****

# School of computing

Department of software engineering

COURSES TITLE:- ADVANCED DATABASE SYSTEM

COURSE CODE:-**SEng3041**

hotel management database

group-2

NAME id

1. DAWIT ENDASHAW …………………………………………….…………. …WDU145838

2. DAGIM SEWUNET…………………………………………………………..……….WDU145771

3. MOHAMMED HASSEN………… ………………………………………………….WDU146918

4. ASEFU MOLLA………………………………………………………………..………WDU145461

5. KALEWONGIL TENAGNE ……………………………………………….………...WDU146502

SUBMITED TO : MR SEID M.

SUBMITTED DATE: 28/08/2016 E.C

**1: Introduction**

**1.1 Background**

The hotel industry is a highly competitive and customer-centric sector where effective database management plays a crucial role in providing personalized services, optimizing operations, and enhancing guest satisfaction. Hotels need to efficiently manage guest information, reservations, room allocations, billing, and other operational aspects to deliver a seamless experience to their guests. However, manual processes and disparate systems can lead to inefficiencies, errors, and data inconsistencies.

**1.2 Objective**

The primary objective of the hotel database management system is to streamline operations, improve efficiency, and enhance guest experience by centralizing and automating key processes. By implementing a robust database system, hotels can effectively manage guest information, reservations, room allocations, billing, and other operations in a secure and organized manner.

**1.3 Scope**

The scope of the project includes developing a comprehensive database management system that covers functionalities such as user management, guest management, reservation management, and reporting. The system will cater to various stakeholders including hotel administrators, front desk staff, housekeeping teams, and guests. The goal is to create a user-friendly and efficient system that meets the diverse needs of a hotel environment.

**2: Functional Requirements**

**2.1 User Management**

User management functionality will include defining different user roles (e.g., admin, front desk staff, housekeeping) with specific access levels and permissions. Admin users will have full control over the system settings, while front desk staff can manage guest information, reservations, and room allocations. Housekeeping staff may have limited access to view room status and update cleaning schedules.

**2.2 Guest Management**

Guest management features will allow the system to capture and store detailed guest information such as name, contact details, preferences, loyalty program status, and stay history. Front desk staff will be able to efficiently check-in/check-out guests, assign rooms based on preferences, and manage room allocations to ensure a smooth guest experience.

**2.3 Reservation Management**

Reservation management functionality will enable guests to make reservations through the front desk. The system will provide real-time availability of rooms, display room options based on guest preferences (e.g., room type, view), and send confirmation emails to guests upon successful reservation. Front desk staff will have access to reservation details and be able to make changes as needed.

**3: Non-Functional Requirements**

**3.1 Performance**

The system should be designed to handle a large volume of data efficiently without compromising performance. Response times for queries, transactions, and system operations should be optimized to provide a seamless user experience for both hotel staff and guests.

**3.2 Security**

Security measures such as data encryption, user authentication mechanisms, role-based access control, and audit trails should be implemented to protect sensitive guest information and prevent unauthorized access. Regular security audits and updates should be conducted to ensure the system's integrity.

**3.3 Scalability**

The system should be scalable to accommodate the growing needs of the hotel, including an increase in the number of guests, rooms, and operations. It should be designed to handle additional load without significant changes to the system architecture or performance degradation.

The system should be able to handle a large volume of data and user requests without experiencing performance issues or downtime.

**3.4 Reliability**

To ensure data integrity and reliability, regular backups of the database should be performed. Data validation checks should be in place to prevent errors and inconsistencies. The system should also the system must be reliable and available 24/7 to ensure that hotel operations are not disrupted.

**3.5 Backup and Recovery**

The system should have regular automated backups of data to prevent data loss in case of system failure, and should have a well-defined recovery plan in place.

