DATABASE FOR MANAGING PATIENTS AND APPOINTMENTS IN A HOSPITAL

SCHEDULES DOCTOR APPOINTMENTS, TRACKS TREATMENT HISTORY, AND MANAGES BILLING AND PRESCRIPTIONS

404: Team Not Found - Riza Bakyt, Kalykbergen Aiganym, Kakimbek Anel

INTRODUCTION

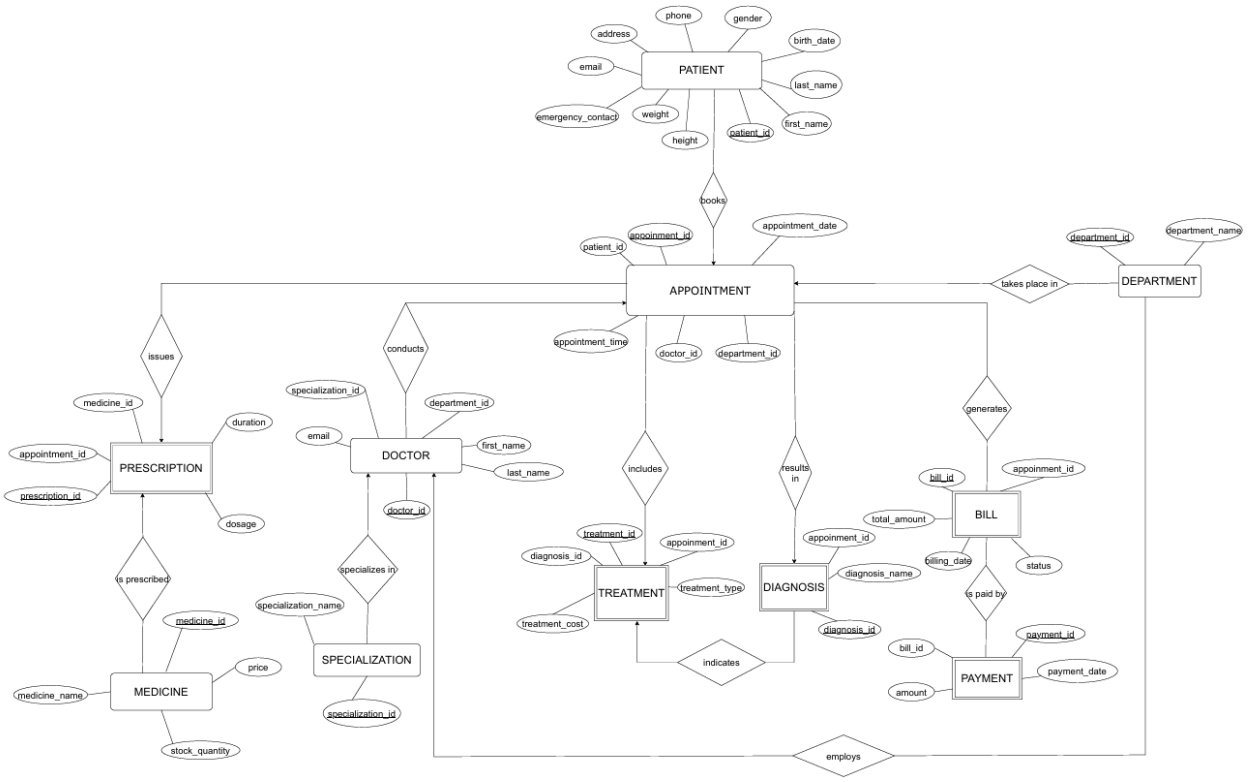
1. Project Overview

The goal of this project was to design a **Hospital Management Database** that can:

- Track patients, their appointments, diagnoses, treatments, and prescriptions
- Manage doctors, their specializations, and departments
- Handle billing and payments for appointments

2. Objectives:

- Demonstrate understanding of **relational database concepts**, including normalization
- Implement **advanced SQL queries, functions, procedures, triggers, collections, cursors, records, exceptions, and packages**
- Show **real-life logic** in database design
- Ensure the system can handle multiple appointments, treatments, and prescriptions efficiently.



