HOME ABOUT CONTACT

HOSPITAL PATIENT MANAGER

PRESENTATION

INTRODUCE MEMBERS

NGUYEN TUAN ANH
TRAN TUAN ANH
VU QUOC HUY
TRAN MINH HOANG

ABOUT THE TOPIC

In the modern context, patient data management in hospitals faces several challenges regarding speed, accuracy, and security. The business problem our project addresses is the need to build an effective Database System capable of organizing, storing, and retrieving crucial information such as patient records, appointment schedules, medical history, invoices, and personnel data (doctors, staff). Our goal is to make the hospital management process more systematic, minimize errors, and support better decision-making in healthcare operations.

PART I: DATABASE ANALYSIS & DESIGN

CREATE TABLE

NORMALIZE TO 3NF

ERD DIAGRAM

CREATE TABLE

A hospital needs to develop a system to manage all data related to:

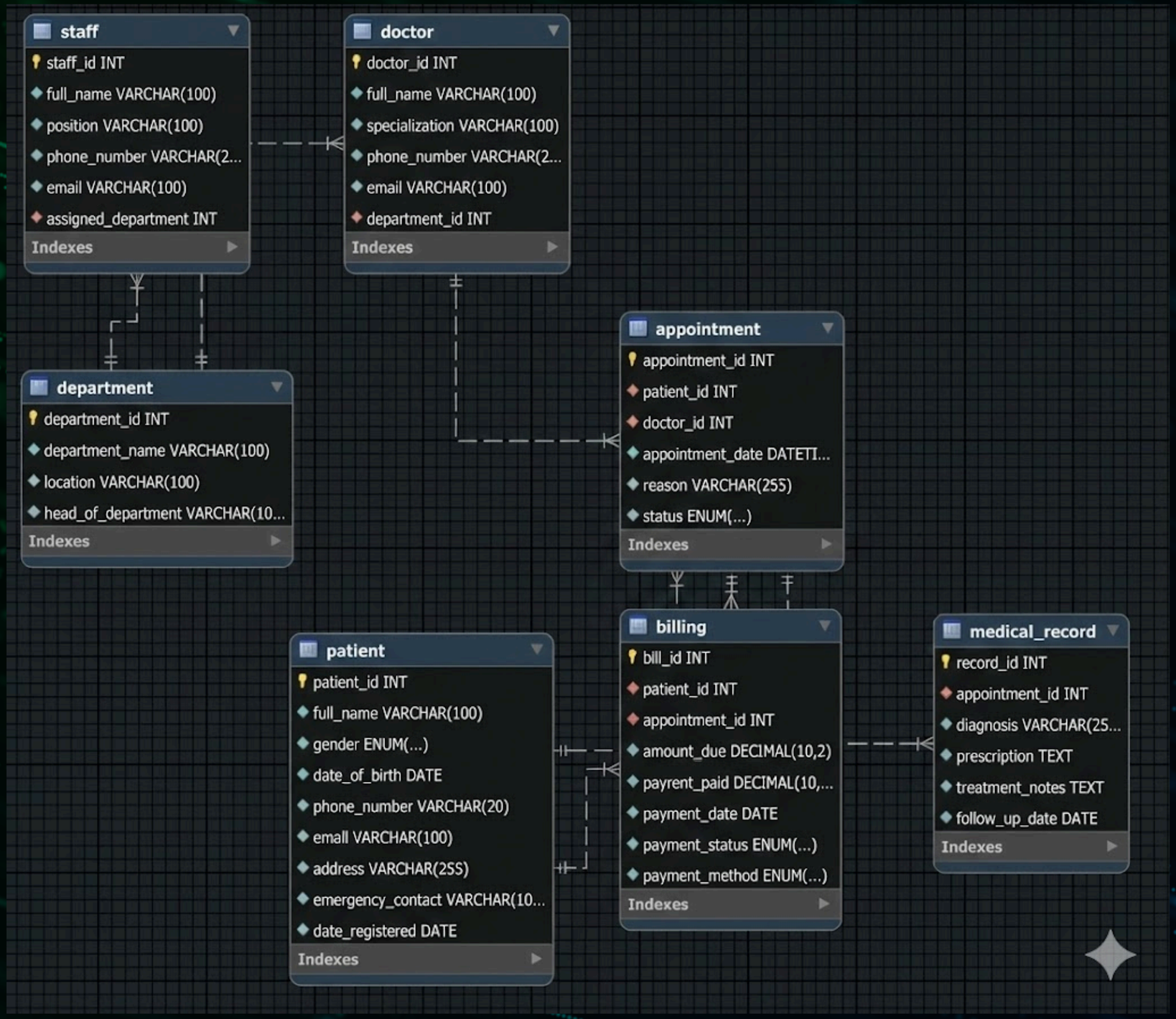
- Patients
- Doctors
- Departments
- Appointments
- Medical Records
- Billing
- Administrative Staff