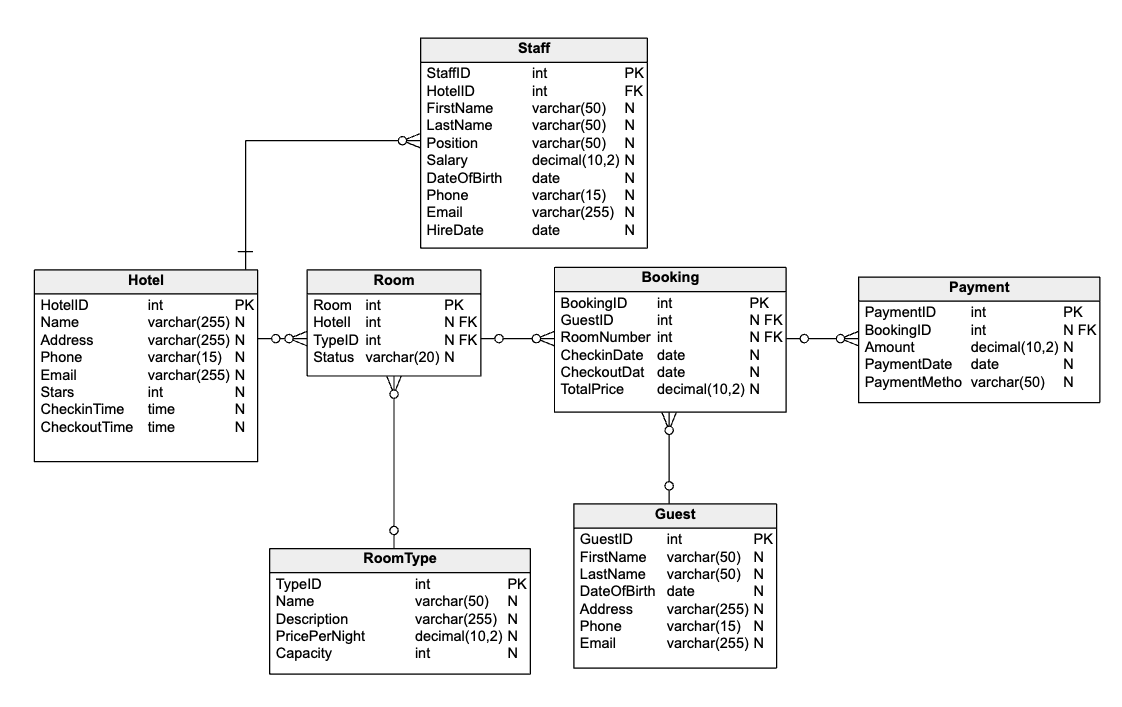
**3.6 Maintenance**

The system should have a maintenance plan in place to ensure that it is regularly updated with the latest security patches and enhancements, and that any issues are promptly addressed by a dedicated support team.

**3.7 Disaster Recovery**

The system should have a disaster recovery plan in place to ensure that critical data and functionality can be restored in the event of a catastrophic event, such as a natural disaster or cyber attack.have fault-tolerant architecture and disaster recovery plans to minimize downtime and ensure continuous operation.

Building a Hotel Management Data Model

Understanding Entities, Attributes, and Relationships Entities and Attributes

Entities are the main objects or concepts in a database; they represent distinct categories of information. They are typically organized into tables within the database. For example, hotel management system entities could include Hotel, Room, Guest, and Booking.

Attributes, on the other hand, are the characteristics or properties that describe entities. They define what specific information is stored for each entity. For instance, attributes for the Hotel entity might include HotelName, Address, and Star Rating; each provides details about a hotel's identity and features.

Entities and attributes collectively structure and organize data within a database, ensuring that information is efficiently stored and managed.

Let’s look at the entities and attributes in our hotel management data model.

**1. Hotel**

The Hotel entity represents an individual hotel property within the system. It serves as the core entity around which all other operations and entities revolve. It stores essential information about each hotel, such as its name, physical address, contact details (phone and email), star rating, check-in time, and check-out time. This data allows the system to manage hotel-specific data and provide accurate information to guests during the booking, check-in, and check-out processes. Its attributes are:

* HotelID: A unique identifier for each hotel. It serves as the primary key for this entity.
* Name: The name of the hotel, providing its distinct identity.
* Address: The physical location of the hotel.
* Phone: The contact phone number for the hotel.
* Email: The contact email address for the hotel.
* Stars: The star rating or level of the hotel, indicating its quality and services.
* CheckinTime: The time at which guests can check in.
* CheckoutTime: The time at which guests are expected to check out.

