Railway Ticket Booking System

Project Title:

Railway Ticket Booking System

Project Background:

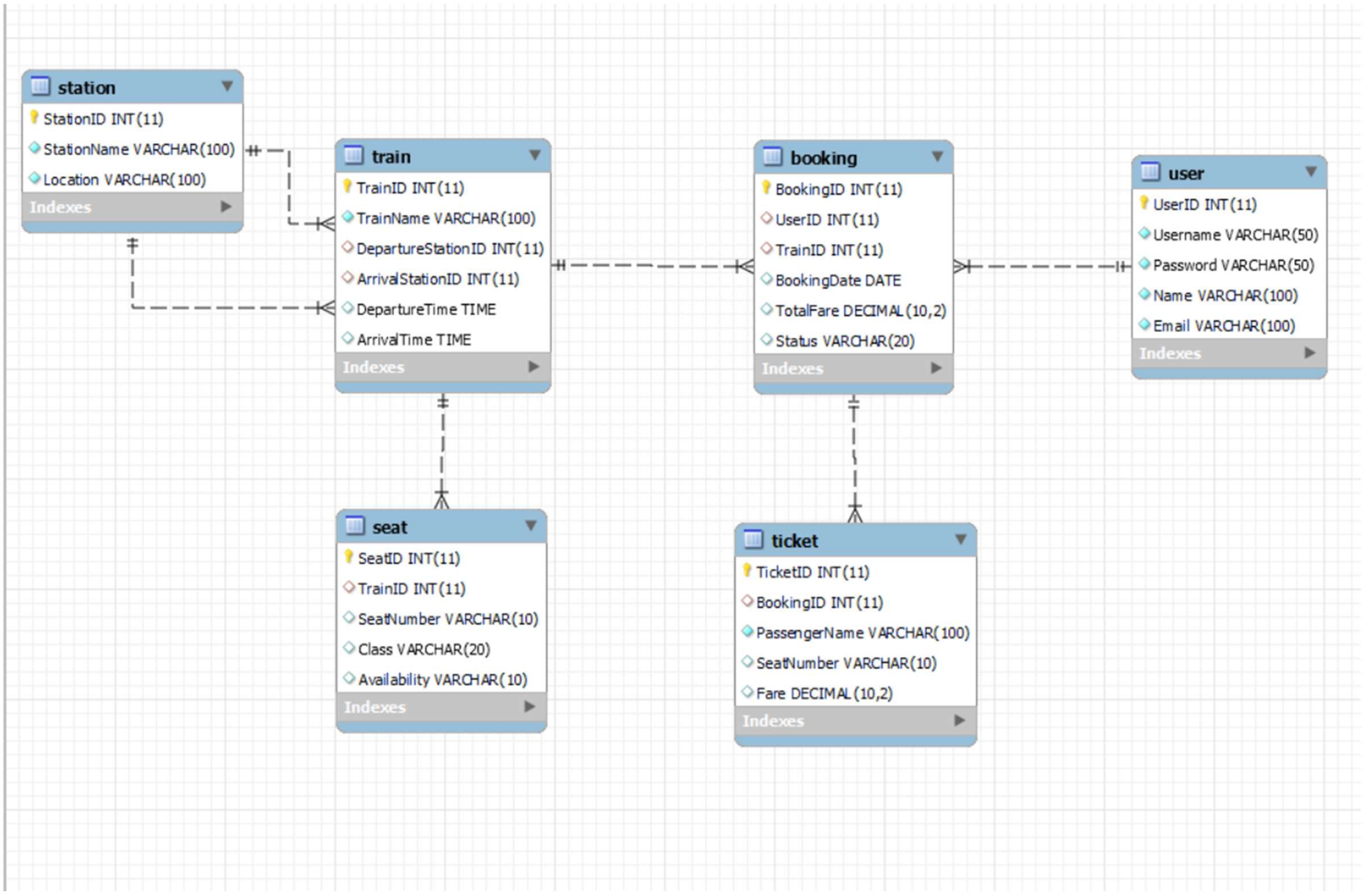
Railway ticket booking systems are vital for managing reservations efficiently in a railway network. This project aims to develop a robust system that allows users to search for trains, book tickets, and manage reservations online.