The system must ensure that the data is accurate, consistent, and provides strong support for operations, reporting, and management.

Tables_in_hospital_manager
appointment
billing
department
doctor
medical_record
patient
staff

ERD DIAGRAM

The diagram represents the ERD of a hospital management system, including key entities such as Patient, Doctor, Department, Appointment, Medical Record, Billing, and Staff. The relationships reflect real hospital workflows—from patient management and appointment scheduling to medical records, billing, and administrative operations. The model ensures accurate, consistent data and supports efficient operations and reporting.



NORMALIZE TO 3NF

From the first table these is step that normalize to 3NF:

- Disable foreign key checks to allow creating new tables.
- Create the _new tables following 3NF rules, including normalizing Billing and separating the Payment table.
- Transfer data from the old tables to the corresponding _new tables.
- Create a trigger to automatically update payment status based on the Payment table.
- Rename the old tables to *_old for backup.
- Rename the _new tables to become the official active tables.
- Re-enable foreign key checks to restore data constraints.

Tables_in_hospital_manager
billing_old
department
department_old
doctor
doctor_old
medical_record
medical_record_old
patient
patient_old
payment
staff
staff_old

PART II:MYSQL IMPLEMENTATION

ABOUT CONTACT

VIEWS

TRIGGER

Stored Procedure

VIEWS

To illustrate how the views operate within the system, this section presents a single representative view: vw_department_stats, which provides statistical summaries of revenue and activity by department. This view helps simplify complex queries and allows users to easily observe aggregated information such as average cost, number of doctors, and total appointments for each department.

```sql
-- 1.2. View doanh thu theo khoa
DROP VIEW IF EXISTS v_department_revenue;

CREATE VIEW v_department_revenue AS
SELECT
    dept.department_id,
    dept.department_name,
    COUNT(DISTINCT a.appointment_id)      AS total_appointments,
    SUM(b.amount_due)                     AS total_amount_due,
    SUM(b.amount_paid)                    AS total_amount_paid,
    SUM(b.amount_due - b.amount_paid)     AS total_outstanding
FROM Department dept
JOIN Doctor d      ON d.department_id = dept.department_id
JOIN Appointment a ON a.doctor_id     = d.doctor_id
JOIN Billing b     ON b.appointment_id = a.appointment_id
GROUP BY
    dept.department_id,
    dept.department_name;
```

	department...	department_na...	total_appointme...	total_amount_d...	total_amount_p...	total_outstandi...
	1	Cardiology	6	1750000.00	1700000.00	50000.00
	2	Neurology	5	1150000.00	550000.00	600000.00
	3	Pediatrics	6	1000000.00	1000000.00	0.00
	4	Orthopedics	7	1900000.00	1500000.00	400000.00
	5	Dermatology	6	1260000.00	1190000.00	70000.00

STORED PROCEDURE

To demonstrate the use of stored procedures in the system, this section presents two representative procedures that handle essential hospital operations. The first procedure, sp_create_appointment, automates the creation of new appointments with all required parameters. The second procedure, sp_monthly_revenue_report, generates a monthly revenue summary by department, enabling efficient financial reporting and analysis

```
DELIMITER //
CREATE PROCEDURE sp_create_appointment(
    IN p_patient_id INT,
    IN p_doctor_id INT,
    IN p_appointment_date DATETIME,
    IN p_reason VARCHAR(255),
    OUT p_new_appointment_id INT
)
BEGIN
    INSERT INTO Appointment (patient_id, doctor_id, appointment_date, reason, status)
    VALUES (p_patient_id, p_doctor_id, p_appointment_date, p_reason, 'Scheduled');

    SET p_new_appointment_id = LAST_INSERT_ID();
END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE sp_monthly_revenue_report(
    IN p_year INT,
    IN p_month INT
)
BEGIN
    SELECT
        dept.department_name,
        COUNT(b.bill_id) AS total_bills,
        SUM(b.amount_paid) AS total_paid,
        SUM(b.amount_due - b.amount_paid) AS total_unpaid
    FROM Billing b
    JOIN Appointment a ON b.appointment_id = a.appointment_id
    JOIN Doctor d ON a.doctor_id = d.doctor_id
    JOIN Department dept ON d.department_id = dept.department_id
    WHERE YEAR(b.payment_date) = p_year
      AND MONTH(b.payment_date) = p_month
    GROUP BY dept.department_id;
END //
DELIMITER ;
SHOW PROCEDURE STATUS WHERE Db = 'hospital_manager';
```

