ITCS413: Database Design

Project 3: Logical Database Design

Airline: Star Airline

(March 2025)

Prepared by

6588070 Nakarin Phoorahong

6588096 Panipak Sittiprasert

6588183 Achiraya Mankham

Presented To

Asst.Prof. Dr. Charnyote Pluempitiwiriyawej

Faculty of Information and Communication Technology

Mahidol University

Table of contents

Overview

In this phase, we are revising the reports which include our own scope of project, entity relationship diagram and provide relationship schema along with integrity constraints based on the updated version of entity relationships diagram. After consultation on the project in the previous class, we decided to trim the airline personnel, automated revenue and auditing reporting, security compliance monitoring and cargo perspective out of scope of the project. So, that does make us have to update on the mapping of major user views and operations, user requirement specifications section and system requirements specification to update information to make inline with the current revised scope. From now on, we will focus entirely on **bookings and flights management** from the airline-side. Also, system requirements have been corrected due to the oversetting system requirement in project phase one.

In summary, this phase we have modified the mapping of major user view and operations, user requirement specifications section, system requirements specification and entity-relationship diagram and we also create relationship schema along with its integrity and other constraints.

Mappings of Major user views and operations

- Based on the scope, we can divide the user group into the following categories.
- Passengers and Airline Personnel.
- The picture below shows the overview picture of the relationship between user groups and data. Each user group is compartmentalized to access data that are needed for their operation and transaction only.

Data/Users	**Passengers**
Passengers Data	Read, write, modify, delete
Flight Data	Read (Limited)
Reservation Data	Read, modify, delete
Aircraft Data	Read (limited)
Seat	Read
CheckIn	Read, write
Baggage	Read, write
FlightSegment	Read
Payment	Read, write

User Requirements Specifications

Currently based on the previous section, we have identified users to be passengers, airline personnel, security officers, the Finance and Audit team, and Cargo Operators as a different group of users.

Data Requirements

The data required for efficient passenger, employees and cargo administration must be handled effectively by the airline database system. The primary entities and attributes include:

- **Passengers:** It is passenger information which include attributes such as PassengerID as a primary key, fullname, phone number, email.
- **Bookings:** Manages reservations, linking passengers to flights and passengers to checkIn. It has BookingID as a sole attribute and primary key.
- **Flights:** Flight information which includes attributes such as FlightID as a primary key, depAirport(DepartureAirport), arrAirport(ArrivalAirport), stops(Number of stops on the flight) as attributes.
- **Flight Segment:** Defines different seat classes (Economy, Business, First Class) with pricing and availability. It has ClassID as a primary key , ClassType, Price, SeatAvailability as attributes.
- **Seat:** Seat Number to each specific booking and passengers. It only has seatID as primary key and it bridges between passenger and booking by linking the creates.
- **Check-in:** Tracks passenger check-in status. It has checkInID as a primary key, checkInTime, and checkInLocation as attributes.
- **Payments:** Stores financial transaction details. It has paymentID as a primary key, amount, PaymentDate, paymentMethod, and paymentStatus as attributes.
- **Baggage:** Store Baggage details of each checkIn that have been made. It has baggageID as a primary key, and baggageType, size, weight as attributes.
- **Airbus:** store type of airplane. It has airbutsID as a primary key, modelName, economySeats (Number of Economy seat), businessSeats (Number of business seat), and firstClassSeats (Number of first class seat) as attributes.

Transaction Requirements

The airline database system must support essential operations such as insert, delete, update, and look-up/search queries to ensure smooth and efficient functionality.

Data Insertion or Data Entry

Data insertion can be performed by different user groups based on their roles and access privileges. While Airline personnel can read, write, modify and delete from almost every table.

- The insertion of new details on Passengers(such as passengerID, fullName, email, phoneNum).
- The insertion of new details on Booking(such as bookingID).
- The insertion of new details on Flights (such as flightID, depAirport, arrAirport, stops)
- The insertion of new details on Flight segment (such as segmentID, fromAirport, toAirport, depTime, arrTime)
- The insertion of new details on Seat (such as seatNoID).
- The insertion of new details on CheckIn (such as checkInID, checkInLocation, checkInTime)
- The insertion of new details on Payments (paymentID, amount, paymentMethod, paymentStatus, paymentDate)
- The insertion of new details on Baggage (such as baggageID, weight, size, baggageType)
- The insertion of new details on Airbus (such as airbusID, modelName, economySeats, businessSeats, firstClassSeats)

Data Deletion/Update

Authorized users can delete or modify data within their permitted domains.