https://drive.google.com/file/d/1bdkl1IEucapyZTp_uMJaMLLS25bZfwJ0/view?usp=drive_link

1.System Design and Architecture

1.1. About our dataset

This dataset represents a historical medical information system covering the years 2023–2025. The core structure is built around autonomous entities: patients, doctors, specializations, and departments. At the center is the appointment, from which all dependent records originate: diagnoses, treatments, prescriptions, the bill, and the payment. Diagnoses and prescriptions capture clinical decisions, while treatments may be linked to a diagnosis or performed directly during the appointment.
The financial model is streamlined: each appointment generates exactly one bill, which is settled by a single payment.

The historical nature of the data ensures full traceability of clinical and financial events throughout 2023–2025.

1.2. Entity Descriptions

PATIENT: stores demographic and contact details.
patient_id (PK)
first/last_name
birth_date, gender
phone, email, address
weight, height
emergency_contact

APPOINTMENT: patient–doctor interactions
appointment_id (PK)
patient_id (FK)
doctor_id (FK)
department_id (FK)
appointment_date
appointment_time

DEPARTMENT : hospital units
department_id (PK)
department_name

SPECIALIZATION: medical specialization
specialization_id (PK)
specialization_name

DOCTOR: Information about medical personnel.
doctor_id (PK)
first/last_name
email
specialization_id (FK)
department_id (FK)

TREATMENT: procedures based on a diagnosis.
treatment_id (PK)
diagnosis_id (FK)
appointment_id (FK)
treatment_type
treatment_cost

DIAGNOSIS: diagnostic outcomes
diagnosis_id (PK)
appointment_id (FK)
diagnosis_name

MEDICINE: medicine catalog
medicine_id (PK)
medicine_name
price
stock_quantity

PRESCRIPTION: links medicines to appointments
prescription_id (PK)
appointment_id (FK)
medicine_id (FK)
dosage
duration

BILL:financial charges
bill_id (PK)
appointment_id (FK)
total_amount
billing_date
status

PAYMENT: records monetary transactions
payment_id (PK)
bill_id (FK)
amount
payment_date

1.3.Relationship Summary

Relationship	Cardinality	Description
Patient → Appointment	1:M	One patient can have many appointments
Appointment → Doctor	M:1	One doctor can conduct many appointment
Appointment → Department	M:1	One department can host many appointments
Appointment → Diagnosis	1:M	One appointment may result in multiple diagnoses
Appointment → Treatment	1:M	Treatment is based on Appointment/Diagnosis
Appointment → Prescription	1:M	Prescription is issued during Appointment
Medicine → Prescription	1:M	One medicine can appear in many prescriptions
Doctor → Specialization	M:1	Doctor has Specialization
Bill → Appointment	1:1	Bill is generated for Appointment
Payment → Bill	1:1	Each bill has one corresponding payment

2.Queries:
2.1. Functions and Procedures /Cursor and records/Packages and Exceptions

PACKAGE : pkg_patient_analytics

The package **pkg_patient_analytics** provides analytical tools for working with patient medical activity in the hospital database. It centralizes procedures and a function that generates summaries, history reports, and doctor-visit analytics for patients.

1.Procedure: proc_patient_history

Retrieves full medical history for a given patient, including appointment dates, diagnoses, and doctor information. Uses a cursor to print each record in chronological order.

2. Procedure: proc_frequent_patients

Identifies patients with five or more appointments and calculates their visit statistics, including total visit count and average interval between visits. Uses a cursor and aggregation functions.

3. Function: fn_patient_most_frequent_doctor

Returns a formatted text report showing the doctor most frequently visited by a specific patient. Includes patient name, doctor name, specialization, department, and number of visits.

PACKAGE : pkg_patient_risk

The package **pkg_patient_risk** provides a set of analytical functions designed to evaluate a patient's health risk based on demographic factors, physical indicators, diagnosis history, and treatment activity. It centralizes risk-calculation logic and returns both numerical scores and a formatted summary report.

1. Function: f_age_risk

Calculates the risk contribution associated with the patient's age by determining age in years and assigning a score according to predefined thresholds.

