**[Student Management System (DBMS)]**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfilment of the requirements to award the degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**AP22110010040**

**AP22110010014**

**AP22110010055**

**AP22110010056**

**A picture containing text

Description automatically generated**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[April, 2024]**

INDEX

Abstract, Aim, Attributes, Entity……..3-4

Relations……..5

ER-Model Diagram……..6

Relational Diagram……..7

Normalization……..8-9

Entity records……..10-16

QUERIES AND VIEWS……..17-20

**Abstract**:

The Hotel Management System is a database system designed to facilitate the management of a hotel's operations, including guest reservations, room occupancy, staff directory, billing, and payments. This report provides an overview of the database schema, functionality, and sample data.

**Aim**: to efficiently manage hotel information and guest data within an hotel.

1. Tables:
   * Guests: Stores information about guests staying at the hotel, including their names, contact information, and room numbers.
   * Rooms: Contains data about the rooms available in the hotel, including room numbers, types (single, double, suite), and rates.
   * Reservations: Records reservations made by guests, including check-in and check-out dates, room numbers, and guest information.
   * RoomServiceOrders: Stores information about room service orders placed by guests, including the items ordered, quantities, and prices.
   * HotelStaff: Stores information about hotel staff members, including their names, positions, and contact information.
2. Views:
   * OccupiedRooms: Provides a view of currently occupied rooms along with guest names and check-in/check-out dates.
   * AvailableRooms: Presents a list of available rooms along with room numbers and types.
   * GuestContactInfo: Offers a view of guest names along with their contact email and phone numbers.
   * HighSpendingGuests: Displays guests with high spending amounts along with the corresponding room numbers and total expenses.
   * RoomServiceRevenue: Presents a summary of revenue generated from room service orders, including total revenue and breakdown by item category.

Attributes:

1. GuestID: Unique identifier for each guest record.
2. Email: Email address of the guest.
3. Phone: Phone number of the guest.
4. Address: Physical address of the guest.
5. RoomNumber: Unique identifier for each room.
6. RoomType: Type of the room (e.g., single, double, suite).
7. Rate: Rate per night for the room.
8. ReservationID: Unique identifier for each reservation.
9. CheckInDate: Date when the guest checks in.
10. CheckOutDate: Date when the guest checks out.
11. StaffID: Unique identifier for each staff member.
12. StaffName: Name of the staff member.
13. Position: Position of the staff member (e.g., receptionist, housekeeper).
14. Department: Department to which the staff member belongs (e.g., front desk, housekeeping).
15. Location: Location of the hotel department within the hotel premises.

Relationships:

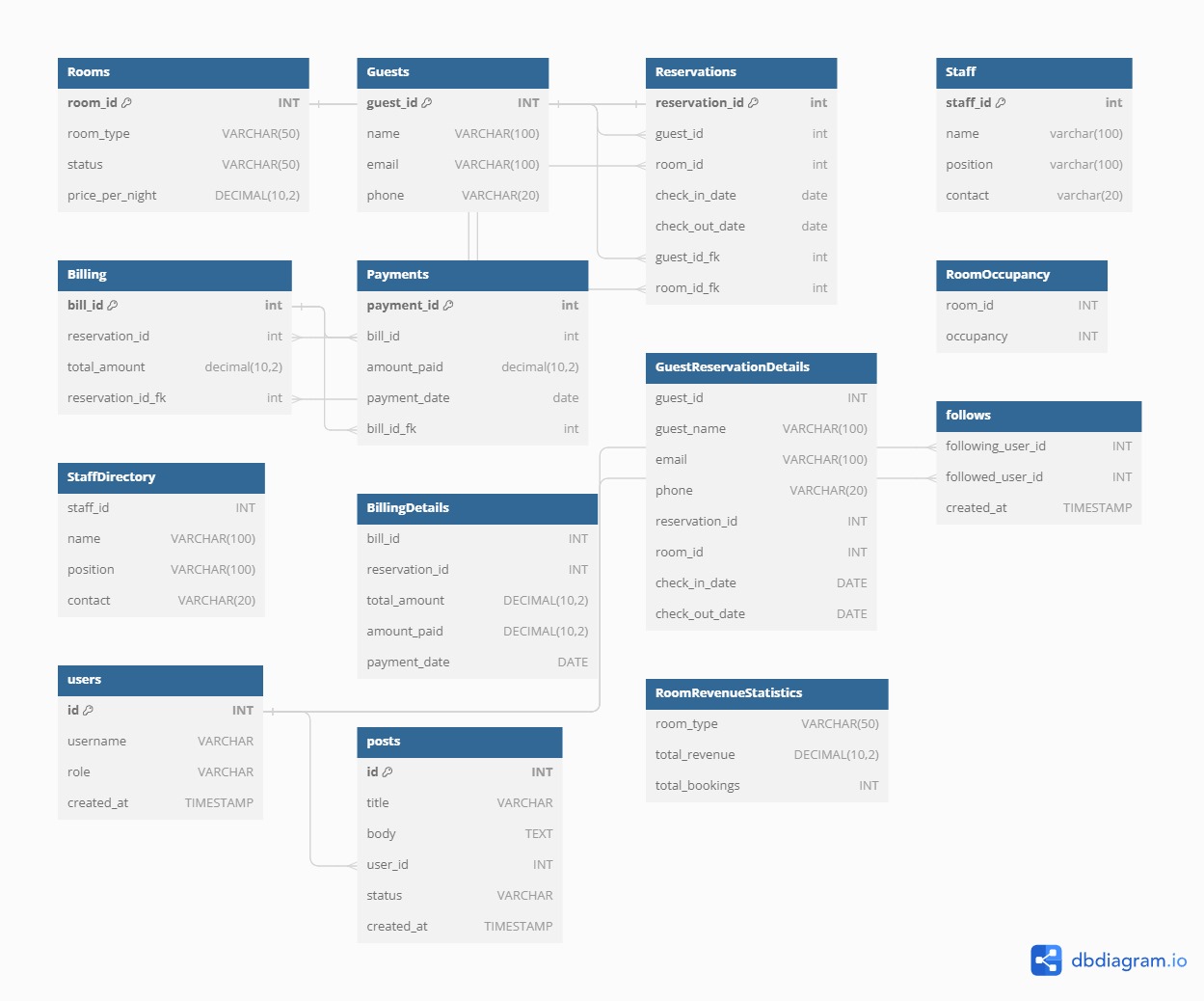
1. Guests:
   * GuestID is related to Email, Phone, and Address.
2. Rooms:
   * RoomNumber is related to RoomType and Rate.
3. Reservations:
   * ReservationID is related to GuestID, RoomNumber, CheckInDate, and CheckOutDate.
4. HotelStaff:
   * StaffID is related to StaffName, Position, Department, and Location.

