can include one or more InvoiceServices but a InvoiceService must belong to a single invoice.

    ii. InvoiceService to Service Relationship:
        - Type: One-to-Many
        - Cardinality: A Service can be billed through one or more InvoiceServices, and an invoiceService is linked to a single service

10. Invoice:

- invoice_id (PK)
- patient_id (FK)
- invoice_date.
- subtotal
- status

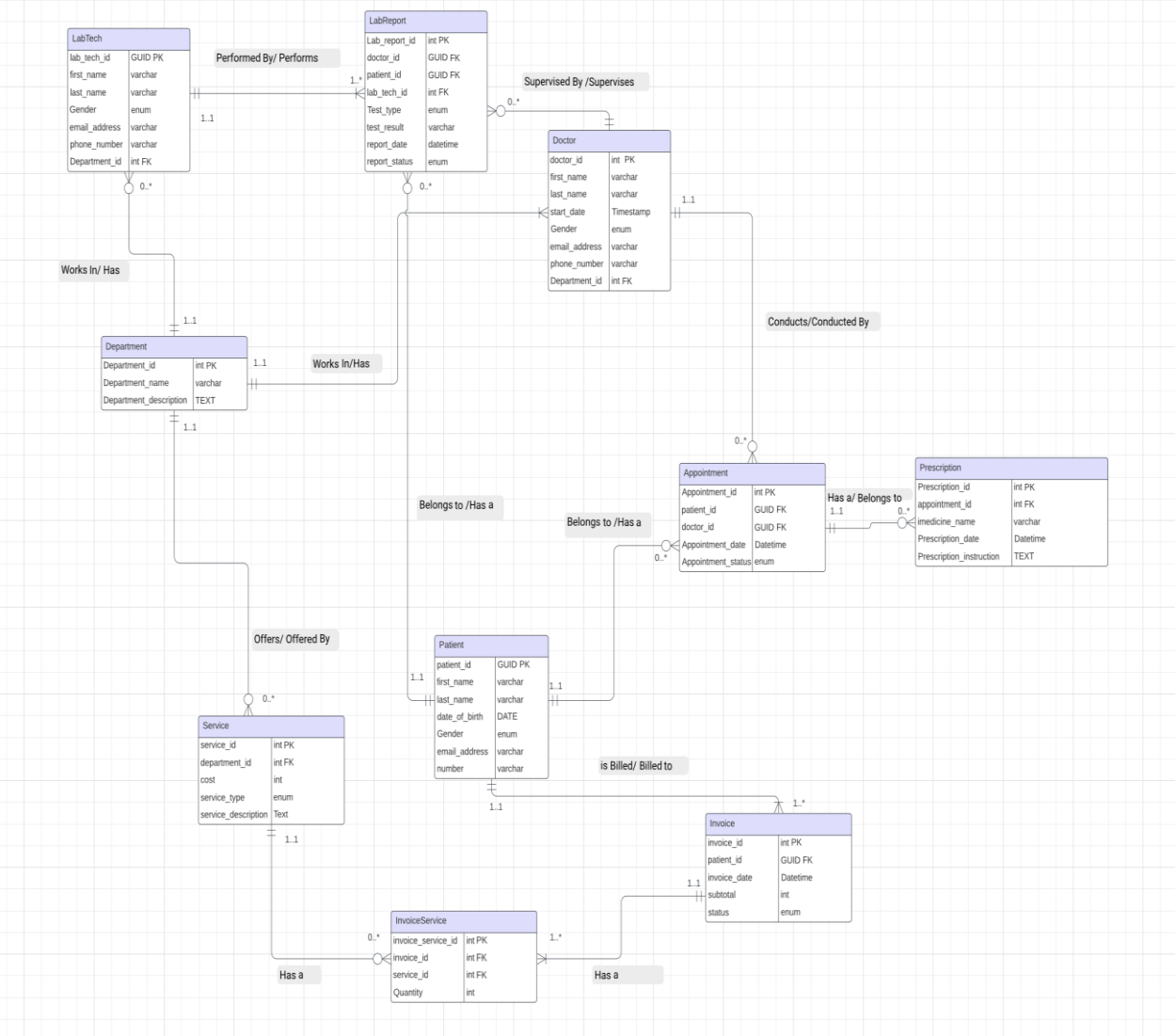
Relationships:

i. Invoice to Patient Relationship:

- Type: One-to-Many
- Cardinality: A Patient can have one or more Invoices and an invoice must be linked to a single patient.

