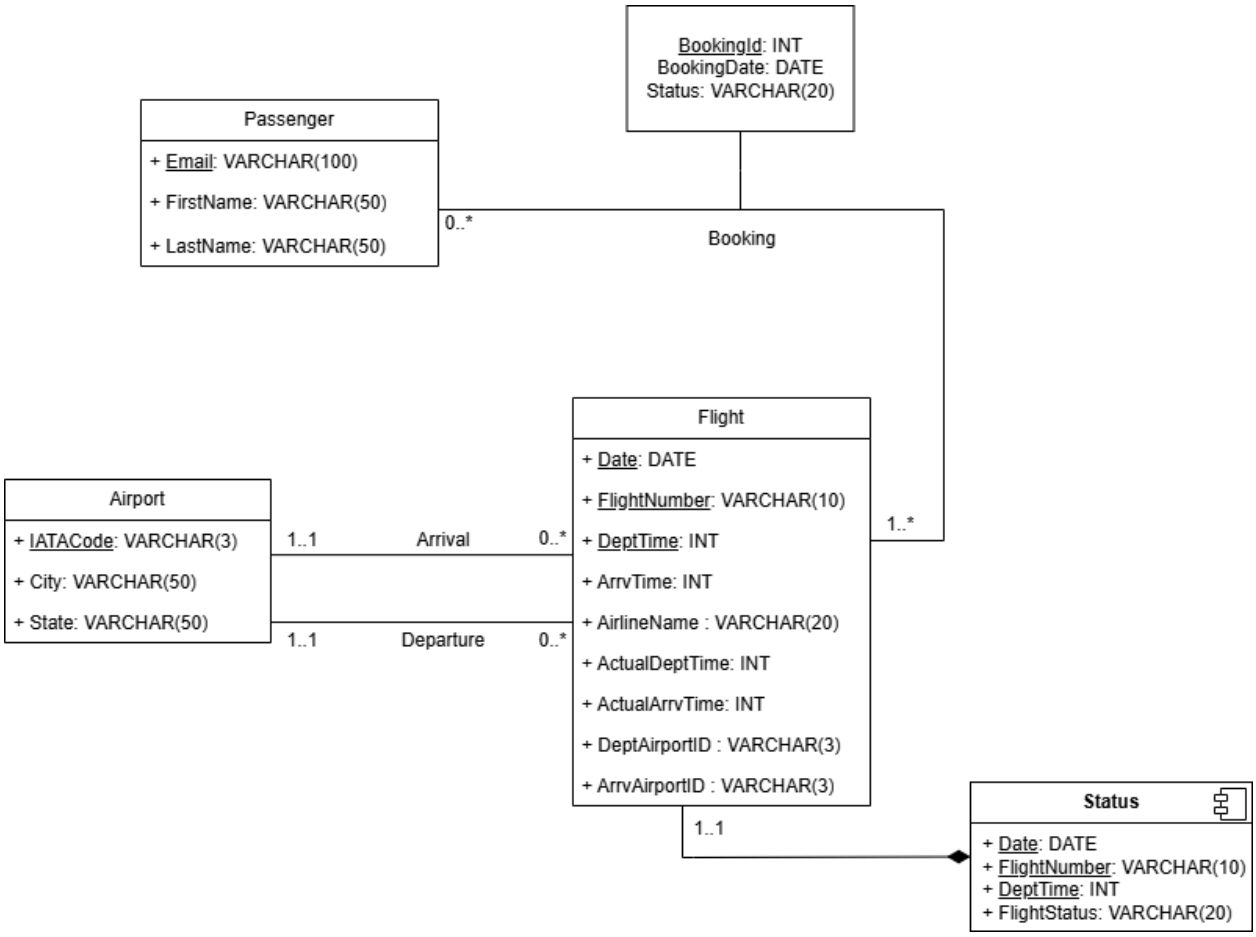
CS 411 Stage2

Project Title: SkyTrack: never miss a flight!

ER/UML diagram



Assumptions:

Flight Assumptions:

1. Each flight has a unique identifier (e.g., flight number + date + time).
2. Flight data includes attributes like operating airline, departure airports, etc.
3. Every flight can only be operated by a specific airline and flies on a specific route (For example UA xxx can only fly from LAX to SFO in every trip it makes)
4. A flight may be a part of a longer flight by being one of the connecting flights. This ensures the flight still takes the same route, however, can either be the last flight after a connecting flight or be an earlier flight before another connecting flight.
5. A flight belongs to only one airline but can have multiple passengers.

Justification for why we modeled flight as Entity and not attribute:

1. A flight contains multiple independent attributes like operating airline, departure airports, and status.
2. Airlines operate multiple flights, so Flight should be a separate entity rather than an attribute of Airline.

