SKYHUB

By Kirill Grichanichenko and Artem Grichanichenko

OVERVIEW

Why?

- To design and implement a database system that manages flights, airports, customers, tickets, and bookings.
- Create a simple airline reservation systems that makes retrieving flight/ booking information efficient.

Who?

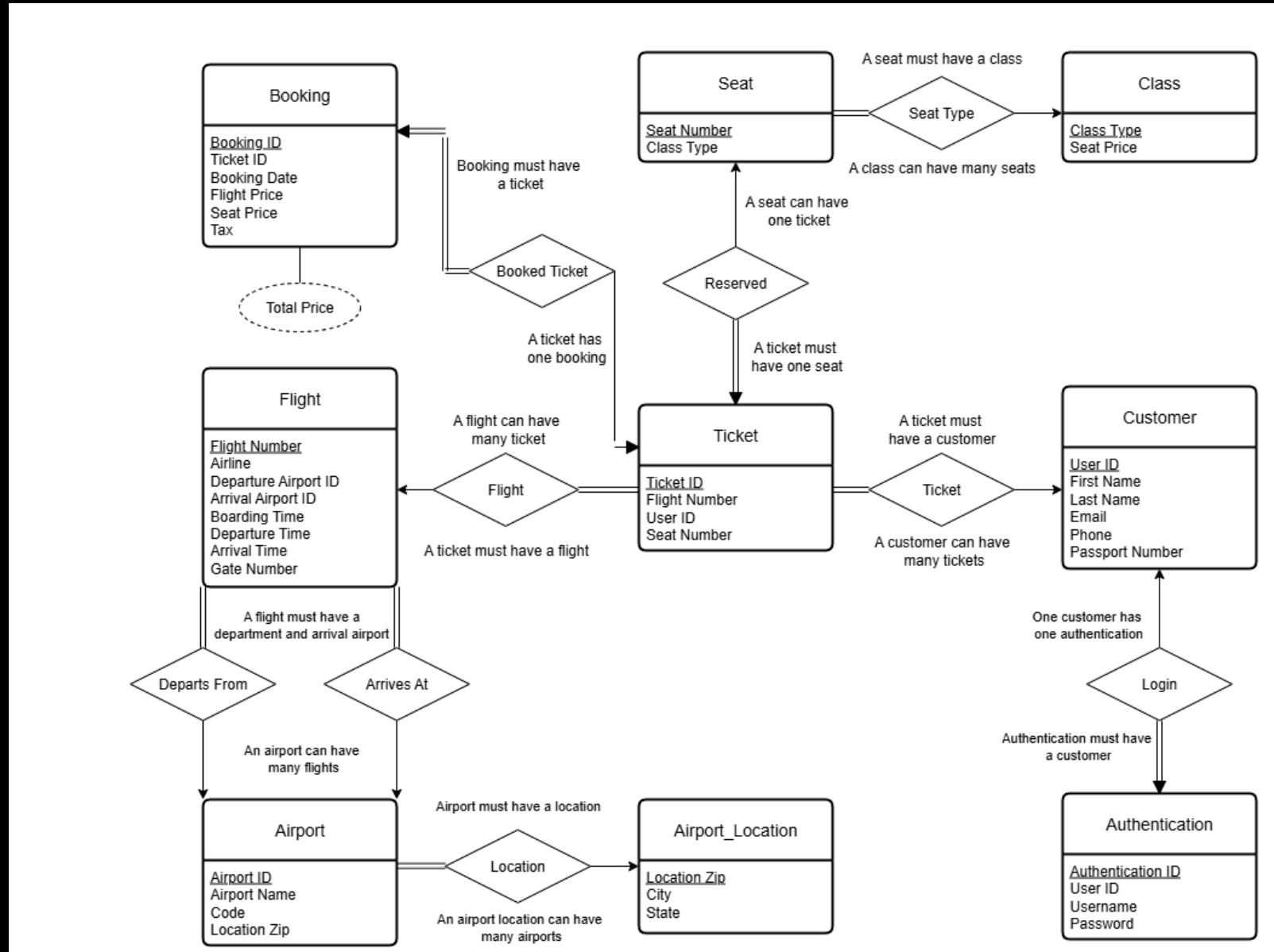
- **Airline staff** to manage flights, customers, tickets, and bookings
- **Customers** to view available flights, reserve seats, and check booking details
- **Developers/admins** to extend and improve the system