PHASE1_Q3.

https://lucid.app/lucidchart/fe328278-3a42-4905-8650-abeaa40a6a94/edit?invitationId=inv_1a39e65d-e6f9-40dd-8728-8c2d062a1ea1&page=0_0#



PHASE1_Q4_A.

1. DEPARTMENT (Department_id, Department_name, Department_description)

Table 1 Department Table

department_id	department_name	department_description
1	Cardiology	Handles heart related diseases and conditions.
2	Neurology	Deals with disorders of the nervous system.
3	Orthopedics	Focuses on diseases and injuries of the musculoskeletal system.
4	Pediatrics	Provides medical care for infants, children, and teenagers.
5	Dermatology	Concerned with disorders of the skin.

2. DOCTOR (doctor_id, first_name, last_name, start_date, Gender, email_address, phone_number, *Department_id)*

DOCTOR(Department_id) mei DEPARTMENT(Department_id)

Table 2 Doctor

doctor_id	first_name	last_name	start_date	gender	email_address	phone_number	department_id
1	Alice	Smith	2015-02-10	F	alice.smith@email.com	555-0101	1
2	Bob	Jones	2010-03-15	M	bob.jones@email.com	555-0102	2
3	Carol	Johnson	2018-07-25	F	carol.johnson@email.com	555-0103	3
4	David	Lee	2012-12-01	M	david.lee@email.com	555-0104	4
5	Eva	Brown	2020-01-20	F	eva.brown@email.com	555-0105	5

3. PATIENT (patient_id, first_name, last_name, date_of_birth, Gender, email_address, number)

Table 3 Patient

patient_id	first_name	last_name	date_of_birth	gender	email_address	number
1	John	Doe	1990-04-01	M	john.doe@email.com	555-0201
2	Mary	Davis	1982-05-10	F	mary.davis@email.com	555-0202
3	Sam	Wilson	1975-06-15	M	sam.wilson@email.com	555-0203
4	Linda	Moore	1998-07-20	F	linda.moore@email.com	555-0204
5	James	Taylor	2000-08-30	M	james.taylor@email.com	555-0205

4. APPOINTMENT (Appointment_id, *patient_id*, *doctor_id,* Appointment_date, Appointment_status)

    APPOINTMENT (patient_id) mei Patient(patient_id)

    APPOINTMENT (doctor_id) mei Doctor(doctor_id)

Table 4 Appointment

appointment...	patient_id	doctor_id	appointment_date	appointment_status
1	1	1	2023-04-15	Scheduled
2	2	2	2023-04-17	Completed
3	3	3	2023-04-18	Cancelled
4	4	4	2023-04-19	No-show
5	5	5	2023-04-20	Scheduled
6	5	4	2024-04-23	Scheduled

5. LABTECH (lab_tech_id, first_name, last_name, Gender, email_address, phone_number, *Department_id)*

    LABTECH(Department_id) mei DEPARTMENT(Department_id)

Table 5 LabTech

lab_tech_id	first_name	last_name	gender	email_address	phone_number	department_id
1	Liam	Gray	M	liamgray@example.com	555-0300	1
2	Sophia	Frost	F	sophiafrost@example.com	555-0301	2
3	Noah	Stone	M	noahstone@example.com	555-0302	3
4	Emma	River	F	emmariver@example.com	555-0303	4
5	Oliver	Lake	M	oliverlake@example.com	555-0304	5

6. LABREPORT (lab_report_id, *doctor_id*, *patient_id*, *lab_tech_id*, Test_type, test_result, report_date, report_status)

    LABREPORT (patient_id) mei PATIENT (patient_id)

    LABREPORT (doctor_id) mei DOCTOR (doctor_id)

    LABREPORT (lab_tech_id) mei LABTECH (lab_tech_id)

Table 6 LabReport

lab_report_id	doctor_id	patient_id	lab_tech_id	test_type	test_result	report_date	report_status
5	5	5	5	ECG	Irregular	2023-04-16	Complete
4	4	4	4	CT Scan	Normal	2023-04-15	Complete
3	3	3	3	Biopsy	Benign	2023-04-14	Cancelled
2	2	2	2	Urine Test	Abnormal	2023-04-13	Pending
1	1	1	1	Blood Test	Normal	2023-04-12	Complete