This table's SQL code is:

|  |
| --- |
| CREATE TABLE Hotel (      HotelID INT PRIMARY KEY,      Name VARCHAR(255),      Address VARCHAR(255),      Phone VARCHAR(15),      Email VARCHAR(255),      Stars INT,      CheckinTime TIME,      CheckoutTime TIME  ); |

**2. RoomType**

The RoomType entity defines the various categories or types of rooms available in the hotel, categorizing rooms based on their features and pricing. It records and stores information like the type's name, description, price per night, and maximum guest capacity. This data assists in room inventory management, pricing strategies, and helping guests choose rooms that best suit their needs and budget.

* TypeID: A unique identifier for each room type and this table’s primary key of this entity.
* Name: The name of the room type (e.g., Standard, Deluxe), defining the category.
* Description: A brief description of the room type, offering additional details.
* Price per Night: The cost of renting this type of room per night.
* Capacity: The maximum number of guests the room type can accommodate.

The following is this table's SQL code:

|  |
| --- |
| CREATE TABLE RoomType (      TypeID INT PRIMARY KEY,      Name VARCHAR(50),      Description VARCHAR(255),      PricePerNight DECIMAL(10, 2),      Capacity INT  ); |

**3. Room**

The Room entity represents individual hotel rooms, serving as the primary entity for room-related operations and management.Each Room record includes a unique RoomNumber and is associated with a specific Hotel and RoomType. The Room entity also tracks the room's status (e.g., available, occupied, or under maintenance), enabling the system to manage room assignments, occupancy, and maintenance schedules.

* RoomNumber: A unique identifier for each room and the primary key of this entity.
* HotelID: A reference to the hotel the room belongs to, establishing a relationship with the Hotel
* TypeID: A reference to the room type of the room, establishing a relationship with the RoomType
* Status: The current status of the room, which assists in managing room availability.

This table's SQL code is:

|  |
| --- |
| CREATE TABLE Room (      RoomNumber INT PRIMARY KEY,      HotelID INT,      TypeID INT,      Status VARCHAR(20),      FOREIGN KEY (HotelID) REFERENCES Hotel(HotelID),      FOREIGN KEY (TypeID) REFERENCES RoomType(TypeID)  ); |

**4. Guest**

The Guest entity captures information about the hotel's guests, forming the basis for managing guest stays and services. It stores essential guest details, such as their name, date of birth, address, and contact information (phone and email). This allows the hotel to personalize services, maintain guest histories, and facilitate communication with guests.

* GuestID: A unique identifier for each guest and the primary key of this entity.
* FirstName: The guest's first name.
* LastName: The guest's last name.
* DateOfBirth: The guest's date of birth, for age verification and personalized service.
* Address: The guest's address.
* Phone: The guest's phone number.
* Email: The guest's email address.

The following is this table's SQL code:

|  |
| --- |
| CREATE TABLE Guest (      GuestID INT PRIMARY KEY,      FirstName VARCHAR(50),      LastName VARCHAR(50),      DateOfBirth DATE,      Address VARCHAR(255),      Phone VARCHAR(15),      Email VARCHAR(255)  ); |

**5. Booking**

The Booking entity manages reservations made by guests for specific rooms on specific dates. Each Booking record is associated with a Guest and a Room, along with check-in and check-out dates. The Booking entity calculates the total price for the stay and assists in managing room availability and guest arrivals and departures.

* BookingID: A unique identifier for each booking and the primary key of this entity.
* GuestID: A reference to the guest making the booking, establishing a relationship with the Guest
* RoomNumber: A reference to the room being booked, establishing a relationship with the Room
* CheckinDate: The date the guest plans to check in.
* CheckoutDate: The date the guest plans to check out.
* Total Price: The total price for the booking.