Db	Name	Type	Definer	Modified	Created	Security_type	Comment	character_set_client	collation_connection	Database Collation
hospital_manager	sp_create_appointment	PROCEDURE	root@localhost	2025-11-29 12:13:10	2025-11-29 12:13:10	DEFINER		utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
hospital_manager	sp_monthly_revenue_report	PROCEDURE	root@localhost	2025-11-29 12:13:10	2025-11-29 12:13:10	DEFINER		utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

TRIGGER

To ensure data integrity and automate system updates, this section introduces a trigger that manages billing status changes. The trigger trg_update_bill_status automatically adjusts the payment status and payment date whenever a billing record is updated, helping maintain consistent and accurate financial information across the system.

```
DELIMITER //
CREATE TRIGGER trg_update_bill_status
BEFORE UPDATE ON Billing
FOR EACH ROW
BEGIN
    IF NEW.amount_paid >= NEW.amount_due THEN
        SET NEW.payment_status = 'Paid';
        SET NEW.payment_date = CURDATE();
    ELSEIF NEW.amount_paid > 0 THEN
        SET NEW.payment_status = 'Partially Paid';
    ELSE
        SET NEW.payment_status = 'Unpaid';
    END IF;
END //
DELIMITER ;

SHOW TRIGGERS;
```

Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client	collation_connection	Database Collation
trg_update_bill_status	UPDATE	billing	BEGIN IF NEW.amount_paid >= NEW.amoun...	BEFORE	2025-11-29 12:18:41.76	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

SALFORD & CO.

