ISE 503: Data Management

Course Project Report

Travel Agency System

Submitted by:

Praveen Anand

Student ID: 116061610

Spring 2025

Project Report: Travel Agency System

1. Project Overview

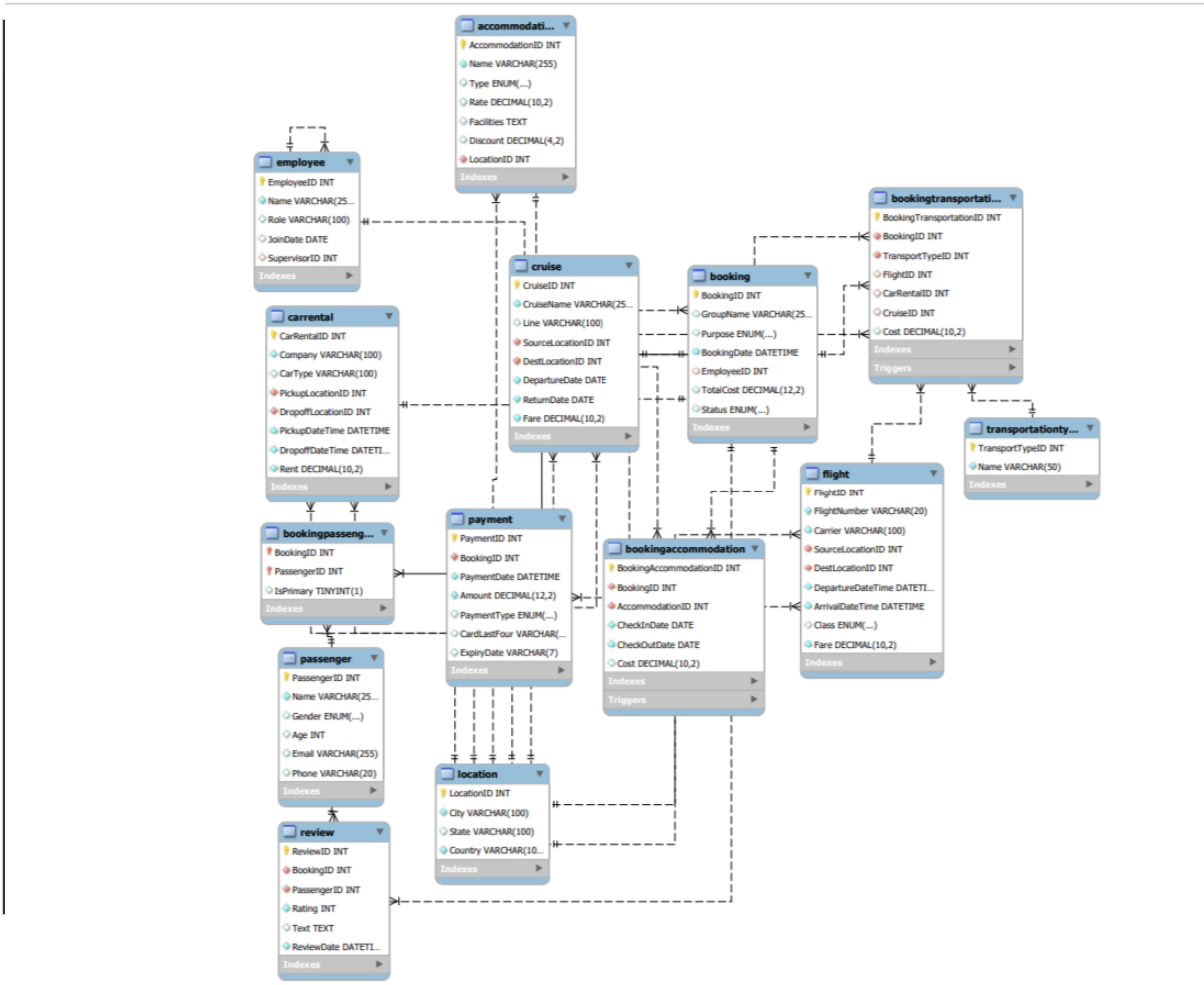
This report details the Travel Agency Database System developed as part of the ISE 503: Data Management course project for Spring 2025. The project involved designing and implementing a relational database system to model real-world travel agency data, supporting common transaction operations. The system utilizes a MySQL database and includes a basic web application interface built with Flask (Python) for demonstrating database interactions and transaction functionalities.

The primary goal was to create a robust schema grounded in sound data modeling practices and to implement key transaction features as requested, specifically adding passengers, listing available travel options (hotels and flights), and viewing detailed passenger itineraries, without altering the pre-existing database schema structure.

2. Database Schema and EER Diagram

2.1 EER Diagram

The following Enhanced Entity-Relationship (EER) diagram visually represents the structure of the travel agency database, illustrating the entities, their attributes, and the relationships between them.