- The delete or update of details on Passengers(such as passengerID, fullName, email, phoneNum).
- The delete or update of details on Booking(such as bookingID).
- The delete or update of details on Flights (such as flightID, depAirport, arrAirport, stops)
- The delete or update of details on Flight segment (such as segmentID, fromAirport, toAirport, depTime, arrTime)
- The delete or update of details on Seat (such as seatNoID).

- The delete or update of details on CheckIn (such as checkInID, checkInLocation, checkInTime)
- The delete or update of details on Payments (paymentID, amount, paymentMethod, paymentStatus, paymentDate)
- The delete or update of details on Baggage (such as baggageID, weight, size, baggageType)
- The delete or update of details on Airbus (such as airbusID, modelName, economySeats, businessSeats, firstClassSeats)

Data Queries

The system must provide query capabilities to retrieve and analyze information from various databases and tables.

This table shows an example of queries that typical airlines usually do have.

Query Type	Example
Basic Lookup	Find all flights, passengers, baggage, payment, booking, and checkIn information with specific conditions such as time and location or specified attributes.
Aggregate Query	Count total passengers on a specific flight. Count the number of flights in a specific time. Count total baggage per flight in specific condition. Count total transaction from payment in 1 month period. Find the total revenue from passenger' booking in a specific time period.

Systems Requirements Specification

A lot of information in this section is referenced in the conversation with ChatGPT throughout several prompts. https://chatgpt.com/share/67a0687e-b80c-8000-a923-0e2f07e6beeb

Initial database size and Database growth

The estimated number of passengers for Star Airlines each year is around 1,500,000.

This is an example estimation of database size that considers each component.

Component	Records per year	Avg. Record Size	Yearly Growth	Total Size
Passenger Data	500 K	5 KB	5-10%	2.44 GB
Flight Data	33 K	10 KB	3-5%	0.32 GB
Booking Data	666.667	8 KB	5-10%	5.08 GB
Aircraft Data (Airbus)	1000	15 KB	3%	0.01 GB
Seat Data	166,667	2 KB	6%	0.32 GB
CheckIn	500 K	3 KB	5%	1.46 GB
Baggage Data	400 K	6 KB	8%	2.29 GB
Flight Segment Data	133,333	12 KB	4%	1.53 GB
Payment	666,667	4 KB	6%	2.61 GB

Based on ChatGPT's estimation:

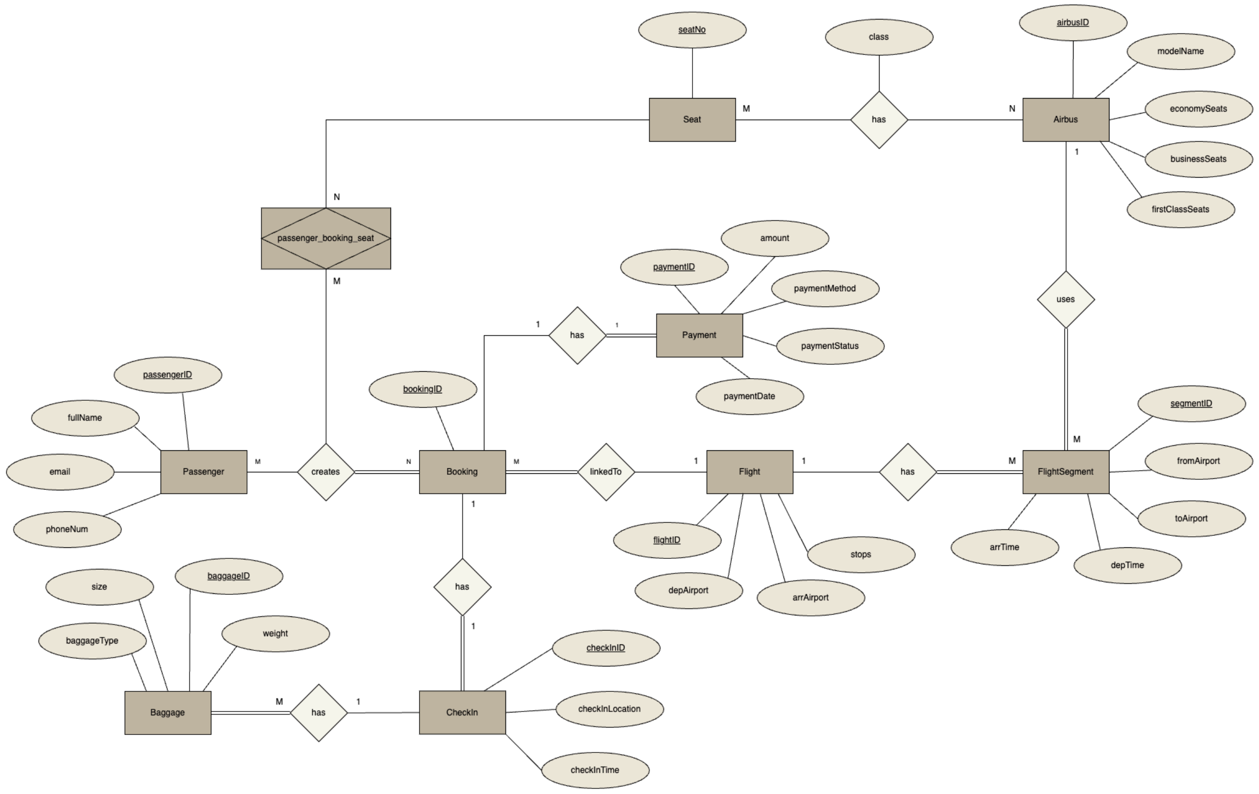
Result in a total initial database size of 16.77 TB for the first year.

