**How to Design Database for Flight Reservation System**

Last Updated : 27 Feb, 2024

Database design for Airline reservation system is crucial for managing flight bookings effectively. In this guide, we’ll explore the essential components and steps involved in creating such a system for flight booking, ensuring smooth operations, and user satisfaction.

**Database Design for Flight Reservation System**

Database design for airline reservation systems must manage various entities such as flights, passengers, airlines, airports, bookings, and payments. It should support functionalities such as searching for available flights, making reservations, managing passenger information, and processing payments securely. By designing a database that addresses these requirements, airlines, and travel agencies can streamline their operations and provide a seamless booking experience to customers.

**Flight Reservation System Features:**

These are the following features of the airline reservation system.

* **Flight Management:** Efficiently manage flight information, including flight numbers, departure and arrival times, origin and destination airports, and available seats.
* **Passenger Management:** Maintain passenger details, such as names, contact information, and passport numbers.
* **Airline Management:** Keep track of airline information, including airline names, contact details, and operating regions.
* **Airport Management:** Store information about airports, including airport codes, names, locations, and facilities.
* **Booking Management:** Handle flight reservations, including booking IDs, flight details, passenger information, and payment status.
* **Payment Processing:** Securely process payments for flight bookings, including transaction IDs, payment methods, and amounts.

**Entities and Attributes for the Flight Reservation System**

Entities serve as the building blocks of our database, representing the fundamental objects or concepts that need to be stored and managed. Attributes define the characteristics or properties of each entity. Let’s explore each entity and attribute in detail:

**Flight: Represents individual flights operated by airlines.**

* FlightID (Primary Key): Unique identifier for each flight.
* FlightNumber: Unique identifier for the flight assigned by the airline.
* DepartureDateTime: Date and time of flight departure.
* ArrivalDateTime: Date and time of flight arrival.
* OriginAirportCode: Three-letter code representing the origin airport.
* DestinationAirportCode: Three-letter code representing the destination airport.
* AvailableSeats: Number of available seats on the flight.

**Passenger: Represents individuals booking flights.**

* PassengerID (Primary Key): Unique identifier for each passenger.
* FirstName: First name of the passenger.
* LastName: Last name of the passenger.
* Email: Email address of the passenger.
* PassportNumber: Passport number of the passenger.

**Airline: Represents airlines operating flights.**

* AirlineID (Primary Key): Unique identifier for each airline.
* AirlineName: Name of the airline.
* ContactNumber: Contact number of the airline.
* OperatingRegion: Region(s) where the airline operates.

**Airport: Represents airports serving as departure or arrival points for flights.**

* AirportCode (Primary Key): Three-letter code representing the airport.
* AirportName: Name of the airport.
* Location: Location (city, country) of the airport.
* Facilities: Facilities available at the airport (e.g., parking, lounges).

**Booking: Represents flight reservations made by passengers.**

* BookingID (Primary Key): Unique identifier for each booking.
* FlightID (Foreign Key): Reference to the flight booked.
* PassengerID (Foreign Key): Reference to the passenger making the booking.
* PaymentStatus: Status of the payment for the booking (e.g., pending, paid).

**Payment: Represents payments made for flight bookings.**

* PaymentID (Primary Key): Unique identifier for each payment.
* BookingID (Foreign Key): Reference to the booking associated with the payment.
* PaymentMethod: Method used for payment (e.g., credit card, PayPal).
* Amount: Amount paid for the booking.
* TransactionDateTime: Date and time of the payment transaction.

**Relationships Between Entities**

**Flight – Booking Relationship**

* Each booking is associated with one flight (FlightID in the Booking table references FlightID in the Flight table).
* This is a one-to-many relationship, as one flight can have multiple bookings, but each booking corresponds to only one flight.

**Passenger – Booking Relationship**

* Each booking is associated with one passenger (PassengerID in the Booking table references PassengerID in the Passenger table).
* This is a one-to-many relationship, as one passenger can make multiple bookings, but each booking corresponds to only one passenger.

**Flight – Airport Relationship**

* Each flight has an origin airport and a destination airport (OriginAirportCode and DestinationAirportCode in the Flight table reference AirportCode in the Airport table).
* This is a many-to-one relationship, as multiple flights can depart from or arrive at the same airport, but each flight has only one origin and one destination airport.

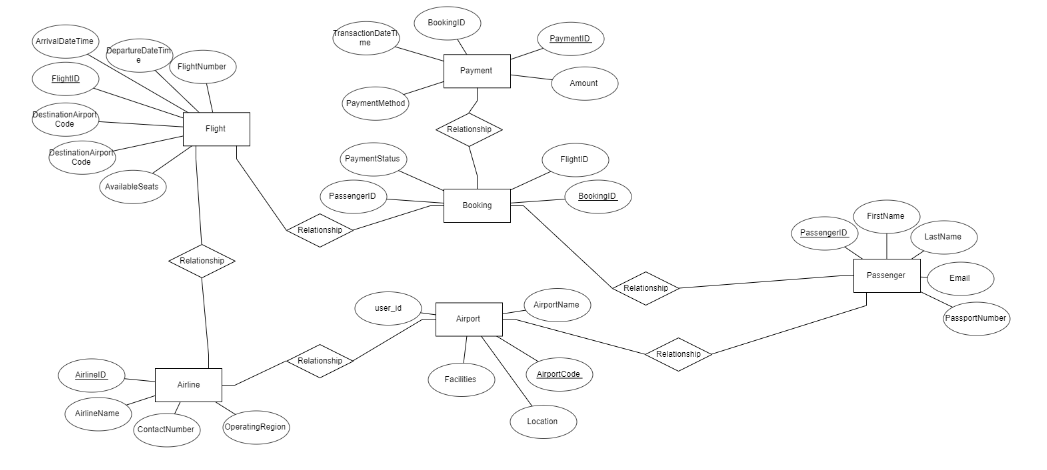
**Airline – Flight Relationship**

* Each flight is operated by one airline (AirlineID in the Flight table references AirlineID in the Airline table).
* This is a one-to-many relationship, as one airline can operate multiple flights, but each flight is operated by only one airline.