This table's SQL code is:

|  |
| --- |
| CREATE TABLE Booking (      BookingID INT PRIMARY KEY,      GuestID INT,      RoomNumber INT,      CheckinDate DATE,      CheckoutDate DATE,      TotalPrice DECIMAL(10, 2),      FOREIGN KEY (GuestID) REFERENCES Guest(GuestID),      FOREIGN KEY (RoomNumber) REFERENCES Room(RoomNumber)  ); |

**7. Payment**

The Payment entity stores financial transactions related to guest bookings and services. Each Payment record is linked to a specific Booking, detailing the payment amount, date, and method (e.g., credit card, cash). This entity helps in tracking and managing payments, ensuring accurate billing and financial reporting.

* PaymentID: A unique identifier for each payment and the primary key of this entity.
* BookingID: A reference to the booking associated with the payment, establishing a relationship with the Booking
* Amount: The amount of the payment.
* PaymentDate: The date when the payment was made.
* PaymentMethod: The method used for payment (e.g., credit card, cash).

This table's SQL code is:

|  |
| --- |
| CREATE TABLE Payment (      PaymentID INT PRIMARY KEY,      BookingID INT,      Amount DECIMAL(10, 2),      PaymentDate DATE,      PaymentMethod VARCHAR(50),      FOREIGN KEY (BookingID) REFERENCES Booking(BookingID)  ); |

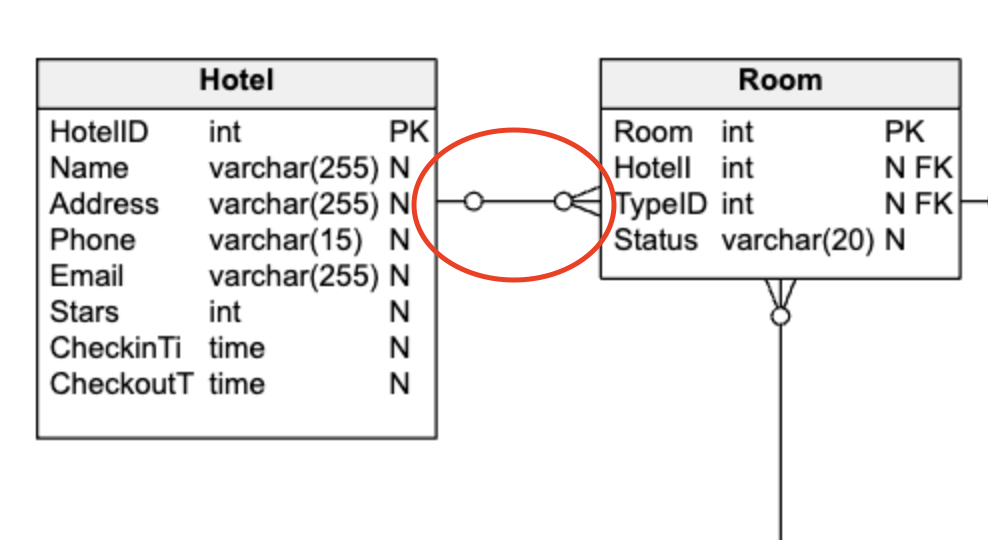
**Relationships and Cardinalities**

Relationships in the context of a [DATABASE](https://vertabelo.com/blog/database-schema-examples/) model describe how different entities or tables within a database are connected or related to each other. These relationships define how data in one entity is associated with data in another entity. Relationships are established through keys or fields in tables; they are essential for organizing and querying data efficiently.