We agree with ChatGPT that at least in each year growth should be around 5-10%, so let's say that average growth rate is 5.44% annually.

Year	Total Estimated Database Size
Year 1	16.77 TB
Year 2	17.86 TB
Year 3	18.65 TB
Year 4	19.66 TB
Year 5	20.73 TB

ER diagram

Conceptual Data Model



Link:
https://drive.google.com/file/d/1tmD7hiHS1Di6IlUNfKhzCRHeIveDfJsN/view?usp=sharing

Entity

1. Passenger

- passengerID (PK) – Unique identifier for the passenger.
- fullName – Full name of the passenger.
- email – Email of the passenger.
- phoneNum – Contact number of the passenger.

2. Booking

- bookingID (PK) – Unique booking identifier.

3. Payment

- paymentID (PK) – Unique payment identifier.
- amount – Total amount paid for the booking.
- paymentMethod – The method used for payment.
- paymentStatus – Status of the payment.
- paymentDate – The date when payment was made.

4. Flight

- flightID (PK) – Unique flight identifier.
- depAirport – Departure airport of the flight.
- arrAirport – Arrival airport of the flight.
- stops – Number of stops on the flight.

5. FlightSegment

- segmentID (PK) – Unique identifier for each segment of a flight.
- fromAirport – Departure airport of this segment.
- toAirport – Arrival airport of this segment.
- depTime – Departure time of the segment.
- arrTime – Arrival time of the segment.

6. Airbus

- airbusID (PK) – Unique identifier for the aircraft.
- modelName – Name of the aircraft model (e.g., Boeing 747, Airbus A380).
- economySeats – Number of economy class seats.
- businessSeats – Number of business class seats.
- firstClassSeats – Number of first-class seats.

7. Seat

- seatNo (PK) – Unique seat number.

8. CheckIn

- checkInID (PK) – Unique identifier for the check-in process.
- checkInLocation – Location where check-in occurred.
- checkInTime – Timestamp of the check-in.

9. Baggage

- baggageID (PK) – Unique identifier for baggage.
- weight – Weight of the baggage.
- size – Size category (Small, Medium, Large).
- baggageType – Type of baggage (e.g., carry-on, checked baggage).

Relationships and Foreign Keys

1. Passenger creates Booking (M:N)

- A Passenger can create multiple Bookings.
- A Booking can belong to multiple Passengers (group bookings).

PassengerBooking linking table:

- passengerID (FK) – References Passenger.
- bookingID (FK) – References Booking.

2. Booking has Payment (1:1)

- Each Booking has exactly one Payment.
- A Payment is linked to a single Booking.
- paymentID (FK) – References Payment.

3. Booking is linked to Flight (M:1)

- A Flight can be booked by multiple Bookings.
- flightID (FK) – References Flight.

4. Flight has FlightSegment (1:M)

- A Flight consists of multiple FlightSegments (for layovers).
- A FlightSegment belongs to exactly one Flight.
- flightID (FK) – References Flight.

5. FlightSegment uses Airbus (M:1)

- Each FlightSegment is operated by an Airbus.
- An Airbus can be used in multiple FlightSegments.
- airbusID (FK) – References Airbus.

6. Airbus has Seat (M:N)

- An Airbus has multiple Seats.
- Each Seat belongs to an Airbus.