Features

- Stores airports, flights, seats, and classes
- Manage customer authentication
- Allow booking of flights and seat reservation
- Price management
- Filter flights by location
- View available/booked seats
- View booking details
- Cancel a booked flight

ENTITY RELATIONSHIP DIAGRAM

- **Core Entities:** Customer, flight, ticket, for the main airline booking process.
- **Design Choices:** Build around three main entities: Ticket, Flight, and Customer. All other entities build upon those.
- **Relationships:** Ticket is the connecting entity between the customer and flight relationship

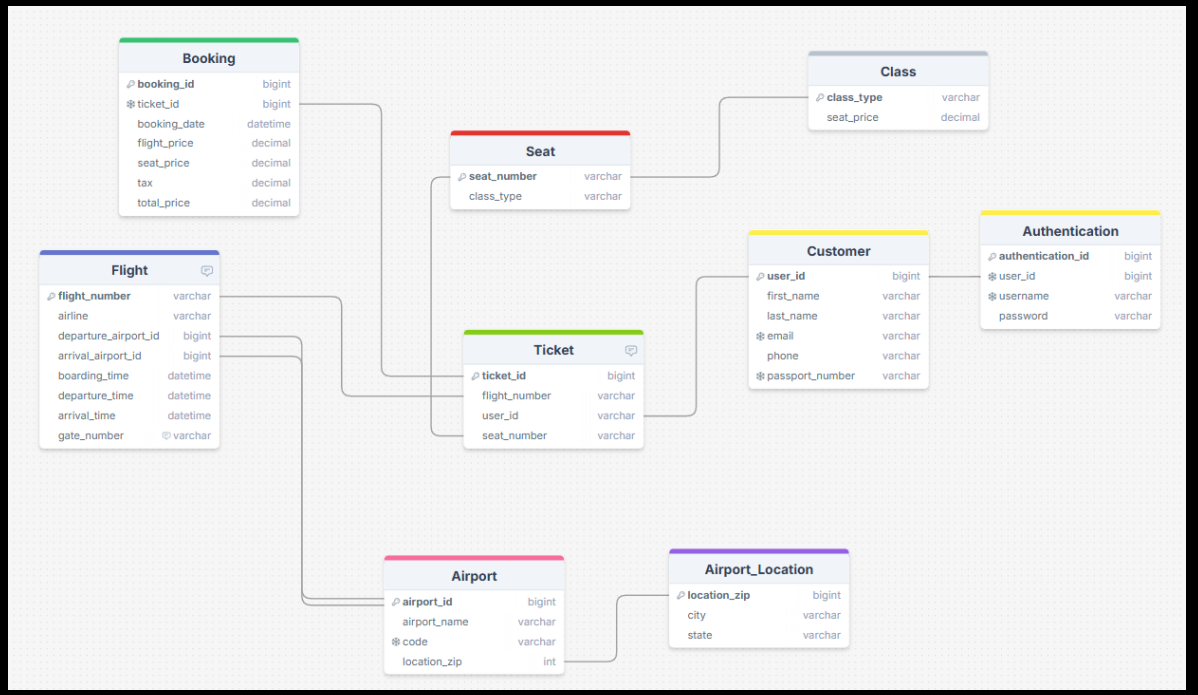


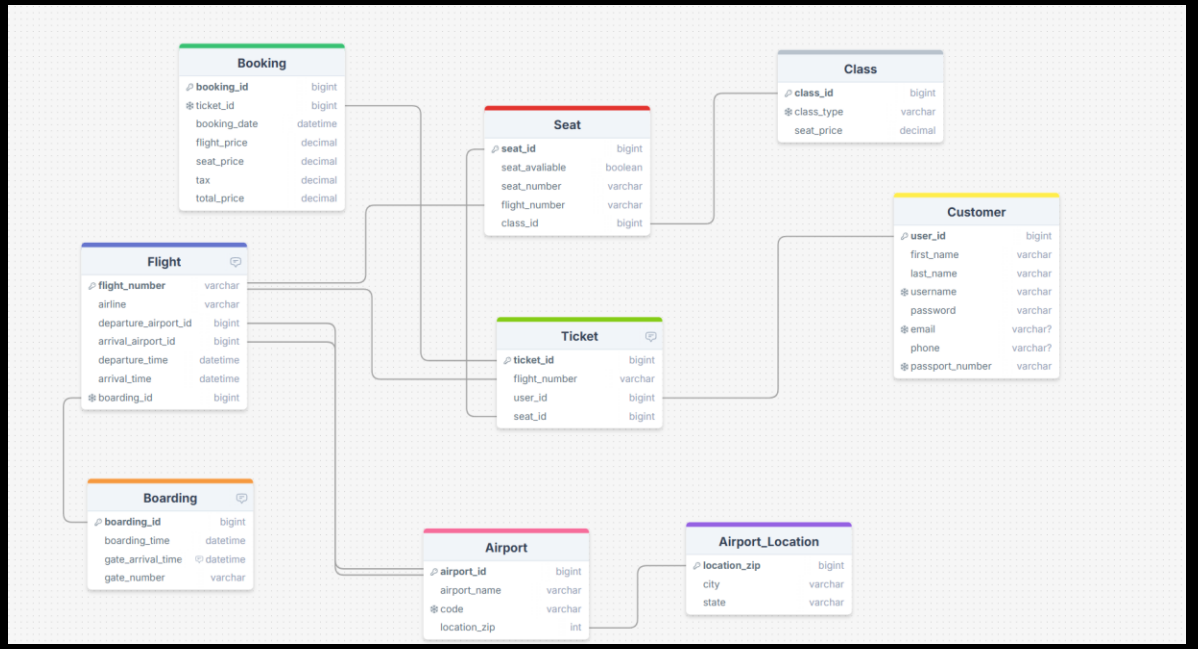
Phase 3 -

RELATIONAL SCHEMA

Major Changes From Phase 2

- Created an authentication table to hold customer authentication information separately from the customer table.
- Combined boarding information directly into the flight table.
- Complete seat entity restructure.
  - **Prev** – includes Boolean data type to manage seat availability on specific flights (requires a lot of inconvenient dynamic data management).
  - **Current** – includes seat table as a general representation of seats on a flight (reduces complex data management by putting the management responsibility on the ticket table).





Phase 2 -

NORMALIZATION (PART 1/2)

Ticket Relation:

- Ticket has the following functional dependencies:
  - Ticket id - candidate key which identifies all attributes within the ticket table.
  - Flight number & Seat number - super key which can uniquely identify the ticket id and user id.
- Therefore – since all determinants in the functional dependencies are super keys the relation is in BCNF.

Ticket

ticket_id	flight_number	user_id	seat_number

FD7:

{ticket_id} -> {flight_number, user_id, seat_number}

{flight_number, seat_number} -> {ticket_id, user_id}

Therefore: ticket_id and flight_number with seat_number are super keys that can be used to determine all attributes within the table

Proof: Since all determinants are super keys, Ticket is in BCNF

NORMALIZATION (PART 2/2)

Flight Relation:

- Flight has the following functional dependencies:
  - Flight number - candidate key which identifies all attributes within the flight table.
  - Departure airport & Gate number - super key which can uniquely identify all attributes within the flight table.
- Therefore – since all determinants in the functional dependencies are super keys the relation is in BCNF.

