CS306 - Web-Integrated Database Project: Final Report

Group Name: CS306_GROUP_74
Group Members:

- Ahmet Çavuşoğlu - 32394
- Yunus Emre Gök - 32028
- Ufuk Çimen - 32672

---

1. Introduction

This project presents the final phase of a flight tracking database system for a government institution. The system tracks flights, passengers, aircrafts, baggage, crew members, and supports administrative control via triggers and stored procedures using MySQL. Additionally, a support system for users is developed using MongoDB, enabling real-time feedback, commenting, and resolution tracking.

---

2. Triggers

2.1 CancelFlightIfNoTickets

- **Purpose:** Automatically cancels a flight when all its tickets are deleted.
- **Trigger Type:** AFTER DELETE ON Ticket
- **Logic:** If no tickets remain for a flight, its status is updated to 'Cancelled'.
- **SQL Script:**

```
CREATE TRIGGER CancelFlightIfNoTickets
AFTER DELETE ON Ticket
FOR EACH ROW
BEGIN
    IF NOT EXISTS (SELECT * FROM Ticket WHERE flight_id = OLD.flight_id) THEN
        UPDATE Flight SET status = 'Cancelled' WHERE flight_id =
OLD.flight_id;
    END IF;
END;
```

- **Web Page & Case:** Triggered via ticket deletion operation in **trigger1.php.**

2.2 ReactivateFlightIfTicketAdded

- **Purpose:** Reactivates a flight previously cancelled, if a new ticket is added.
- **Trigger Type:** AFTER INSERT ON Ticket
- **Logic:** If a cancelled flight receives a ticket, its status is changed to 'On Time'.
- **SQL Script:**

```
CREATE TRIGGER ReactivateFlightIfTicketAdded
AFTER INSERT ON Ticket
FOR EACH ROW
BEGIN
    IF (SELECT status FROM Flight WHERE flight_id = NEW.flight_id) =
'Cancelled' THEN
        UPDATE Flight SET status = 'On Time' WHERE flight_id = NEW.flight_id;
    END IF;
END;
```

- **Web Page & Case:** Triggered when a new ticket is added in **trigger1_2.php.**

2.3 PreventOverbooking

- **Purpose:** Prevents selling more tickets than the aircraft capacity.
- **Trigger Type:** BEFORE INSERT ON Ticket
- **Logic:** Counts current tickets and checks against aircraft capacity. Aborts if overbooked.
- **SQL Script:**

```
CREATE TRIGGER PreventOverbooking
BEFORE INSERT ON Ticket
FOR EACH ROW
BEGIN
    DECLARE capacity INT;
    DECLARE sold_tickets INT;
    SELECT A.capacity INTO capacity
    FROM Flight F
    JOIN Aircraft A ON F.aircraft_id = A.aircraft_id
    WHERE F.flight_id = NEW.flight_id;

    SELECT COUNT(*) INTO sold_tickets
    FROM Ticket
    WHERE flight_id = NEW.flight_id;

    IF sold_tickets >= capacity THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Flight is fully booked!';
    END IF;
END;
```

- **Web Page & Case:** Triggered when attempting to insert a new ticket via **trigger1_3.php.**

2.4 AutoCalculateExtraBaggageFee

- **Purpose:** Calculates extra fee automatically if baggage weight exceeds 25kg.
- **Trigger Type:** BEFORE INSERT ON Baggage
- **Logic:** Fee = (weight - 25) * unit_fee if overweight, else 0.
- **SQL Script:**

```
CREATE TRIGGER AutoCalculateExtraBaggageFee
BEFORE INSERT ON Baggage
FOR EACH ROW
BEGIN
    IF NEW.weight > 25 THEN
        SET NEW.extra_fee = (NEW.weight - 25) * NEW.extra_fee;
    ELSE
        SET NEW.extra_fee = 0;
    END IF;
END;
```

- **Web Page & Case:** Trigger runs on baggage insert via **trigger1_4.php.**

---

3. Stored Procedures

3.1 GetPassengerFlights

- **Purpose:** Retrieves all flights of a specific passenger.
- **Parameters:** `input_passenger_id INT`
- **SQL Script:**

```
CREATE PROCEDURE GetPassengerFlights(IN input_passenger_id INT)
BEGIN
    SELECT F.flight_id, F.flight_number, A.name AS airline_name,
F.departure_time, F.arrival_time, P.f_name, P.l_name
FROM Flight F
JOIN Ticket T ON F.flight_id = T.flight_id
JOIN Passenger P ON T.passenger_id = P.passenger_id
JOIN Airline A ON F.airline_id = A.airline_id
WHERE P.passenger_id = input_passenger_id;
END;
```

- **Usage Page & Input Box:** procedure1.php via an input field named **passenger_id**

3.2 AssignCrewToFlight

- **Purpose:** Assigns a crew member to a flight with a role.
- **Parameters:** `input_flight_id INT`, `input_crew_id INT`, `input_role VARCHAR`
- **SQL Script:**

```
CREATE PROCEDURE AssignCrewToFlight(
    IN input_flight_id INT,
    IN input_crew_id INT,
    IN input_role VARCHAR(50)
)
BEGIN
    INSERT INTO assigned_to (flight_id, crew_id, role)
    VALUES (input_flight_id, input_crew_id, input_role);
END;
```

- **Usage Page & Input Boxes:** procedure2.php with input fields: **flight_id, crew_id, role**

3.3 GetFlightSummary

- **Purpose:** Shows detailed summary of a flight: tickets, passengers, baggage, crew, revenue.
- **Parameters:** input_flight_id INT
- **SQL Script:**

```
CREATE PROCEDURE GetFlightSummary(IN input_flight_id INT)
BEGIN
    SELECT F.flight_number, COUNT(DISTINCT T.ticket_id) AS total_tickets,
       COUNT(DISTINCT B.bag_number) AS total_baggage,
       COUNT(DISTINCT W.crew_id) AS crew_count,
       SUM(T.price) AS total_revenue
FROM Flight F
LEFT JOIN Ticket T ON F.flight_id = T.flight_id
LEFT JOIN Baggage B ON T.ticket_id = B.ticket_id
LEFT JOIN Works_in W ON F.flight_id = W.flight_id
WHERE F.flight_id = input_flight_id
GROUP BY F.flight_number;
END;
```

- **Usage Page & Input Box:** `procedure3.php` via an input field named **flight_id**

---

4. MongoDB Queries in Support System (Admin Panel)

Support Ticket Listing (Active & Resolved)

- **File:** `admin/index.php`
- **Query:** `find()` with `status` filter to list active or resolved tickets.

Ticket Details View

- **File:** `admin/detail.php`
- **Query:** `findOne()` by `_id` to fetch detailed ticket data.

Add Comment to Ticket

- **File:** `admin/add_comment.php`
- **Query:** `update()` using `$push` to append new comment.

Resolve a Ticket

- **File:** `admin/resolve_ticket.php`
- **Query:** `update()` using `$set` to mark ticket as resolved (`status = false`).

---

5. Conclusion

This project demonstrates a fully functional flight database system backed by relational and non-relational databases. Triggers and procedures ensure integrity and automation. The MongoDB-based support system provides real-time user interaction for ticket resolution, making the solution complete and ready for deployment in an institutional setting.