Description of the Project:

The Railway Ticket Booking System will provide functionalities for users to:

1. Search for trains based on various criteria such as source, destination, date, and class.
2. View available seats and fares for selected trains.
3. Book tickets by providing passenger details and making payments securely.
4. Manage bookings, including cancellation and modification of reservations.
5. Generate e-tickets and receipts for successful bookings.

ER Diagram:


station
StationID INT(11)
StationName VARCHAR(100)
Location VARCHAR(100)
Indexes
train
TrainID INT(11)
TrainName VARCHAR(100)
DepartureStationID INT(11)
ArrivalStationID INT(11)
DepartureTime TIME
ArrivalTime TIME
Indexes
booking
BookingID INT(11)
UserID INT(11)
TrainID INT(11)
BookingDate DATE
TotalFare DECIMAL(10,2)
Status VARCHAR(20)
Indexes
user
UserID INT(11)
Username VARCHAR(50)
Password VARCHAR(50)
Name VARCHAR(100)
Email VARCHAR(100)
Indexes
seat
SeatID INT(11)
TrainID INT(11)
SeatNumber VARCHAR(10)
Class VARCHAR(20)
Availability VARCHAR(10)
Indexes
ticket
TicketID INT(11)
BookingID INT(11)
PassengerName VARCHAR(100)
SeatNumber VARCHAR(10)
Fare DECIMAL(10,2)
Indexes

Description of ER Diagram:

The ER (Entity-Relationship) diagram for the Railway Ticket Booking System illustrates the structure of the database, including entities, attributes, and relationships between them. Here's a description of the main components:

Entities:

1. User: Represents individuals who interact with the system to book tickets. It contains attributes such as UserID (unique identifier), Username, Password, Name, Email, etc.
2. Train: Represents the trains available for booking. Attributes include TrainID (unique identifier), TrainName, DepartureStationID (foreign key referencing StationID), ArrivalStationID (foreign key referencing StationID), DepartureTime, ArrivalTime, etc.
3. Station: Represents the stations where trains stop. It includes attributes like StationID (unique identifier), StationName, Location, etc.
4. Booking: Represents individual bookings made by users. It contains attributes such as BookingID (unique identifier), UserID (foreign key referencing UserID), TrainID (foreign key referencing TrainID), BookingDate, TotalFare, Status, etc.
5. Ticket: Represents individual tickets booked for a particular booking. Attributes include TicketID (unique identifier), BookingID (foreign key referencing BookingID), PassengerName, SeatNumber, Fare, etc.
6. Seat: Represents the seats available on trains. It includes attributes like SeatID (unique identifier), TrainID (foreign key referencing TrainID), SeatNumber, Class, Availability, etc.

Relationships:

1. User-Booking: Represents the association between users and their bookings. It's a one-to-many relationship, indicating that a user can make multiple bookings, but each booking is made by only one user.
2. Train-Station: Represents the stops of trains at stations. It's a many-to-many relationship resolved by an associative entity, TrainStation, which contains TrainID and StationID as foreign keys.

3. Booking-Ticket: Represents the association between bookings and tickets. It's a one-to-many relationship, indicating that each booking can have multiple tickets, but each ticket belongs to only one booking.
4. Train-Seat: Represents the allocation of seats to trains. It's a one-to-many relationship, indicating that each train can have multiple seats, but each seat is associated with only one train.

The ER diagram visually represents how these entities are related to each other, helping in the design and understanding of the database structure for the Railway Ticket Booking System.