2. Function: f_bmi_risk

Computes BMI from height and weight and returns a risk score based on whether the BMI indicates normal, overweight, or obese status.

3. Function: f_diag_risk:

Counts the number of diagnoses recorded across all patient appointments and assigns a risk value reflecting clinical complexity.

4. Function: f_treat_risk:

Evaluates how many treatments the patient has undergone and frequent treatments increase the risk level.

5. Function: f_patient_risk_score(Function: f_pr_sum):

These two functions work together to evaluate a patient's health risk: f_patient_risk_score calculates the total numerical score, while f_pr_sum presents this result as a formatted summary with patient details, component scores, and the final risk category.

PACKAGE: pkg_treatment_analytics

The package **pkg_treatment_analytics** is created to organize all treatment-related analytics in one place. It helps to monitor medicine stocks, identify common diagnoses, analyze treatment trends, and summarize prescription usage. By keeping these procedures together, the package makes the system easier to manage and more convenient for generating medical reports.

1.Procedure: pr_low_stock_medicines(p_threshold)

Identifies medicines with stock levels below the given threshold.Provides a list sorted by quantity and highlights critical shortages.

2.Procedure: pr_top_diagnoses(p_limit)

Displays the most common diagnoses based on the number of unique patientsUseful for identifying medical trends and high-occurrence conditions.

3. Procedure: proc_treatment_trends(p_days)

Analyzes treatment usage trends over the last *p_days*. Shows total assignments and unique patient counts for each treatment type.

4. Procedure: proc_prescription_summary(p_medicine_id)

Generates a summary for a specific medicine: total number of prescriptions, average treatment duration. Helps monitor medicine usage intensity.

5. Procedure: proc_frequent_treatments(p_limit)

Lists the most frequently assigned treatments. Ranks them by total usage and number of unique patients.

Procedure: proc_upcoming_appointments

This procedure displays all upcoming appointments for a given doctor. Using a cursor, it retrieves future appointment dates, times, and patient names, printing each entry in chronological order. If no upcoming visits are found, it outputs a corresponding message. Basic exception handling ensures stable execution and clear error reporting.

Procedure: proc_doctor_daily_report

This procedure produces a daily operational summary for a selected doctor. It retrieves the doctor's identifying information and specialization, then calculates key activity indicators for the specified date, including the total number of appointments, unique patients, treatments performed, and prescriptions issued. By aggregating data from related clinical tables, the procedure presents a concise overview of the doctor's workload for that day. Exception handling ensures reliable execution in cases of missing data or unexpected errors.

2.2.Collections

PACKAGE : price_update_pkg (Collection-Based Package)

The package **price_update_pkg** implements bulk price updates using PL/SQL collections. It uses BULK COLLECT to retrieve data, FORALL to apply updates efficiently, and collection-based storage to generate a detailed before-and-after price summary.

The package performs the following tasks:
- Loads medicine names and prices into collections.
- Applies percentage-based updates through bulk operations.
- Produces a formatted report showing old and new prices.

1. Function: increase_prices

This function increases all medicine prices using a collection-driven approach.
It first validates the input percentage, then retrieves medicine data into collections with BULK COLLECT. Using FORALL, it performs a high-performance bulk update, reloads the updated prices into another collection, and returns a formatted summary comparing old and new values.

2.3. Triggers

Trigger: trg_presc_dos

This trigger ensures data quality by preventing insertion of prescriptions with an empty dosage value. It validates the dosage field before a new record is added to the **prescriptions** table. If the dosage is missing, the trigger raises a clear custom error message and stops the insertion.

Trigger: trg_bill_default_values

This trigger automatically assigns default values when a new bill is inserted. If the billing date is missing, it is set to the current system date. If the status is not provided, it is set to 'UNPAID'. This ensures consistent and complete billing data without requiring manual input.

Trigger Name: trg_bill_auto_payment

Automatically creates a payment record whenever a bill's status changes to 'PAID'. It fires after an update on the bills table and inserts a new row into payments with:payment_id generated by payments_seq, bill_id from the updated bill, payment_date set to current date, amount from the bill.