**Payment – Booking Relationship**

* Each payment is associated with one booking (BookingID in the Payment table references BookingID in the Booking table).
* This is a one-to-one relationship, as each payment corresponds to only one booking, and each booking can have only one payment.

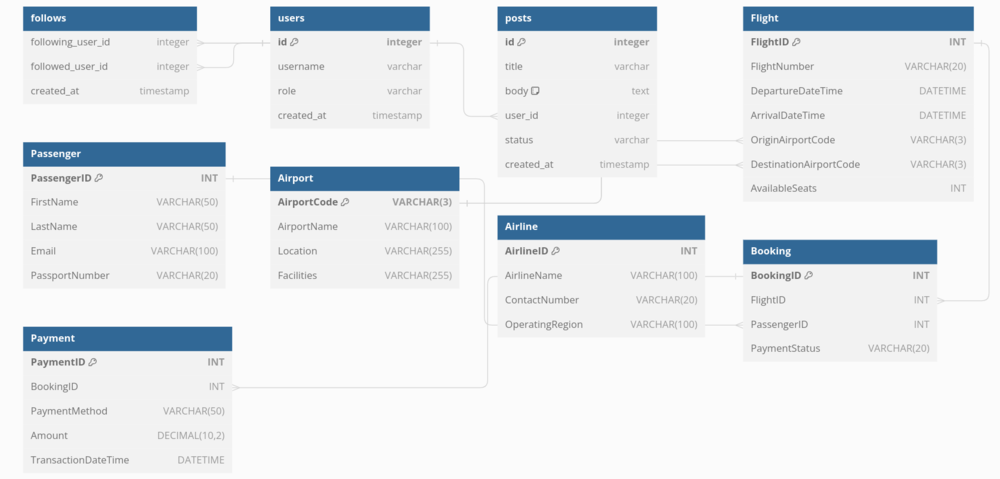
**Representataion of ER Diagram**



**Entities Structures in SQL Format**

-- Flight Table  
CREATE TABLE Flight (  
 FlightID INT PRIMARY KEY,  
 FlightNumber VARCHAR(20) UNIQUE,  
 DepartureDateTime DATETIME,  
 ArrivalDateTime DATETIME,  
 OriginAirportCode VARCHAR(3),  
 DestinationAirportCode VARCHAR(3),  
 AvailableSeats INT,  
 FOREIGN KEY (OriginAirportCode) REFERENCES Airport(AirportCode),  
 FOREIGN KEY (DestinationAirportCode) REFERENCES Airport(AirportCode)  
);  
  
-- Passenger Table  
CREATE TABLE Passenger (  
 PassengerID INT PRIMARY KEY,  
 FirstName VARCHAR(50),  
 LastName VARCHAR(50),  
 Email VARCHAR(100),  
 PassportNumber VARCHAR(20)  
);  
  
-- Airport Table  
CREATE TABLE Airport (  
 AirportCode VARCHAR(3) PRIMARY KEY,  
 AirportName VARCHAR(100),  
 Location VARCHAR(255),  
 Facilities VARCHAR(255)  
);  
  
-- Airline Table  
CREATE TABLE Airline (  
 AirlineID INT PRIMARY KEY,  
 AirlineName VARCHAR(100),  
 ContactNumber VARCHAR(20),  
 OperatingRegion VARCHAR(100)  
);  
  
-- Booking Table  
CREATE TABLE Booking (  
 BookingID INT PRIMARY KEY,  
 FlightID INT,  
 PassengerID INT,  
 PaymentStatus VARCHAR(20),  
 FOREIGN KEY (FlightID) REFERENCES Flight(FlightID),  
 FOREIGN KEY (PassengerID) REFERENCES Passenger(PassengerID)  
);  
  
-- Payment Table  
CREATE TABLE Payment (  
 PaymentID INT PRIMARY KEY,  
 BookingID INT UNIQUE,  
 PaymentMethod VARCHAR(50),  
 Amount DECIMAL(10, 2),  
 TransactionDateTime DATETIME,  
 FOREIGN KEY (BookingID) REFERENCES Booking(BookingID)  
);

**Database Model for Flight Reservation System**



**Conclusion**

Designing a relational database for a flight booking system involves identifying the entities, defining their attributes, establishing relationships between them, and enforcing data integrity. By following a systematic approach and considering the specific requirements of the system, a well-designed database can facilitate efficient flight booking processes and enhance the overall user experience.

[Three 90 Challenge is back on popular demand! After processing refunds worth INR 1CR+, we are back with the offer if you missed it the first time. Get 90% course fee refund in 90 days. Avail now!](https://www.geeksforgeeks.org/courses?utm_source=geeksforgeeks&utm_medium=bottomtext_ad&utm_campaign=three90)

Summer-time is here and so is the time to skill-up! More than 5,000 learners have now completed their journey from **basics of DSA to advanced level development programs** such as Full-Stack, Backend Development, Data Science.   
  
And why go anywhere else when our [DSA to Development: Coding Guide](https://gfgcdn.com/tu/Q8V/) will help you master all this in a few months! Apply now to our [DSA to Development Program](https://gfgcdn.com/tu/Q8V/) and our counsellors will connect with you for further guidance & support.