Cardinalities refer to the numerical nature of these relationships, indicating how many instances of one entity are associated with how many instances of another entity. Cardinalities specify whether the relationship is one-to-one (1:1), one-to-many (1:N), or many-to-many (N:N). Cardinalities help define the structure and integrity of the database model

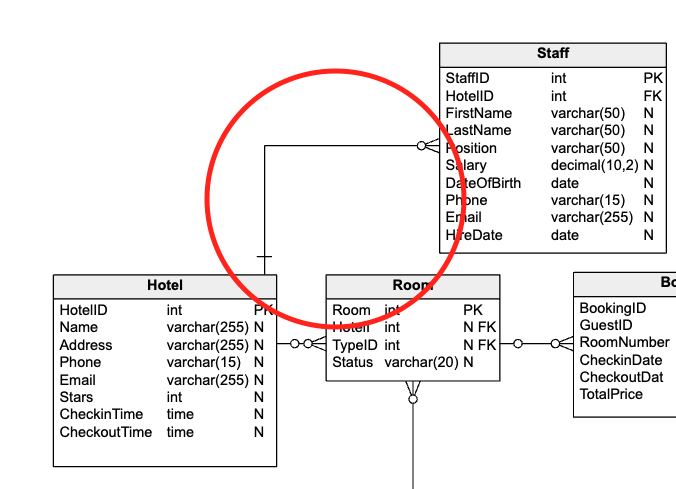
**1.    Hotel–Room**

The relationship between the Hotel and Room entities exhibits a one-to-many (1:N) cardinality. This means that one hotel can have multiple rooms, but each room is associated with one specific hotel. In this model, the HotelID in the Room entity serves as a foreign key referencing the Hotel entity; this indicates which hotel a particular room belongs to. This allows hotels with various room types and configurations to efficiently manage their room inventory while ensuring that each room is tied to a specific hotel.



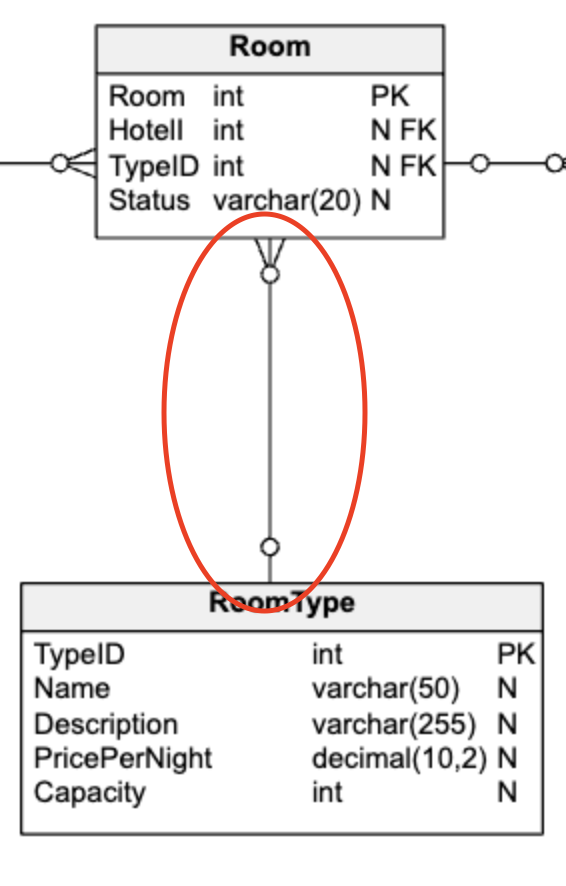
**2.    Hotel–Staff**

The Hotel and Staff entities are related in a one-to-many (1:N) cardinality, meaning a single hotel can employ multiple staff members, but each staff member is linked to only one hotel. In this model, the HotelID in the Staff entity acts as a foreign key that references the Hotel entity, specifying the hotel to which a particular staff member is affiliated. This structure facilitates the efficient management of staff in hotels with diverse departments and functions while ensuring each staff member is associated with a specific hotel.



