Hospital Patient Management System

Course Project – Advanced Database Systems

---

1. Project Overview

This project implements a **Hospital Patient Management System** using **Oracle SQL and PL/SQL**. The goal is to automate core hospital operations such as patient admission, appointment scheduling, treatment billing, room management, discharge processing, and reporting, while ensuring **data integrity, security, and concurrency control**.

The system demonstrates the use of: - Relational database design - User management and privileges - PL/SQL procedures, functions, triggers, and cursors - Transaction management (COMMIT / ROLLBACK) - Audit logging - Concurrency control and locking

---

2. Database Users and Roles

Three database users were used to enforce security and separation of responsibilities:

User	Purpose
SYS	Initial setup, system privileges, and monitoring blocking sessions
HOSP_MANAGER	Creates application users and grants privileges
HOSP_USER1	Owns all tables, sequences, procedures, functions, and triggers
HOSP_USER2	Limited user for inserting and querying data

This structure follows real-world Oracle security best practices.

---

3. Database Schema Design

Core Tables

- **Patients** (id, name, date_of_birth, status, total_bill, room_id)
- **Doctors** (id, name, specialty, available_hours)
- **Appointments** (id, patient_id, doctor_id, appointment_date, status)
- **Treatments** (id, patient_id, doctor_id, treatment_description, cost)
- **Rooms** (id, room_type, capacity, availability)
- **Warnings** (id, patient_id, warning_reason, warning_date)

- **AuditTrail** (id, table_name, operation, old_data, new_data, action_date)

Primary keys, foreign keys, and constraints were applied to maintain referential integrity.

---

4. Sequences

Sequences were created for all tables with primary keys to ensure unique identifier generation:

- seq_patients
- seq_doctors
- seq_appointments
- seq_treatments
- seq_rooms
- seq_warnings
- seq_audit

Sequences were synchronized with table data during testing to avoid primary key conflicts.

---

5. Implemented Tasks

Task 1 – User Management and Privileges

- Created users and granted appropriate privileges
- Ensured limited access for non-owner users

Task 2 – Patient Admission Validation

- Trigger validates room availability before admitting a patient
- Prevents admission if no rooms are available
- Updates room availability and logs actions in AuditTrail

Task 3 – Appointment Scheduling

- PL/SQL procedure checks doctor availability
- Schedules appointments and updates available hours

Task 4 – Treatment Cost Calculation

- Function calculates total treatment cost per patient
- Automatically updates patient total_bill

Task 5 – Room Assignment Tracking

- Trigger automatically assigns rooms based on availability
- Updates room availability and audit records

Task 6 – Discharge Processing

- Procedure marks patient as discharged
- Frees room availability
- Logs discharge in AuditTrail

Task 7 – Hospital Performance Report

- Cursor-based report showing:
- Total admissions and discharges
- Average patient stay duration
- Top doctors by treatments handled

Task 8 – Multi-Appointment Cancellation

- Transaction-safe cancellation of multiple appointments
- Uses ROLLBACK to ensure consistency if any cancellation fails

Task 9 – Patient Warnings and Status Update

- Procedure issues warnings for missed appointments or delayed payments
- Automatically flags patients after three warnings

Task 10 – Advanced Data Manipulation

- Function: get_doctor_patient_count(doctor_id)
- Procedure: update_patient_status_by_bill(threshold)
- Updates patient status to "High-Value" when applicable

---

6. Bonus Tasks

Task 11 – Blocker–Waiting Situation

- Demonstrated row-level locking using two concurrent sessions
- One session updates a room without committing (BLOCKER)
- Another session attempts to update the same row and enters a waiting state

Task 12 – Identifying Blocker and Waiting Sessions

- Used the `V$SESSION` view (as SYS)
- Identified blocking and waiting sessions using:
- SID
- SERIAL#
- BLOCKING_SESSION
- Event: `enq: TX - row lock contention`

---

7. Testing Strategy

Each task was tested using controlled inputs: - Valid and invalid cases (positive and negative testing) - Verification using SELECT queries - Trigger behavior validated through automatic data changes - Error handling verified using RAISE_APPLICATION_ERROR

Concurrency was tested using multiple sessions to simulate real-world usage.

---

8. Transaction Management

- DML operations (INSERT, UPDATE, DELETE) were controlled using COMMIT and ROLLBACK
- No COMMIT statements were placed inside triggers or core procedures
- Transactions are finalized by the caller to ensure atomicity and consistency

---

9. Conclusion

This project successfully demonstrates a complete **Oracle-based hospital management system** using advanced database concepts. It follows professional standards for:

- Security and privilege management
- Data integrity
- Automation using PL/SQL
- Auditing and logging
- Concurrency control

The system is robust, scalable, and suitable as a foundation for a real-world hospital information system.

End of Report