2.2 Schema Summary

The database models a travel agency system, managing information about locations, passengers, employees, accommodations, various transportation modes (flights, car rentals, cruises), bookings, payments, and reviews. The schema is implemented in MySQL using the InnoDB engine to support relational integrity through foreign keys.

Entities and Relationships:

- **Location**: Stores geographical information (City, State, Country). Referenced by Accommodation, Flight, CarRental, and Cruise tables. (PK: `LocationID`)
- **Passenger**: Stores customer details (Name, Gender, Age, Email, Phone). Linked to Booking via `BookingPassenger` and to Review. (PK: `PassengerID`)
- **Employee**: Stores staff information (Name, Role, JoinDate), including a self-referencing `SupervisorID`. Linked to Booking. (PK: `EmployeeID`)
- **Accommodation**: Represents lodging options (Name, Type, Rate, Facilities, Location). Linked to Location and Booking via `BookingAccommodation`. (PK: `AccommodationID`)
- **TransportationType**: Lookup table for transport modes (Flight, Car Rental, etc.). (PK: `TransportTypeID`)
- **Flight**: Details flight segments (Number, Carrier, Source/Destination, Times, Class, Fare). Linked to Location and `BookingTransportation`. (PK: `FlightID`)
- **CarRental**: Details car rentals (Company, Type, Pickup/Dropoff, Rent). Linked to Location and `BookingTransportation`. (PK: `CarRentalID`)
- **Cruise**: Details cruise trips (Name, Line, Source/Destination, Dates, Fare). Linked to Location and `BookingTransportation`. (PK: `CruiseID`)
- **Booking**: Represents a travel package booking (Group, Purpose, Date, Agent, Cost, Status). Central entity linked to Passengers, Accommodations, and Transportations. (PK: `BookingID`)
- **BookingPassenger**: Junction table for the M:N relationship between Booking and Passenger. (Composite PK: `BookingID`, `PassengerID`)
- **BookingAccommodation**: Junction table linking Booking with Accommodation, storing dates and cost. (PK: `BookingAccommodationID`)
- **BookingTransportation**: Junction table linking Booking with specific transportation details (Flight, CarRental, Cruise), using `TransportTypeID`. (PK: `BookingTransportationID`)
- **Payment**: Records payment details for a Booking. (PK: `PaymentID`)
- **Review**: Allows passengers to review a Booking. (PK: `ReviewID`)

Additional Schema Features:

- **Indexes**: Implemented on various columns to enhance query performance.
- **Views**: `PopularDestinations` and `EmployeePerformance` provide summarized insights.
- **Stored Procedure**: `GenerateMonthlyReport` facilitates reporting.
- **Triggers**: `CalculateBookingTotal` and `CalculateBookingTotalTransport` automatically update booking costs.

This relational structure provides a comprehensive foundation for managing travel agency data and operations.

3. Implemented Transaction Operations

The following common transaction operations were implemented within the Flask web application layer, interacting with the existing database schema without modifying its structure:

3.1 Add Passenger

- **Purpose**: Allows agency staff to add new passengers to the database.
- **Interface**: A web form at `/add_passenger` collects Name, Gender, Age, Email, and Phone.
- **Backend Logic**: Handles form submission, performs validation, executes an `INSERT` statement into the `Passenger` table using direct MySQL connection, handles errors (e.g., duplicate email), provides feedback, and redirects upon success.
- **Schema Interaction**: Inserts new rows into the `Passenger` table.

3.2 List Available Hotels (Accommodations) and Flights

- **Purpose**: Provides an overview of available lodging and flight options.
- **Interface**: A page at `/available_options` displays separate tables for accommodations and flights.
- **Backend Logic**: Fetches data using `SELECT *` queries on the `Accommodation` and `Flight` tables via a helper function and passes it to the template.
- **Schema Interaction**: Reads data from the `Accommodation` and `Flight` tables.

3.3 View Passenger Itinerary

- **Purpose**: Displays the complete travel history and upcoming trips for a selected passenger.
- **Interface**: Accessed via a button on the passenger list (`/passengers`), leading to a dedicated page (`/passenger_itinerary/<passenger_id>`). Shows passenger details and a breakdown of each associated booking, including accommodation and transportation segments.
- **Backend Logic**: Retrieves passenger details and executes a complex multi-table `JOIN` query to gather all itinerary components (bookings, accommodations, flights, car rentals, cruises, locations). Processes and structures the results before passing them to the template. Handles errors like invalid passenger IDs.
- **Schema Interaction**: Reads data extensively across multiple tables, utilizing foreign key relationships to construct the complete itinerary view.

These operations enhance the functionality of the travel agency application by providing essential tools for managing passenger information and viewing travel arrangements.

4. Conclusion

The project successfully implements a relational database for a travel agency system, adhering to the specified requirements. The schema effectively models the core entities and relationships involved in travel booking. Furthermore, key transaction operations requested by the user—adding passengers, listing available options, and viewing detailed itineraries—were integrated into the web application layer without altering the established database structure, demonstrating the system's capability to support common operational tasks.