# **PHASE III: Hotel Management System**

# **Logical Model Design**

# **Prepared by:** NGENZI Blaise **ID:** 27488 **Course:** Database Development with PL/SQL (INSY 8311)

**Lecturer:** Eric Maniraguha

# Diagram.drawio

## 1. Introduction

This document shows a clear and simple design for a Hotel Management System database. The design is based on the  **ER diagram**. This design is also the starting point for building a strong database to help manage hotel operations.

## 2. Project Overview

The Hotel Management System aims to streamline hotel operations by efficiently managing rooms, guests, staff, bookings, and payments. The system will enable hotel staff to track room availability, process bookings, manage guest information, and handle payments through a centralized database.

## 3. Entity Relationship Analysis

### 3.1 Core Entities

The logical model consists of five primary entities:

1. **Room**: Represents the hotel's accommodation units.
2. **Guest**: Contains information about hotel patrons.
3. **Staff**: Stores details about hotel employees.
4. **Booking**: Manages reservation information.
5. **Payment**: Tracks financial transactions related to bookings.

**3.2 Relationships**

The relationships between entities are defined as follows:

**Room to Booking**: One-to-many relationship

* One room can have multiple bookings over time
* Each booking is associated with exactly one room

**Guest to Booking**: One-to-many relationship

* One guest can make multiple bookings
* Each booking is associated with exactly one guest

**Staff to Booking**: One-to-many relationship

* One staff member can process multiple bookings
* Each booking is handled by exactly one staff member

**Booking to Payment**: One-to-one relationship

* Each booking has exactly one payment record
* Each payment is associated with exactly one booking

## Detailed Entity Definitions

### 4.1 Room Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Constraints** | **Description** |
| room\_id | INT | PRIMARY KEY | Unique identifier for each room |
| room\_number | VARCHAR2(10) | UNIQUE | Room number displayed to guests and staff |
| room\_type | VARCHAR2(30) | NOT NULL | Category of room (e.g., Standard, Deluxe, Suite) |
| price\_per\_night | DECIMAL2(10,2) | NOT NULL | Cost per night for the room |
| availability\_status | VARCHAR2(15) | DEFAULT 'Available', CHECK('Available','Occupied') | Current status of the room |

**Business Rules:**

* Room numbers must be unique within the hotel
* Room availability must be either 'Available' or 'Occupied'
* Default status for new rooms is 'Available'

### 4.2 Guest Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Constraints** | **Description** |
| guest\_id | INT | PRIMARY KEY | Unique identifier for each guest |
| first\_name | VARCHAR(50) | NOT NULL | Guest's first name |
| last\_name | VARCHAR(50) | NOT NULL | Guest's last name |
| email | VARCHAR(100) | UNIQUE | Guest's email address for correspondence |

**Business Rules:**

* Each guest must have a unique email address
* Guest contact information (email or phone) must be provided

### 4.3 Staff Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Constraints** | **Description** |
| staff\_id | INT | PRIMARY KEY | Unique identifier for each staff member |
| name | VARCHAR(100) | NOT NULL | Staff member's full name |
| position | VARCHAR(50) | NOT NULL | Job title or role within the hotel |
| phone | VARCHAR(15) |  | Staff contact number |
| email | VARCHAR(100) |  | Staff email address |

**Business Rules:**

* Each staff member must have a defined position
* Contact information must be provided for communication purposes

### 4.4 Booking Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Constraints** | **Description** |
| booking\_id | INT | PRIMARY KEY | Unique identifier for each booking |
| guest\_id | INT | FOREIGN KEY references Guest(guest\_id) | Links to the guest making the booking |
| room\_id | INT | FOREIGN KEY references Room(room\_id) | Links to the room being booked |
| staff\_id | INT | FOREIGN KEY references Staff(staff\_id) | Links to staff member processing the booking |
| check\_in\_date | DATE | NOT NULL | Scheduled arrival date |
| check\_out\_date | DATE | NOT NULL | Scheduled departure date |
| booking\_date | DATE | NOT NULL | Date when booking was created |

**Business Rules:**

* Check-out date must be after check-in date
* Each booking is associated with exactly one guest, one room, and one staff member
* Booking date should be the current date when the booking is created