7. CheckIn has Baggage (1:M)

- Each CheckIn can have multiple pieces of Baggage.
- Each Baggage belongs to one CheckIn.
- checkInID (FK) – References CheckIn.

8. Booking is assigned to Passenger (M:N)

- A Passenger can have multiple Bookings.
- A Booking can have multiple assigned Passengers.

Changes from the Previous Version

- The Booking entity continues to exist, but now it establishes clearer relationships with entities like Payment, Flight, and Passenger.
- The Booking ↔ Passenger relationship is better defined through the PassengerBooking table to allow a booking to be assigned to multiple passengers, facilitating family and group bookings.
- The Payment entity includes a new attribute, PaymentDate, which records the date when payment was made.

- The relationship between Booking and Payment was formalized as a 1:1 relationship, ensuring that every Booking is linked to exactly one Payment.
- The Flight entity has been enhanced to include more detailed attributes, including departure and arrival airports, as well as the number of stops.
- A new FlightSegment entity was introduced to better model the different segments of a flight. Each Flight can now be associated with multiple FlightSegments, and each FlightSegment corresponds to a specific segment of the flight (e.g., a layover or direct segment).
- Flight ↔ FlightSegment: One flight can consist of multiple segments, and each segment is part of exactly one flight.
- FlightSegment ↔ Airbus: Each flight segment is operated by an aircraft (Airbus), creating a many-to-many relationship between FlightSegment and Airbus.
- The Airbus entity now includes additional attributes such as the number of economy, business, and first-class seats available on the aircraft. This allows for more granular control over seat availability and better modeling of class-specific features.
- The CheckIn entity now includes attributes such as CheckInLocation and CheckInTime, and it is explicitly linked to the Booking entity.
- Booking ↔ CheckIn: One Booking can only perform one CheckIn, and each CheckIn is linked to exactly one Booking.
- CheckIn ↔ Baggage: A CheckIn can have multiple pieces of baggage associated with it, ensuring baggage tracking is tied to the check-in process.

Relation Schemas and Constraints

List of Relation Schemas

passenger (passengerID, fullName, email, phoneNum)

booking (bookingID)

passenger_booking (passengerID, bookingID)

payment (paymentID, amount, paymentMethod, paymentStatus, paymentDate, bookingID)

flight (flightID, depAirport, arrAirport, stops)

booking_flight (bookingID, flightID)

flightSegment (segmentID, fromAirport, toAirport, depTime, arrTime, airbusID, flightID)

airbus (airbusID, modelName, economySeats, businessSeats, firstClassSeats)

seat(seatNo)

airbus_seat (seatNo, airbusID, class)

checkIn (checkInID, checkInLocation, checkInTime, bookingID)

baggage (baggageID, weight, size, baggageType, checkInID)

passenger_booking_seat (passengerID, bookingID, seatNo)

- **passenger_booking:** This linking table manages the many-to-many relationship between Passengers and Bookings. A single booking can have multiple passengers, and a passenger can make multiple bookings.
- **airbus_seat**: This linking table associates seats with specific aircraft and their class types. It allows for efficient seat management across different aircraft.
- **passenger_booking_seat:** It acts as a bridge between the ‘creates’ relationship (Passenger_Booking) and Seat. It ensures that each passenger, for a specific booking, is assigned a specific seat.

Integrity Constraints

Domain constraints

1. Email (passenger.email): The email field in the passenger table should follow a standard email format (e.g., abc@gmail.com). It must contain valid characters and conform to common email validation rules.
2. Phone Number (passenger.phoneNum): The phoneNum field should store valid phone numbers, adhering to specific formats depending on the region (e.g., a 10-digit number or an international format).
3. Amount (payment.amount): The amount attributed in the payment table must be a positive numeric value representing the total payment amount.
4. Stops (flight.stops): The stops attribute in the flight table must be a non-negative integer (e.g., 0 for direct flights, 1 for one stop, etc.).
5. Weight (baggage.weight): The weight attribute in the baggage table should be a positive numeric value representing the weight of the baggage, typically in kilograms or pounds.
6. Size (baggage.size): The size attribute in the baggage table should have predefined values like Small, Medium, or Large.
7. Class (airbus_seat.class): The class attribute in the airbus_seat table should have predefined values such as Economy, Business, or First Class.

Uniqueness constraints

1. passengerID (passenger.passengerID): The passengerID is unique for each passenger and acts as the primary key in the passenger table. No two passengers can have the same passengerID.
2. email (passenger.email): The email attribute must be unique for each passenger, meaning that no two passengers can share the same email address.
3. bookingID (booking.bookingID): The bookingID is unique for each booking and serves as the primary key for the booking table. This ensures that each booking is uniquely identifiable.
4. paymentID (payment.paymentID): The paymentID is unique for each payment and acts as the primary key for the payment table.