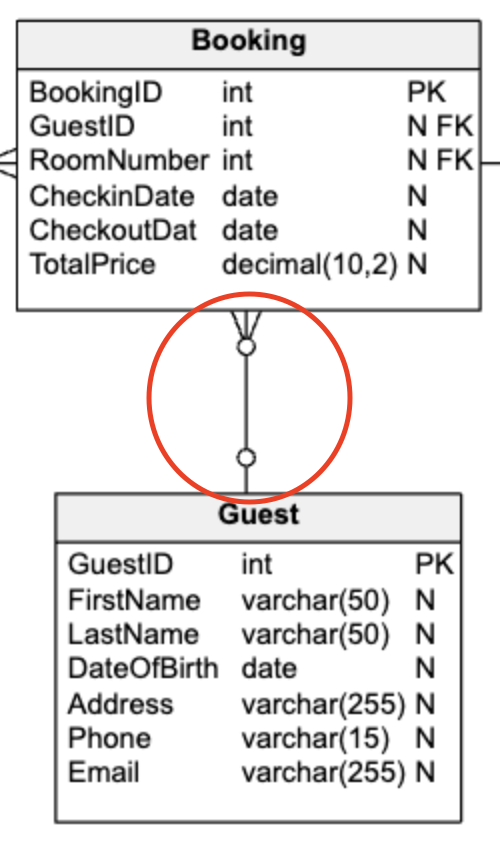
**3.    RoomType–Room**

The connection between the RoomType and Room entities also demonstrates a one-to-many (1:N) cardinality. This implies that each room type can be assigned to multiple rooms, but each room is associated with only one room type. The TypeID attribute in the Room entity acts as a foreign key pointing to the RoomType entity, indicating the specific type of room that it represents. This cardinality allows hotels to offer diverse room categories while maintaining consistency in room type definitions.



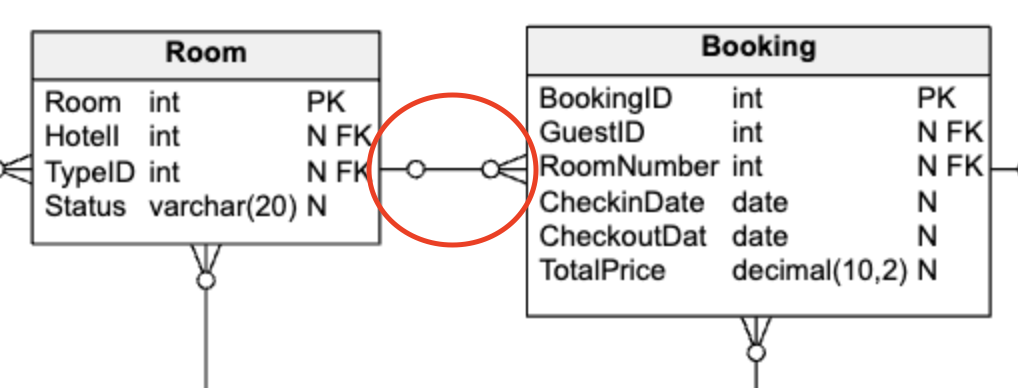
**4. Guest–Booking:**

The relationship between the Guest and Booking entities exhibits a one-to-many (1:N) cardinality as well. This signifies that a guest can make multiple bookings, but each booking is linked to one specific guest. The GuestID attribute in the Booking entity serves as a foreign key referencing the Guest entity, indicating who made the booking. This cardinality facilitates guest-specific booking records, allowing hotels to keep track of reservations made by individual guests over time.



**5.    Booking–Room**

The connection between the Booking and Room entities represents a one-to-many (1:N) cardinality; a single booking can include multiple rooms, but each room is associated with one booking. The RoomNumber attribute in the Booking entity acts as a foreign key referencing the Room entity, indicating the rooms reserved as part of that booking. This cardinality enables hotels to accommodate various booking scenarios, including reservations for multiple rooms within a single booking.



**6.    Booking–Payment**

The relationship between the Booking and Payment entities showcases a one-to-many (1:N) cardinality. This implies that one booking can be associated with multiple payment transactions, but each payment is linked to a single booking. The BookingID attribute in the Payment entity serves as a foreign key referencing the Booking entity, connecting each payment to the corresponding booking. This cardinality allows hotels to manage payment records associated with individual bookings, ensuring accurate financial tracking.