Conversion of ER diagram into Tables :

```sql
-- User Table

CREATE TABLE User (

    UserID INT PRIMARY KEY,

    Username VARCHAR(50) NOT NULL,

    Password VARCHAR(50) NOT NULL,

    Name VARCHAR(100) NOT NULL,

    Email VARCHAR(100) NOT NULL

);

-- Train Table

CREATE TABLE Train (

    TrainID INT PRIMARY KEY,

    TrainName VARCHAR(100) NOT NULL,

    DepartureStationID INT,

    ArrivalStationID INT,

    DepartureTime TIME,

    ArrivalTime TIME,

    FOREIGN KEY (DepartureStationID) REFERENCES Station(StationID),

    FOREIGN KEY (ArrivalStationID) REFERENCES Station(StationID)

);

-- Station Table

CREATE TABLE Station (

    StationID INT PRIMARY KEY,

    StationName VARCHAR(100) NOT NULL,

    Location VARCHAR(100) NOT NULL

);
```

```
-- Booking Table

CREATE TABLE Booking (

    BookingID INT PRIMARY KEY,

    UserID INT,

    TrainID INT,

    BookingDate DATE,

    TotalFare DECIMAL(10,2),

    Status VARCHAR(20),

    FOREIGN KEY (UserID) REFERENCES User(UserID),

    FOREIGN KEY (TrainID) REFERENCES Train(TrainID)

);
```

```
-- Ticket Table

CREATE TABLE Ticket (

    TicketID INT PRIMARY KEY,

    BookingID INT,

    PassengerName VARCHAR(100) NOT NULL,

    SeatNumber VARCHAR(10),

    Fare DECIMAL(10,2),

    FOREIGN KEY (BookingID) REFERENCES Booking(BookingID)

);
```

-- Seat Table

```
CREATE TABLE Seat (
    SeatID INT PRIMARY KEY,
    TrainID INT,
    SeatNumber VARCHAR(10),
    Class VARCHAR(20),
    Availability VARCHAR(10),
    FOREIGN KEY (TrainID) REFERENCES Train(TrainID)
);
```

Description of Tables:

1. User Table:

- **UserID (INT):** Primary key uniquely identifying each user.
- **Username (VARCHAR(50)):** Username of the user for login.
- **Password (VARCHAR(50)):** Password of the user for login (stored securely).
- **Name (VARCHAR(100)):** Name of the user.
- **Email (VARCHAR(100)):** Email address of the user.

2. Train Table:

- **TrainID (INT):** Primary key uniquely identifying each train.
- **TrainName (VARCHAR(100)):** Name of the train.
- **DepartureStationID (INT):** Foreign key referencing the StationID of the departure station.
- **ArrivalStationID (INT):** Foreign key referencing the StationID of the arrival station.
- **DepartureTime (TIME):** Departure time of the train.
- **ArrivalTime (TIME):** Arrival time of the train.

3. Station Table:

- **StationID (INT):** Primary key uniquely identifying each station.
- **StationName (VARCHAR(100)):** Name of the station.
- **Location (VARCHAR(100)):** Location of the station.

4. Booking Table:

- **BookingID (INT):** Primary key uniquely identifying each booking.
- **UserID (INT):** Foreign key referencing the UserID of the user who made the booking.
- **TrainID (INT):** Foreign key referencing the TrainID of the booked train.
- **BookingDate (DATE):** Date when the booking was made.
- **TotalFare (DECIMAL(10,2)):** Total fare for the booking.
- **Status (VARCHAR(20)):** Status of the booking (e.g., confirmed, canceled).

5. Ticket Table:

- **TicketID (INT):** Primary key uniquely identifying each ticket.
- **BookingID (INT):** Foreign key referencing the BookingID of the booking associated with the ticket.
- **PassengerName (VARCHAR(100)):** Name of the passenger for the ticket.
- **SeatNumber (VARCHAR(10)):** Seat number allocated for the passenger.
- **Fare (DECIMAL(10,2)):** Fare for the ticket.

6. Seat Table:

- **SeatID (INT):** Primary key uniquely identifying each seat.
- **TrainID (INT):** Foreign key referencing the TrainID of the train associated with the seat.
- **SeatNumber (VARCHAR(10)):** Seat number.
- **Class (VARCHAR(20)):** Class of the seat (e.g., economy, business).
- **Availability (VARCHAR(10)):** Availability status of the seat (e.g., available, booked).

Normalization of Tables up to 3-NF:

To normalize the tables up to the third normal form (3-NF) in the Railway Ticket Booking System, we need to ensure that:

1. Each table has a unique primary key.
2. Each non-key attribute is fully functionally dependent on the primary key.
3. There are no transitive dependencies.