5. flightID (flight.flightID): The flightID is unique for each flight and serves as the primary key in the flight table, ensuring that each flight is identifiable.
6. seatNo (seat.seatNo): The seatNo is unique for each seat and is the primary key in the seat table. This ensures that there are no duplicate seat numbers within the system.
7. segmentID (flightSegment.segmentID): The segmentID is unique for each flight segment and serves as the primary key in the flightSegment table.
8. airbusID (airbus.airbusID): The airbusID is unique for each aircraft and serves as the primary key in the airbus table.
9. checkInID (checkIn.checkInID): The checkInID is unique for each check-in and serves as the primary key in the checkIn table.
10. baggageID (baggage.baggageID): The baggageID is unique for each baggage item and serves as the primary key in the baggage table.

Not null constraints

1. passengerID (passenger.passengerID): This attribute cannot be null, as it uniquely identifies each passenger.
2. fullName (passenger.fullName): A passenger must always have a fullName, so this attribute cannot be null.
3. email (passenger.email): Every passenger must have an email, and this field cannot be left blank.
4. bookingID (booking.bookingID): Every booking record must have a bookingID, which cannot be null.
5. paymentID (payment.paymentID): Each payment record must be associated with a unique paymentID, which cannot be null.
6. amount (payment.amount): Every payment must have a valid amount, and this field cannot be null.
7. paymentMethod (payment.paymentMethod): The paymentMethod (e.g., credit card, PayPal) must be specified for each payment and cannot be null.
8. flightID (flight.flightID): Each flight record must have a unique flightID and cannot be null.

9. depAirport (flight.depAirport): The departure airport must be specified for each flight and cannot be null.
10. arrAirport (flight.arrAirport): The arrival airport must be specified for each flight and cannot be null.
11. segmentID (flightSegment.segmentID): Each flightSegment must have a segmentID, and this attribute cannot be null.
12. airbusID (flightSegment.airbusID): Every flightSegment must reference an airbus and cannot have a null airbusID.
13. checkInID (checkIn.checkInID): Each check-in must have a unique checkInID, and it cannot be null.
14. checkInLocation (checkIn.checkInLocation): The checkInLocation must always be specified and cannot be null.
15. checkInTime (checkIn.checkInTime): The checkInTime must be specified for each check-in and cannot be null.
16. baggageID (baggage.baggageID): Every baggage record must have a unique baggageID, and this attribute cannot be null.
17. weight (baggage.weight): The weight of the baggage must always be provided and cannot be null.
18. size (baggage.size): The size category must be specified for each baggage and cannot be null.

Multiplicity

1. Passenger ↔ Booking (M:N):
   - A Passenger can create multiple Bookings (e.g., booking multiple flights), and a Booking can belong to multiple Passengers (e.g., group bookings).
     Managed by the passenger_booking table.
2. Booking ↔ Payment (1:1):
   - A Booking has exactly one Payment, and each Payment corresponds to one Booking.

3. Booking ↔ Flight (M:N):
    - A Booking can be linked to multiple Flights (e.g., round trips or connecting flights), and a Flight can be booked by multiple Bookings.
      Managed by the booking_flight table.
4. Flight ↔ FlightSegment (1:M):
    - A Flight consists of multiple FlightSegments (for layovers or different parts of the journey), but each FlightSegment belongs to only one Flight.
5. FlightSegment ↔ Airbus (M:1):
    - A FlightSegment is operated by one Airbus, but an Airbus can operate multiple FlightSegments.
6. Airbus ↔ Seat (M:N):
    - An Airbus has multiple Seats, and each Seat belongs to one Airbus.
      Managed by the airbus_seat table.
7. CheckIn ↔ Baggage (1:M):
    - A CheckIn can be associated with multiple Baggage items, but each Baggage belongs to one CheckIn.
8. Passenger ↔ Seat (M:N):
    - A Passenger can be assigned multiple Seats within different Bookings, and each Seat can be assigned to multiple Passengers for different Bookings.
      Managed by the passenger_booking_seat table.

References

[1] “What if the number of passenger data were all go down to 500,000 passengers per year, adjusting other data so that it correlates to the decrease number of passengers. Recalculate everything through again and put it out in nice table format ” prompt. *ChatGPT4o*, 11 Mar. version, OpenAI, Mar. 2025, chat.openai.com.