CS306 – Project Phase 2 Report

Triggers and Stored Procedures in Airplane DBMS

Group Members:

Yunus Emre Gök
Ufuk Çimen
Ahmet Çavuşoğlu

1. Introduction

This phase of the project focuses on implementing **triggers** and **stored procedures** using **MySQL** to improve the functionality and automation of the Airplane Database Management System. Triggers help automatically enforce rules and log changes, while stored procedures streamline complex operations and ensure consistency in execution.

2. Triggers

2.1.1 Trigger: CancelFlightIfNoTickets

Purpose: Cancels a flight automatically if all its tickets are deleted.
Before:1.1

flight_id	flight_status	ticket_count
1	On Time	1

After:1.2

flight_id	flight_status	ticket_count
1	Cancelled	0

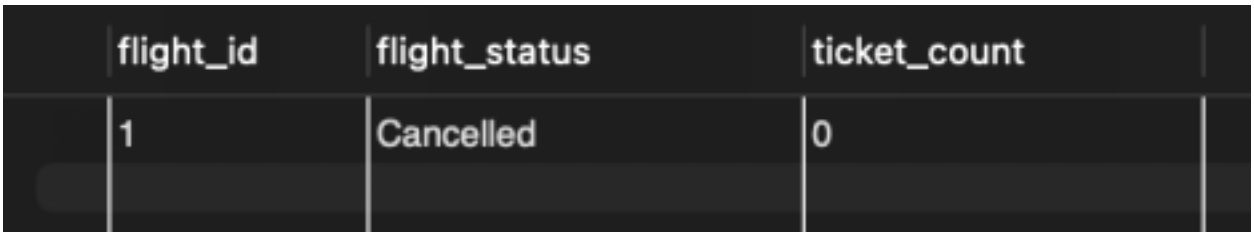
DELIMITER //

```
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//
DELIMITER ;
```

2.1.2 Trigger: ReactivateFlightIfTicketAdded

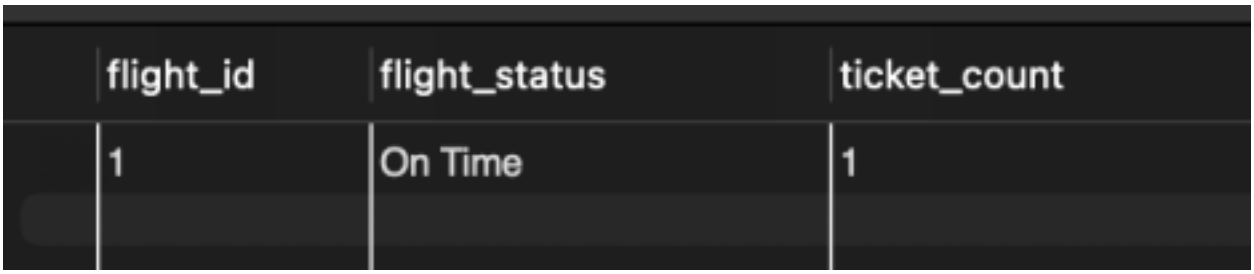
Purpose: Reactivates a cancelled flight if a new ticket is inserted.

Before:



flight_id	flight_status	ticket_count
1	Cancelled	0

After:



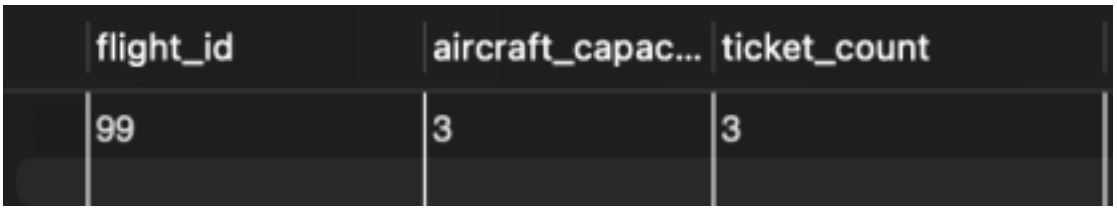
flight_id	flight_status	ticket_count
1	On Time	1

```
DELIMITER //
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//
DELIMITER ;
```

2.2 Trigger: PreventOverbooking

Purpose: Prevents ticket sales beyond the aircraft's capacity.

Before: 2.1



flight_id	aircraft_capac...	ticket_count
99	3	3

After: 2.2

```
DELIMITER //
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//
DELIMITER ;
```

2.1 Trigger: AutoCalculateExtraBaggageFee

Purpose: Calculates extra baggage fee automatically if weight exceeds 25kg.

Before: 3.1

ticket_id	passenger_id	bag_number	weight	calculated_f...
4001	1001	NULL	NULL	NULL

After: 3.2

ticket_id	passenger_id	bag_number	weight	calculated_f...
4001	1001	1	30.00	50.00
4001	1001	2	32.00	105.00

```
DELIMITER //
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//
DELIMITER ;
```

3. Stored Procedures

3.1 Procedure Name:GetPassengerFlights

```
 DELIMITER //
CREATE PROCEDURE GetPassengerFlights(IN input_passenger_id INT)
BEGIN
   SELECT
      T.ticket_id,
      F.flight_number,
      A.name AS airline,
      DA.name AS departure_airport,
      AA.name AS arrival_airport,
      F.departure_time,
      F.arrival_time,
      T.seat_number,
      T.class,
      T.price
   FROM Ticket T
   JOIN Flight F ON T.flight_id = F.flight_id
   JOIN Airline A ON F.airline_id = A.airline_id
   JOIN Airport DA ON F.departure_airport_id = DA.airport_id
   JOIN Airport AA ON F.arrival_airport_id = AA.airport_id
   WHERE T.passenger_id = input_passenger_id;
```

```
END//
DELIMITER ;
```

3.2 Procedure Name:AssignCrewToFlight

```
DELIMITER //
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//
DELIMITER ;
```

3.3 Procedure Name: GetFlightSummary

```
DELIMITER //
CREATE PROCEDURE GetFlightSummary(IN input_flight_id INT)
BEGIN
    SELECT
        F.flight_id,
        F.flight_number,
        F.status,
        A.model AS aircraft_model,
        A.capacity AS aircraft_capacity,
        -- Count of unique tickets
        (SELECT COUNT(*) FROM Ticket WHERE flight_id = input_flight_id) AS total_tickets_sold,
        -- Count of unique passengers
        (SELECT COUNT(DISTINCT passenger_id) FROM Ticket WHERE flight_id = input_flight_id) AS
total_passengers,
        -- Total revenue from tickets
        (SELECT IFNULL(SUM(price), 0) FROM Ticket WHERE flight_id = input_flight_id) AS total_revenue,
        -- Accurate count of baggage
        (SELECT COUNT(*)
         FROM Baggage B
         JOIN Ticket T ON B.ticket_id = T.ticket_id
         WHERE T.flight_id = input_flight_id) AS total_baggage_items,
```

```
        -- Accurate count of assigned crew
        (SELECT COUNT(*)
         FROM assigned_to
         WHERE flight_id = input_flight_id) AS total_crew_assigned
    FROM Flight F
    JOIN Aircraft A ON F.aircraft_id = A.aircraft_id
    WHERE F.flight_id = input_flight_id;
END//
DELIMITER ;
```