**E-R Model:**

ER model stands for an Entity-Relationship model.

It is a high-level data model. This model is used to define the data elements and relationships for a specified system.

It develops a conceptual design for the database. It also develops a very simple and easy-to-design view of data.

 ER diagram her

**Relational Diagram**:

Converting ER model to tables/relations, commonly used, flexible.

Each and every column header is called an attribute. The row header is called a tuple.

Here's the relational diagram for the hotel management system based on the provided schema:

```

Guests(GuestID, Email, Phone, Address)

|

|\_\_ Reservations(ReservationID, GuestID, RoomNumber, CheckInDate, CheckOutDate)

|

|\_\_ GuestContactInfo (View: Guests.GuestID, Guests.Email, Guests.Phone)

Rooms(RoomNumber, RoomType, Rate)

|

|\_\_ Reservations(ReservationID, GuestID, RoomNumber, CheckInDate, CheckOutDate)

|

|\_\_ AvailableRooms (View: Rooms.RoomNumber, Rooms.RoomType, Rooms.Rate)

HotelStaff(StaffID, StaffName, Position, Department, Location)

```

In this diagram:

- \*\*Guests\*\* table stores information about hotel guests, including their contact details.

- \*\*Reservations\*\* table records reservations made by guests, including room assignments and dates.

- \*\*Rooms\*\* table contains data about available rooms, including room numbers, types, and rates.

- \*\*HotelStaff\*\* table stores information about hotel staff, including their positions, departments, and locations.

The relationships between these entities allow for the effective management of guest reservations, room assignments, room availability, and staff operations in a hotel setting.

**Normalization:**

Normalization is used to minimize the redundancy from a relation or set of relations.

1. First Normal Form ( 1NF ):

A relation is said to be in its First Normal form if it has got no non-atomic attribute.

(Non-atomic attribute means the attribute which can’t be subdivided).

1. Second Normal Form (2NF):

A relation that is in 1NF is said to have a second normal form if it satisfies any one of the following conditions.

1. The primary key contains only one attribute.
2. There exist no non-key attributes.
3. Every non-key attribute present in the relation should functionally depend upon a full set of the primary key.
4. Third Normal Form (3NF).

The relation in 2Nf is said to be 3NF if there exists no transitive dependency of any non-key attribute on the set of the primary key.

**ENTITY RECORDS**

1. Table

Table: Contact

Below is the SQL command to Create rooms Table

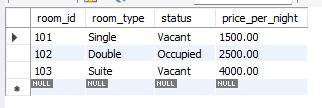
INSERT INTO Rooms (room\_id, room\_type, status, price\_per\_night)

VALUES

(101, 'Single', 'Vacant', 1500),

(102, 'Double', 'Occupied', 2500),

(103, 'Suite', 'Vacant', 4000);

););

2. Table: Guests

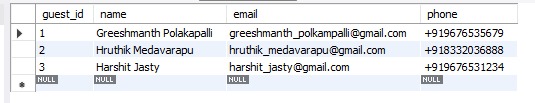
INSERT INTO Guests (guest\_id, name, email, phone)