Flight

flight_number	airline	departure_airport_id	arrival_airport_id	boarding_time	departure_time	arrival_time	gate_number

FD5:

{flight_number} -> {airline, departure_airport_id, arrival_airport_id, boarding_time, departure_time, arrival_time, gate_number}

{departure_airport_id, gate_number} -> {flight_number, airline, arrival_airport_id, boarding_time, departure_time, arrival_time}

Therefore: flight_number and departure_airport with gate_number are determinants that can identify all attributes within the Flight table.

Conclusion

All relations in the database follow the same pattern of all determinates being either candidate or super keys and therefore proving to be in BCNF.

PROTOTYPE / DEMO OVERVIEW

Four Required Queries From Phase II Part C

Query 1/4
Purpose: Show all flights at an airport, including its name and code.
Expected: List of flight numbers, airline names, and departure airport details.
Query:

```
SELECT
    flight.flight_number,
    flight.airline,
    departure_airport.code AS
departure_airport_code,
    arrival_airport.code AS
arrival_airport_code,
    flight.boarding_time,
    flight.departure_time,
    flight.arrival_time,
    flight.gate_number
    FROM flight
    JOIN airport departure_airport
        ON flight.departure_airport_id =
departure_airport.airport_id
    JOIN airport arrival_airport
        ON flight.arrival_airport_id =
arrival_airport.airport_id
    WHERE departure_airport.airport_id = ?;
```

Query 2/4
Purpose: List available seat numbers for given flight.
Expected: List of seat numbers that are still available for the given flight.
Query:

```
SELECT seat.seat_number
    FROM seat
    WHERE seat.seat_number NOT IN
(
        SELECT ticket.seat_number
        FROM ticket
        WHERE ticket.flight_number = ?
    )
    ORDER BY seat.seat_number;
```

Query 3/4
Purpose: Show all bookings made by user.
Expected: Booking ID, total price, and booking date.
Query:

```
SELECT
    ticket.ticket_id,
    ticket.flight_number,
    departure_airport.code AS
departure_airport,
    arrival_airport.code AS arrival_airport,
    seat.seat_number,
    class.class_type
  FROM ticket
    JOIN seat ON ticket.seat_number =
seat.seat_number
    JOIN class ON seat.class_type =
class.class_type
    JOIN flight ON ticket.flight_number =
flight.flight_number
    JOIN airport AS departure_airport ON
flight.departure_airport_id =
departure_airport.airport_id
    JOIN airport AS arrival_airport ON
flight.arrival_airport_id =
arrival_airport.airport_id
  WHERE user_id = ?
ORDER BY ticket.ticket_id ;
```

Query 4/4
Purpose: Get all flight details of a given flight including the airport and airport location of the flight
Expected: List of flights with their flight information, boarding details, and airport with location details.
Query:

```
SELECT
    flight.flight_number,
    flight.airline,
    airport.code AS departure_airport,
    arrival_airport.code AS arrival_airport,
    flight.boarding_time,
    flight.departure_time,
    flight.arrival_time,
    flight.gate_number,
    airport.airport_name,
    airport_location.city,
    airport_location.state,
    airport_location.location_zip
  FROM flight
    JOIN airport ON
flight.departure_airport_id =
airport.airport_id
    JOIN airport_location on
airport.location_zip =
airport_location.location_zip
    JOIN airport AS arrival_airport ON
flight.arrival_airport_id =
arrival_airport.airport_id
  WHERE flight_number = ?
```

CHALLENGES AND FUTURE WORK

Challenges

- Designing the overall database structure of the application.
- Managing data integrity with correct delete and update operations as well as creation order.
- Implementing triggers to automatically calculate derived attributes (e.g. booking prices)

Future Work

- Expand to international airports
- Add more dynamic pricing on flights calculated through distance
- Add roles to the authentication system with an admin to manage flights and airports through CRUD operations
- Add account balance to customers and customer info page.

THANK YOU!