7. PRESCRIPTION (Prescription_id, *appointment_id*, medicine_name, Prescription_date, Prescription_instruction)

   PRESCRIPTION (appointment_id) mei APPOINTMENT (appointment_id)

Table 7 Prescription

prescription_id	appointment_id	medicine_name	prescription_date	prescription_instruction
1	1	Amoxicillin	2023-04-12	Take one tablet three times a day
2	2	Ibuprofen	2023-04-13	Take two tablets twice a day
3	3	Acetaminophen	2023-04-14	Take one tablet every 6 hours
4	4	Metformin	2023-04-15	Take one tablet twice a day with meals
5	5	Simvastatin	2023-04-16	Take one tablet in the evening

8. SERVICE (service_id, *department_id*, cost, service_type, service_description)

   SERVICE (department_id) mei DEPARTMENT (department_id)

Table 8 Service

service_id	department_id	cost	service_type	service_description
1	1	100.00	Echocardiogram	Ultrasound of the heart
2	2	200.00	MRI Brain	Magnetic resonance imaging of the brain
3	3	150.00	X-Ray	Radiography used to view the internal form of o…
4	4	120.00	Vaccination	Administration of a vaccine
5	5	130.00	Skin Allergy Test	Test for allergic reactions on the skin

9. INVOICE (invoice_id, *patient_id*, invoice_date, subtotal, status)

   INVOICE (patient_id) mei PATIENT (patient_id)

Table 9 Invoice

invoice_id	patient_id	invoice_date	subtotal	status
1	1	2023-04-12	150.00	Paid
2	2	2023-04-13	200.00	Unpaid
3	3	2023-04-14	250.00	Pending
4	4	2023-04-15	300.00	Paid
5	5	2023-04-16	350.00	Unpaid

10. INVOICESERVICE (invoice_service_id, *invoice_id*, *service_id*, Quantity)

    INVOICESERVICE (invoice_id) mei INVOICE (invoice_id)

    INVOICESERVICE (service_id) mei SERVICE (service_id)

Table 10 InvoiceService

invoice_service_id	invoice_id	service_id	quantity
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	1
5	5	5	2

PHASE1_Q6.

For this report, we chose Lab_Report relation for the purpose of normalization,

LABREPORT (lab_report_id, *doctor_id*, *patient_id*, *lab_tech_id*, Test_type, test_result, report_date, report_status)

Functional Dependencies:

1. Lab_report_id → doctor_id, patient_id, lab_tech_id, Test_type, test_result, report_date, report_status

   This is the only functional dependency that exists for the relation LABREPORT, because Lab_report_id is the only primary determinant for all the other attributes. No other attributes except Lab_report_id is determining any other attributes i.e., there is no interdependences between the other attributes.

1NF:

There are no repeating groups, and all the values are atomic, so the relation is in 1NF.

2NF:

The relation meets the second normal form because it does not have a composite primary key, which means no subset of the primary key determines other attributes. There are no partial dependencies, as shown by the functional dependencies and it follows the first normal form as well, therefore, it is in the second normal form.

3NF:

There are no transitive dependencies i.e., there are no non-key attributes determining other non-key attributes and the relation is in 2NF, therefore the relation is in 3NF.

PHASE 2_Q1.

I. Relational Algebra:

1. Description: Finding all doctors who have appointments scheduled today.

$$\Pi_{first\_name,last\_name,appointment\_date}(\sigma_{appointment\_status="Scheduled" \wedge appointment\_date=today}(Appointment) \bowtie_{Doctor.doctor\_id=Appointment.doctor\_id} Doctor)$$

```
SELECT DISTINCT first_name, last_name, appointment_date  FROM
DOCTOR JOIN APPOINTMENT
ON Doctor.doctor_id=Appointment.doctor_id  WHERE
Appointment.appointment_date = CURRENT_DATE AND
Appointment.appointment_status = 'Scheduled';
```