Given the entities and their attributes in the system, let's go through the normalization process:

1. User Table:

- UserID (PK)
- Username
- Password
- Name
- Email

2. Train Table:

- TrainID (PK)
- TrainName
- DepartureStationID (FK)
- ArrivalStationID (FK)
- DepartureTime
- ArrivalTime

3. Station Table:

- StationID (PK)
- StationName
- Location

4. Booking Table:

- BookingID (PK)
- UserID (FK)
- TrainID (FK)
- BookingDate
- TotalFare
- Status

5. Ticket Table:

- TicketID (PK)
- BookingID (FK)
- PassengerName
- SeatNumber
- Fare

6. Seat Table:

- SeatID (PK)
- TrainID (FK)
- SeatNumber
- Class
- Availability

Now, let's analyze the tables for normalization:

First Normal Form (1-NF):

All tables already satisfy 1-NF since each attribute contains atomic values, and there are no repeating groups.

Second Normal Form (2-NF):

- No partial dependencies exist.
- All non-key attributes are fully functionally dependent on the primary key.

Third Normal Form (3-NF):

- No transitive dependencies exist.

All tables appear to be in 3-NF. Each table has a unique primary key, and all non-key attributes are directly dependent on the primary key without any transitive dependencies.

Thus, the tables in the Railway Ticket Booking System are normalized up to the third normal form (3-NF), ensuring data integrity and minimizing redundancy.

Creation of Data in the Tables:

<u>User Table:</u>

SQL Command:

-- Inserting data into User Table

INSERT INTO User (UserID, Username, Password, Name, Email)

VALUES

(1, 'user123', '********', 'John Doe', 'john@example.com'),

(2, 'janesmith', '********', 'Jane Smith', 'jane@example.com'),

(3, 'mike23', '********', 'Mike Brown', 'mike@example.com'),

(4, 'sarah88', '********', 'Sarah Lee', 'sarah@example.com'),

(5, 'davidK', '********', 'David Kim', 'david@example.com');

UserID	Username	Password	Name	Email
1	user123	********	John Doe	john@example.com
2	janesmith	********	Jane Smith	jane@example.com
3	mike23	********	Mike Brown	mike@example.com
4	sarah88	********	Sarah Lee	sarah@example.com
5	davidK	********	David Kim	david@example.com

<u>Train Table:</u>

SQL Command:

-- Inserting data into Train Table

INSERT INTO Train (TrainID, TrainName, DepartureStationID, ArrivalStationID, DepartureTime, ArrivalTime)

VALUES

(101, 'Express 101', 1, 5, '08:00', '12:00'),

(102, 'FastTrack 102', 3, 4, '10:30', '14:30'),

(103, 'SuperSonic 103', 2, 2, '12:00', '16:30'),

(104, 'Speedy 104', 5, 3, '15:00', '19:30'),

(105, 'RapidTransit 105', 4, 1, '18:30', '22:30');

TrainID	**TrainName**	**DepartureStationID**	**ArrivalStationID**	**DepartureTime**	**ArrivalTime**
101	**Express 101**	**1**	**5**	**08:00**	**12:00**
102	**FastTrack 102**	**3**	**4**	**10:30**	**14:30**
103	**SuperSonic 103**	**2**	**2**	**12:00**	**16:30**
104	**Speedy 104**	**5**	**3**	**15:00**	**19:30**

105	**RapidTransit 105**	**4**	**1**	**18:30**	**22:30**

<u>**Station Table:**</u>

SQL Command:

-- Inserting data into Station Table

```
INSERT INTO Station (StationID, StationName, Location)

VALUES

(1, 'Station A', 'City X'),

(2, 'Station B', 'City Y'),

(3, 'Station C', 'City Z'),

(4, 'Station D', 'City W'),

(5, 'Station E', 'City V');
```

StationID	StationName	Location
1	**Station A**	**City X**
2	**Station B**	**City Y**

3	**Station C**	**City Z**
4	**Station D**	**City W**
5	**Station E**	**City V**

Booking Table:

SQL Command:

-- Inserting data into Booking Table

```
INSERT INTO Booking (BookingID, UserID, TrainID, BookingDate, TotalFare, Status)

VALUES

(1001, 1, 101, '2024-04-20', 50.00, 'Confirmed'),

(1002, 2, 102, '2024-04-21', 70.00, 'Confirmed'),

(1003, 3, 103, '2024-04-22', 80.00, 'Confirmed'),

(1004, 4, 104, '2024-04-23', 90.00, 'Confirmed'),

(1005, 5, 105, '2024-04-24', 60.00, 'Confirmed');
```