VALUES

(1, 'Prashanth Sahu', 'prashanth\_sahu@gmail.com', '+1234567890'),

(2, 'Rishi Arasada', 'rishi\_arasada@gmail.com', '+1987654321'),

(3, 'Jasti Kushal', 'jasti\_kushal@gmail.com', '+1122334455');



3.Table: reservations

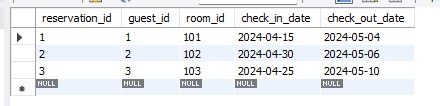
INSERT INTO Reservations (reservation\_id, guest\_id, room\_id, check\_in\_date, check\_out\_date)

VALUES

(1, 1, 101, '2024-04-15', '2024-05-04'),

(2, 2, 102, '2024-04-30', '2024-05-06'),

(3, 3, 103, '2024-04-25', '2024-05-10');



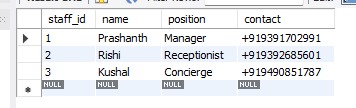
4.Table : Staff

INSERT INTO Staff (staff\_id, name, position, contact)

VALUES

(1, 'Prashanth', 'Manager', '+919391702991'),

(2, 'Rishi', 'Receptionist', '+919392685601'),

(3, 'Kushal', 'Concierge', '+919490851787');

5.Table : Billing info

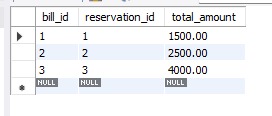
INSERT INTO Billing (bill\_id, reservation\_id, total\_amount)

VALUES

(1, 1, 1500),

(2, 2, 2500),

(3, 3, 4000);



6.Table: Payments

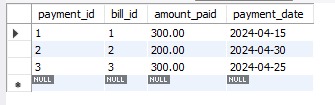
INSERT INTO Payments (payment\_id, bill\_id, amount\_paid, payment\_date)

VALUES

(1, 1, 300, '2024-04-15'),

(2, 2, 200, '2024-04-30'),

(3, 3, 300, '2024-04-25');



QUERIES:

Here are some example queries for the provided hotel management system:

1. Retrieve all guests along with their reservation details:

sql

SELECT \* FROM GuestReservationDetails;

2. Check the occupancy status of all rooms on a specific date ('2024-05-04'):

sql

SELECT \* FROM RoomOccupancy;

3. View the directory of hotel staff:

sql

SELECT \* FROM StaffDirectory;

4. Display billing details along with payment information:

sql

SELECT \* FROM BillingDetails;

5. Get revenue statistics for each room type:

sql

SELECT \* FROM RoomRevenueStatistics;

6. Find out the total amount paid by each guest:

sql

SELECT G.name AS guest\_name, SUM(P.amount\_paid) AS total\_paid

FROM Guests G

JOIN Reservations R ON G.guest\_id = R.guest\_id

JOIN Billing B ON R.reservation\_id = B.reservation\_id

JOIN Payments P ON B.bill\_id = P.bill\_id

GROUP BY G.name;

7. List all guests who haven't paid their bills yet:

sql

SELECT G.name AS guest\_name

FROM Guests G

JOIN Reservations R ON G.guest\_id = R.guest\_id

LEFT JOIN Billing B ON R.reservation\_id = B.reservation\_id

WHERE B.bill\_id IS NULL;

8. Find the highest revenue-generating room type:

sql

SELECT room\_type, total\_revenue

FROM RoomRevenueStatistics

ORDER BY total\_revenue DESC

LIMIT 1;

VIEWS:

1. GuestReservationDetails:

sql

CREATE OR REPLACE VIEW GuestReservationDetails AS

SELECT G.guest\_id, G.name AS guest\_name, G.email, G.phone,

R.reservation\_id, R.room\_id, R.check\_in\_date, R.check\_out\_date

FROM Guests G

JOIN Reservations R ON G.guest\_id = R.guest\_id;

2. RoomOccupancy:

sql

CREATE OR REPLACE VIEW RoomOccupancy AS

SELECT room\_id,

COUNT(CASE WHEN check\_in\_date <= '2024-05-04' AND check\_out\_date >= '2024-05-04' THEN 1 END) AS occupancy

FROM Reservations

GROUP BY room\_id;

3. StaffDirectory:

sql

CREATE OR REPLACE VIEW StaffDirectory AS

SELECT staff\_id, name, position, contact

FROM Staff;

4. BillingDetails:

sql

CREATE OR REPLACE VIEW BillingDetails AS

SELECT B.\*, P.amount\_paid, P.payment\_date

FROM Billing B

JOIN Payments P ON B.bill\_id = P.bill\_id;

5. RoomRevenueStatistics:

sql

CREATE OR REPLACE VIEW RoomRevenueStatistics AS

SELECT R.room\_type,

COALESCE(SUM(B.total\_amount), 0) AS total\_revenue,

COALESCE(COUNT(RV.reservation\_id), 0) AS total\_bookings

FROM Rooms R

LEFT JOIN Reservations RV ON R.room\_id = RV.room\_id

LEFT JOIN Billing B ON RV.reservation\_id = B.reservation\_id

GROUP BY R.room\_type;