### 4.5 Payment Entity

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Constraints** | **Constraints** |
| payment\_id | INT | PRIMARY KEY | PRIMARY KEY |
| booking\_id | INT | FOREIGN KEY references Booking(booking\_id), UNIQUE | FOREIGN KEY references Booking(booking\_id), UNIQUE |
| payment\_date | DATE | NOT NULL | NOT NULL |
| amount\_paid | DECIMAL(10,2) | NOT NULL, CHECK(amount\_paid > 0) | NOT NULL, CHECK(amount\_paid > 0) |
| payment\_method | VARCHAR(30) | NOT NULL | NOT NULL |

**Business Rules:**

* Each payment is linked to exactly one booking
* Payment amount must be positive
* Payment method must be specified

## 5. Normalization Analysis

The logical model adheres to the principles of normalization through the Third Normal Form (3NF):

### 5.1 First Normal Form (1NF)

* All entities have a primary key
* All attributes contain atomic values
* No repeating groups or arrays

### 5.2 Second Normal Form (2NF)

* All entities satisfy 1NF
* All non-key attributes are fully functionally dependent on the primary key

### 5.3 Third Normal Form (3NF)

* All entities satisfy 2NF
* No transitive dependencies exist (non-key attributes depend only on the primary key)

## 6. Additional Constraints and Indexes

### 6.1 Check Constraints

**Room Entity:**

* CHECK (availability\_status IN ('Available', 'Occupied'))

**Booking Entity:**

* CHECK (check\_out\_date > check\_in\_date)

**Payment Entity:**

* CHECK (amount\_paid > 0)

### 6.2 Indexes

**Primary Key Indexes:**

* Automatically created on room\_id, guest\_id, staff\_id, booking\_id, and payment\_id

**Foreign Key Indexes:**

* Create indexes on guest\_id, room\_id, and staff\_id in the Booking table
* Create index on booking\_id in the Payment table

**Performance Indexes:**

* Create index on room\_number for faster room lookups
* Create index on email in the Guest table for faster guest lookups
* Create composite index on check\_in\_date and check\_out\_date in the Booking table for efficient date range queries

## 7. Data Integrity Rules

### 7.1 Referential Integrity

* Delete or update operations on parent tables (Room, Guest, Staff) should cascade or restrict based on business needs.
* Foreign key constraints ensure data consistency across tables

### 7.2 Entity Integrity

* Primary key constraints ensure uniqueness and non-null values for entity identifiers
* Unique constraints prevent duplicate records for business-critical attributes

### 7.3 Domain Integrity

* Data types enforce appropriate value formats
* CHECK constraints ensure valid values within defined domains
* DEFAULT values provide consistent initial states

## 8. Database Diagram

The logical model is visualized in the provided ER diagram, illustrating the entities, attributes, and relationships. The diagram shows:

* Primary keys indicated with the PK notation
* Foreign keys indicated with the FK notation
* Relationships between entities with appropriate cardinality notation
* Constraints and default values for key attributes

## 9. Implementation Considerations

### 9.1 Physical Database Design

When implementing this logical model in Oracle:

* Use appropriate Oracle data types (NUMBER for INT, VARCHAR2 for VARCHAR, etc.)
* Consider partitioning for large tables like Booking and Payment based on date ranges
* Implement appropriate tablespaces for optimal storage

### 9.2 Performance Optimization

* Use appropriate indexing strategies for frequently queried columns
* Consider materialized views for complex reporting queries
* Implement table partitioning for historical data

### 9.3 Security Considerations

* Implement role-based access control for different staff positions
* Encrypt sensitive guest information
* Establish audit trails for tracking changes to critical data

## 10. PL/SQL Implementation Plan

The following PL/SQL components will be developed based on this logical model:

**Stored Procedures:**

* Room management procedures (add, update, delete)
* Booking procedures (create, modify, cancel)
* Guest registration procedures
* Payment processing procedures

**Functions:**

* Calculate total stay cost based on room type and dates
* Check room availability for given dates
* Generate booking confirmation numbers
* Validate guest information

**Triggers:**

* Update room availability status when bookings are created or modified
* Enforce business rules for bookings and payments
* Implement auditing for sensitive operations
* Apply the weekday and holiday restrictions as specified in the requirements

**Packages:**

* Organize related procedures and functions into logical units
* Implement comprehensive error handling
* Create modular, reusable code components

## 11. Conclusion

This logical model design provides a comprehensive foundation for implementing the Hotel Management System database. The design ensures data integrity, optimizes performance, and supports all the business rules specified in the requirements. The model is normalized to the Third Normal Form and includes all necessary constraints and relationships to maintain a consistent and accurate representation of the hotel's operations.

The logical model is ready for implementation in Oracle using PL/SQL and will serve as the blueprint for the physical database design and development of the application layer components.