2. Description: Getting a Prescription Report for Completed Appointments of patients.

$$\Pi_{first\_name,\ last\_name,\ medicine\_name,\ prescription\_date}(Patient \bowtie_{patient.patient\_id=appointment.patient\_id} (\sigma_{appointment\_status="completed"}(Appointment) \bowtie_{Appointment.appointment\_id=Prescription.appointment\_id} Prescription)$$

```
SELECT  DISTINCT P.first_name, P.last_name, Pr.prescription_date,
Pr.medicine_name FROM Patient P JOIN Appointment A ON P.patient_id
= A.patient_id          JOIN Prescription Pr ON A.appointment_id =
Pr.appointment_id    WHERE A.appointment_status = 'Completed';
```

II. SQL Queries using SQL clauses:

1. Selecting all doctors who have scheduled appointments, for this purpose we use SELECT, WHERE EXISTS clauses:

```
SELECT * FROM Doctor WHERE EXISTS ( SELECT * FROM
Appointment WHERE Doctor.doctor_id = Appointment.doctor_id
AND appointment_status = 'Scheduled' );
```

2. For selecting all doctors where their email address is not null, and their last name starts with 'S' we made use of IS NULL and LIKE clauses.

```
SELECT * FROM Doctor WHERE email_address IS NOT NULL
AND last_name LIKE 'S%';
```

3. For Deleting appointments marked as 'Cancelled' or 'No-show' we used the clauses IN and DELETE,

```
DELETE FROM Appointment WHERE appointment_status IN
('Cancelled', 'No-show') ;
```

4. For Updating the status of unpaid invoices to 'Pending',

```
UPDATE Invoice
SET status = 'Pending' WHERE status = 'Unpaid';
```

5. For Finding patients with appointments with specific doctors

```
SELECT * FROM Patient WHERE patient_id IN
(SELECT patient_id FROM Appointment WHERE doctor_id
IN (1, 2, 3));
```

6. Retrieving patients with either scheduled appointments or unpaid invoices using UNION clause,

```
SELECT patient_id, first_name, last_name FROM Patient
WHERE patient_id IN ( SELECT patient_id FROM Appointment
WHERE appointment_status = 'Scheduled')
UNION
SELECT patient_id, first_name, last_name FROM Patient
WHERE patient_id IN ( SELECT patient_id FROM Invoice
WHERE status = 'Unpaid' );
```

III. SQL Functions

1. Using LOWER () to find doctors with email addresses containing 'gmail' in case insensitive manner,

```
SELECT *

FROM Doctor

WHERE LOWER (email_address) LIKE '%gmail%';
```

2. Finding the length of last names of patients using LENGTH(),

```
SELECT last_name, LENGTH (last_name) AS name_length
FROM Patient;
```

3. Using CONCAT () to combine first name and last name of doctors,

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name

FROM Doctor;
```

4. Finding the minimum and maximum appointment dates using MIN() and MAX() functions ,

```
SELECT MIN (appointment_date) AS min_date,
MAX (appointment_date) AS max_date
FROM Appointment;
```

5. Calculating the total cost of services for each invoice using SUM() and GROUP BY

```
SELECT InS.invoice_id, SUM(S.cost * InS.quantity) AS total_cost

FROM InvoiceService InS

JOIN Service S ON InS.service_id = S.service_id

GROUP BY InS.invoice_id;
```

6. Counting the number of appointments per doctor and it sorts the results in descending order by the appointment_count, so the doctor with the most appointments appears first in the list. For this we use COUNT and ORDER BY

```
SELECT doctor_id, COUNT (*) AS appointment_count

FROM Appointment

GROUP BY doctor_id

ORDER BY appointment_count DESC;
```

7. Finding the average cost of services for each department, only including departments with an average cost greater than 100 using AVG() function and HAVING clause,

```
SELECT department_id, AVG(cost) AS avg_cost
FROM Service GROUP BY department_id
HAVING avg_cost > 100;
```

Conclusion:

In conclusion this current hospital database management system is in the introductory step as it was developed in a limited time . However it also represents a strong foundation which is required for an advanced and comprehensive solution . With further investment and development its capabilities can be significantly expanded with user friendly interfaces and real time analytics. Along with this we can so integrate it with the other streamline health care systems . By providing more time to it we can make the current work even better by ensuring several more facilities like scalability, reliability, and security . Such an approach which is incremental allows us to build a solid basis by addressing several challenges and incorporate feedback from the users . Thus, through continued dedication and investment, we can transform this basic system into an innovative tool that revolutionizes healthcare management, ultimately improving patient care and operational efficiency.