PART III: PYTHON MYSQL INTEGRATION

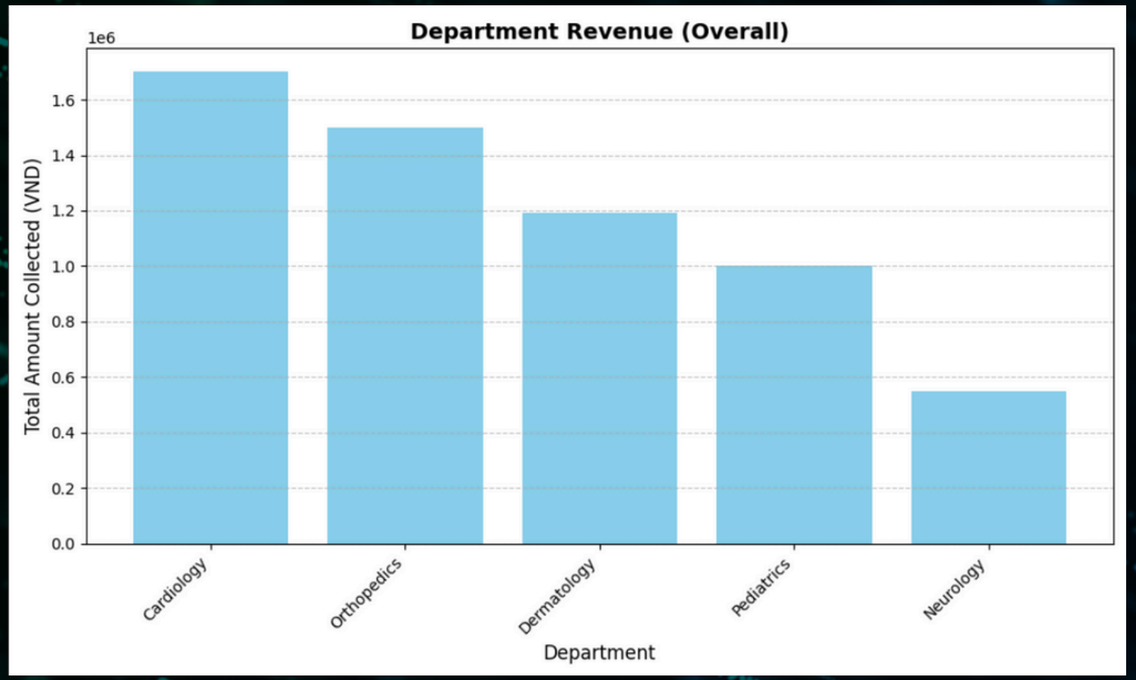
ABOUT CONTACT

REVENUE BY DEPARTMENT

MONTHLY APPOINTMENT

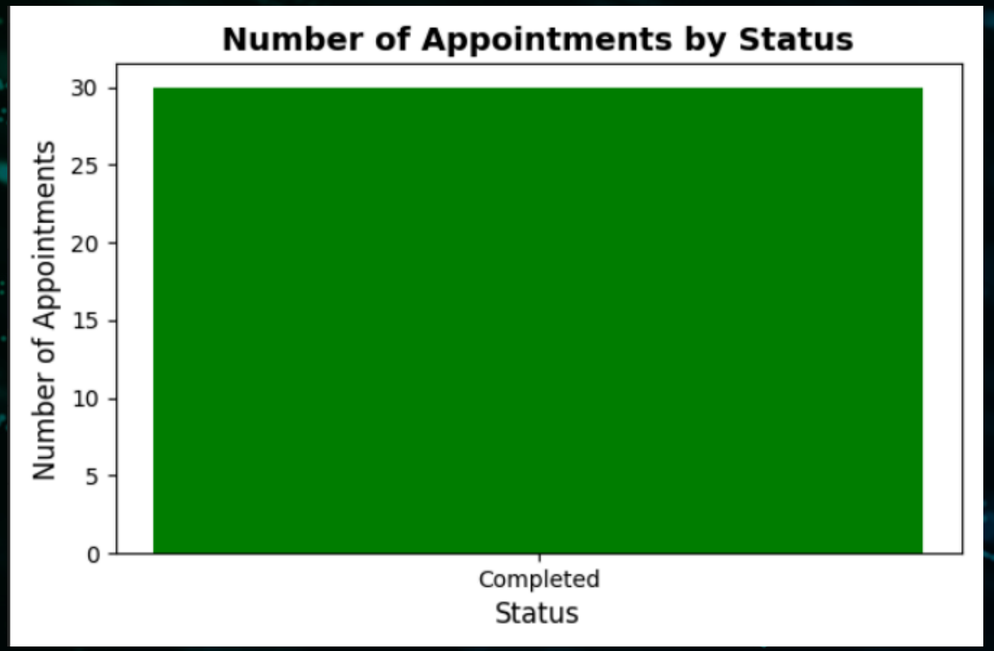
REVENUE BY DEPARTMENT

```python
plot_department_revenue(df: pd.DataFrame, title: str = "Department Revenue"):
"""
Draws a bar chart: revenue collected by department.
"""
if df.empty:
    print(f"Empty DataFrame ({title}), no data to plot.")
    return

plt.figure(figsize=(10, 6))
# Sorts by revenue in descending order before plotting
df_sorted = df.sort_values(by="total_amount_paid", ascending=False)

plt.bar(df_sorted["department_name"], df_sorted["total_amount_paid"], color='skyblue')
plt.xlabel("Department", fontsize=12)
plt.ylabel("Total Amount Collected (VND)", fontsize=12)
plt.title(title, fontsize=14, fontweight='bold')
plt.xticks(rotation=45, ha="right")
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



MONTHLY APPOINTMENT

```python
def plot_appointment_status(df: pd.DataFrame, title: str = "Number of Appointments by Status"):
    """
    Draws a bar chart: number of appointments by status.
    """
    if df.empty:
        print(f"Empty DataFrame ({title}), no data to plot.")
        return

    plt.figure(figsize=(6, 4))
    plt.bar(df["status"], df["total_appointments"], color=['green', 'red', 'gray'])
    plt.xlabel("Status", fontsize=12)
    plt.ylabel("Number of Appointments", fontsize=12)
    plt.title(title, fontsize=14, fontweight='bold')
    plt.tight_layout()
    plt.show()
```



THANK YOU