Airport Assumptions:

1. Each airport has a unique identifier (e.g., iata code).
2. Multiple flights can depart from a specific airport.
3. Multiple flights can arrive at a specific airport.
4. Airports are located in specific cities and states.

Justification for why we modeled airport as Entity and not attribute:

1. Airports are reusable across multiple flights, meaning that the same airport appears in multiple flights as a departure and arrival location.
2. If an airport updates its name (e.g., Istanbul Atatürk Airport changed to Istanbul Airport), updating it in one place ensures data consistency.
3. Separating it allows route-based analysis, such as determining the busiest airports and commonly used flight paths.

Passenger Assumptions:

1. Each passenger has a unique identifier (e.g., email).
2. A passenger can book multiple flights.

Justification for why we modeled passenger as Entity and not attribute:

1. Passengers can have multiple bookings. If passenger details were embedded in Booking, their details would be duplicated multiple times, leading to redundancy.
2. Passengers are **independent entities**, meaning they may book multiple flights over time, which should be tracked separately.

Booking Assumption:

1. Each booking has a unique booking_id.
2. A booking links one passenger to one flight.
3. A passenger can make multiple bookings.

Justification for why we modeled booking as Entity and not attribute:

1. Booking is a **many-to-many (M:M) relationship** between Passenger and Flight.
2. It ensures that **each booking is unique** and keeps track of booking_status, booking_date, etc.

Status Assumption:

1. Each flight will have one and only status

Justification for why we modeled booking as Entity and not attribute:

1. Having status in the flight table violates 3NF since status is also dependent on arrival/departure time, which creates transitive dependency.

<u>Relational Schema:</u>

AIRPORT (Contains airport details)

Airport(IATACode: CHAR(3) [PK], City: VARCHAR(50), State: VARCHAR(50))

{IATACode} -> {City, State}

IATACode implies city, country, and name.

FLIGHT (Stores flight information)

Flight(Date: DATE [PK], FlightNumber: VARCHAR(10) [PK], DeptTime: INT [PK], ArrvTime: INT, AirlineName VARCHAR(20), ActualDeptTime: INT, ActualArrvTime: INT, DeptAirportID: VARCHAR(3) [FK to Airport.IATACode], ArrvAirportID: VARCHAR(3) [FK to Airport.IATACode])

{Date, FlightNumber, DeptTime} ->{ ArrvTime, AirlineName, ActualDeptTime, ActualArrvTime, DeptAirportID, ArrvAirportID }

Non-primary key attributes aren't implied by other non-primary key attributes.

Schema is in 3NF because ArrvTime, AirlineName, ActualDeptTime, ActualArrvTime, DeptAirportID, and ArrvAirportID are dependent on FlightNumber, Date, and DeptTime and there are no transitive dependencies.

Status (Stores flight status)

Status(Date: DATE [PK & FK to Flight.Date], DeptTime: INT [PK & FK to Flight.DeptTime], FlightNumber: VARCHAR(10) [PK & FK to Flight.FlightNumber], FlightStatus: VARCHAR(20))

{Date, DeptTime, FlightNumber} -> {FlightStatus}

Schema is in 3NF because FlightStatus is dependent on Date, DeptTime, and FlightNumber and there are no transitive dependencies.

PASSENGER (Stores passenger information)

Passenger(Email: VARCHAR(100) [PK], FirstName: VARCHAR(50), LastName: VARCHAR(50))

{Email} -> {FirstName, LastName}

Schema is in 3NF because FirstName, LastName are dependent on Email and there are no transitive dependencies.

BOOKING (Stores booking information)

Booking(BookingId: INT [PK], FlightDate: DATE [FK to Flight.Date], FlightDeptTime: INT [FK to Flight.DeptTime], FlightNumber: VARCHAR(10) [FK to Flight.FlightNumber], Email: VARCHAR(100) [FK to Passenger.Email], BookingDate: DATE, Status: VARCHAR(20))

{BookingId} -> {FlightDate, FlightDeptTime, FlightNumber, Email, BookingDate, Status}

Schema is in 3NF because FlightDate, FlightDeptTime, FlightNumber, Email, BookingDate, and Status are dependent on BookingId and there are no transitive dependencies.

Cardinality:

Entity	**Related Entity**	**Cardinality**	**Explanation**
Airport	Flight	one-to-many	An airport may have multiple departing/arrival flights, but each flight departs from/arrives at only one airport.
Flight	Passenger	many-to-many	A flight may be booked by several passengers and each passenger may book many flights.
Flight	Status	one-to-one	Each flight can only be associated with one status at a time.