BookingID	UserID	TrainID	BookingDate	TotalFare	Status
1001	1	101	2024-04-20	50.00	Confirmed
1002	2	102	2024-04-21	70.00	Confirmed
1003	3	103	2024-04-22	80.00	Confirmed
1004	4	104	2024-04-23	90.00	Confirmed
1005	5	105	2024-04-24	60.00	Confirmed

Ticket Table:

SQL Command:

-- Inserting data into Ticket Table

```
INSERT INTO Ticket (TicketID, BookingID, PassengerName, SeatNumber, Fare)

VALUES

(5001, 1001, 'John Doe', 'A12', 25.00),

(5002, 1002, 'Jane Smith', 'B24', 35.00),

(5003, 1003, 'Mike Brown', 'C34', 40.00),

(5004, 1004, 'Sarah Lee', 'D45', 45.00),

(5005, 1005, 'David Kim', 'E56', 30.00);
```

TicketID	BookingID	PassengerName	SeatNumber	Fare
5001	1001	John Doe	A12	25.00
5002	1002	Jane Smith	B24	35.00

5003	**1003**	**Mike Brown**	**C34**	**40.00**
5004	**1004**	**Sarah Lee**	**D45**	**45.00**
5005	**1005**	**David Kim**	**E56**	**30.00**

Seat Table:

SQL Command:

-- Inserting data into Seat Table

```
INSERT INTO Seat (SeatID, TrainID, SeatNumber, Class, Availability)

VALUES

(9001, 101, 'A12', 'Economy', 'Available'),

(9002, 102, 'B24', 'Business', 'Available'),

(9003, 103, 'C34', 'Economy', 'Available'),

(9004, 104, 'D45', 'First', 'Available'),

(9005, 105, 'E56', 'Economy', 'Available');
```

SeatID	TrainID	SeatNumber	Class	Availability
9001	**101**	**A12**	**Economy**	**Available**
9002	**102**	**B24**	**Business**	**Available**
9003	**103**	**C34**	**Economy**	**Available**

9004	**104**	**D45**	**First**	**Available**
9005	**105**	**E56**	**Economy**	**Available**

Few sql queries on the created tables:

1. Query: Retrieve all bookings made by user 'John Doe' including train details.

SELECT * FROM Booking;

Output:

BookingID	UserID	TrainID	BookingDate	TotalFare	Status
1001	1	101	2024-04-20	50.00	Confirmed
1002	2	102	2024-04-21	70.00	Confirmed
1003	3	103	2024-04-22	80.00	Confirmed
1004	4	104	2024-04-23	90.00	Confirmed
1005	5	105	2024-04-24	60.00	Confirmed

2. Query: List all available seats on the train 'Express 101'.

SELECT * FROM Seat WHERE TrainID = 101 AND Availability = 'Available';

Output:

SeatID	TrainID	SeatNumber	Class	Availability
9001	101	A12	Economy	Available

3. Query: Retrieve all bookings made on 2024-04-22.

SELECT * FROM Booking WHERE BookingDate = '2024-04-22';

Output:

BookingID	UserID	TrainID	BookingDate	TotalFare	Status
1003	3	103	2024-04-22	80.00	Confirmed

Creation of 5 views using the tables

1. View: Find John Doe

This view provides information about JohnDoe

```
CREATE VIEW findJohnDoe AS

SELECT *

FROM Ticket

WHERE PassengerName = "John Doe";



SELECT * FROM findJhonDoe;
```

2. View: User Table

This view list whole User Table

```
CREATE VIEW UserTable AS

SELECT *

FROM User;
```

3. View: BookingsToday

This view displays all bookings made on the current date.

```
CREATE VIEW BookingsToday AS

SELECT *

FROM Booking

WHERE BookingDate = CURRENT_DATE;
```

4. View: Display Station Names

```
CREATE VIEW DisplayStationNames AS

SELECT StationName

FROM Station;
```

5. View: TicketSeatInfo

This view lists Ticket Seat information.

```
CREATE VIEW TicketSeatInfo AS

SELECT TicketID, SeatNumber

FROM Ticket;
```