

Hıdır Can Çağlar 32174
Mehmet Berkay Çatak 32229
Ahmet Nusret Avcı 29289

Introduction

This phase focuses on implementing database triggers and stored procedures to enhance our database automation. The assignment involves developing multiple triggers to automate relevant database tasks and creating stored procedures for database operations.

Triggers

Room Vacancy Check:

The 'check_room_vacancy' trigger prevents double-booking by verifying room availability before allowing new reservations. It checks if the requested room is vacant and not already reserved with another active status. If the room is occupied or has an existing reservation, it blocks the new reservation with an error message. When a valid reservation is created, it automatically updates the room's vacancy status to FALSE.

```
DELIMITER //
```

```
CREATE TRIGGER check_room_vacancy  
BEFORE INSERT ON Reserve  
FOR EACH ROW  
BEGIN
```

```
    DECLARE room_is_vacant BOOLEAN;  
    DECLARE existing_reservation VARCHAR(40);
```

```
    -- Check if the room is vacant in the Rooms table
```

```
    SELECT Vacancy INTO room_is_vacant  
    FROM Rooms  
    WHERE RoomID = NEW.RoomID;
```

```
    -- Check if there's already an active reservation for this room
```

```
    SELECT ResStatus INTO existing_reservation  
    FROM Reserve
```

```

WHERE RoomID = NEW.RoomID
AND ResStatus IN ('Confirmed', 'Pending', 'Waiting List')
LIMIT 1;

-- If room is not vacant raise the 45000
IF room_is_vacant = FALSE OR existing_reservation IS NOT NULL THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Cannot reserve room: Room is already occupied or
reserved';
END IF;

-- If new reservation is not set to 'Cancelled', update room vacancy to false
IF NEW.ResStatus IN ('Confirmed', 'Pending', 'Waiting List') THEN
    UPDATE Rooms SET Vacancy = FALSE WHERE RoomID = NEW.RoomID;
END IF;
END //

DELIMITER ;

```

Here is the status of our database after the execution of this trigger:

The screenshot shows the MySQL Workbench interface. The 'Query' window displays a SQL query that selects room details and reservation status for RoomID 101. The 'Result Grid' shows the query results, indicating that RoomID 101 is currently occupied (Vacancy = 0) and has a confirmed reservation (GuestID = 3, ResStatus = Confirmed).

The 'Output' window shows the execution log, including the following messages:

#	Time	Action	Message	Duration / Fetch
338	22:02:50	UPDATE Rooms SET Vacancy = TRUE WHERE RoomID = 101	Error Code: 1544. Parking payment hasn't been completed. checkout request denied.	0.000 sec
339	22:03:55	SELECT r.RoomID, r.Vacancy, g.GuestID, g.Name, p.PSpotNum, p.PayStatus FROM Rooms r JOIN Reserve res ON r.RoomID = ...	1 row(s) returned	0.000 sec / 0.000 sec
340	22:06:33	CREATE TRIGGER room_vacancy_check BEFORE INSERT ON Reserve FOR EACH ROW BEGIN DECLARE room_is_v...	Error Code: 1359. Trigger already exists	0.078 sec
341	22:08:01	SELECT r.RoomID, r.Vacancy, res.GuestID, res.ResStatus FROM Rooms r LEFT JOIN Reserve res ON r.RoomID = res.RoomID ...	1 row(s) returned	0.046 sec / 0.000 sec
342	22:08:52	INSERT INTO Reserve (GuestID, RoomID, BookingDate, Duration, ResStatus) VALUES (5, 101, '2025-11-01', 3, 'Confirmed')	Error Code: 1544. Cannot reserve room: Room is already occupied or reserved	0.015 sec
343	22:08:59	SELECT r.RoomID, r.Vacancy, res.GuestID, res.ResStatus FROM Rooms r LEFT JOIN Reserve res ON r.RoomID = res.RoomID ...	1 row(s) returned	0.000 sec / 0.000 sec

Parking Payment:

The 'check_parking_payment' trigger prevents room checkout (vacancy update to TRUE) if a guest has unpaid parking fees. It verifies the parking payment status when a room's vacancy changes and blocks the checkout process with an error message if the parking payment status is incomplete (if PayStatus is FALSE).

Guest Age Check:

The 'check_guest_age' database trigger validates that all guests making reservations are at least 18 years old. It executes before any insertion into the Reserve table, calculates the reserver's age based on their date of birth, and prevents underage reservations by returning an error message if the age requirement isn't met.

Stored Procedures

Applying Loyalty Discount:

This Stored Procedure named calculate_guest_bill calculates the final bill for the guest that has reserved this room. If the procedure is called upon a guest who is the reservation owner, if their loyalty is "Applied", their final bill will get a 5% discount. To do this, first we check if their ResOwner status is true, if not we throw an error message stating that they cannot see the bill and it won't be calculated. Else, to calculate first we get their loyalty status, then the Room ID they reserved, how long they have stayed in the room, how much the room costs for a day, calculate how much the room costs normally. Then we look at if they have a car parked in. If they did, we take the days this car has been parked and charge 20\$ for parking the car for each day. With this we have calculated the total bill before loyalty. If this reservation owner isn't in loyalty program they will pay this price, else they will pay with 5% discount. Then we print the table to see the details for this guest.

DELIMITER //

```
CREATE PROCEDURE calculate_guest_bill(IN guest_id_param INT)
BEGIN
    DECLARE guest_loyalty VARCHAR(20);
    DECLARE room_id_val INT;
    DECLARE duration_val INT;
    DECLARE room_price DECIMAL(10,2);
    DECLARE total_room_price DECIMAL(10,2);
    DECLARE parking_spot INT;
    DECLARE days_parked INT;
    DECLARE parking_price DECIMAL(10,2);
    DECLARE total_bill DECIMAL(10,2);
    DECLARE final_bill DECIMAL(10,2);
    DECLARE discount_rate DECIMAL(5,2);
    DECLARE is_res_owner BOOLEAN;
    DECLARE has_parking BOOLEAN DEFAULT FALSE;

    -- Check if the guest is the ResOwner
    SELECT ResOwner INTO is_res_owner
    FROM Guests
    WHERE GuestID = guest_id_param;

    -- error if not ResOwner
    IF is_res_owner = FALSE THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Only reservation owners can view bills';
        -- Exit the procedure
    END IF;

    -- Get the loyalty program
    SELECT LoyaltyProg INTO guest_loyalty
    FROM Guests
    WHERE GuestID = guest_id_param;

    -- Get the room details and duration
    SELECT r.RoomID, r.Duration INTO room_id_val, duration_val
    FROM Reserve r
    WHERE r.GuestID = guest_id_param;

    -- Get room price
    SELECT Pricing INTO room_price
    FROM Rooms
```

```

WHERE RoomID = room_id_val;

-- total room price
SET total_room_price = room_price * duration_val;

-- Check if guest has a parking spot
SELECT COUNT(*) > 0 INTO has_parking
FROM P_Use
WHERE GuestID = guest_id_param;

-- Initialize parking variables
SET parking_spot = NULL;
SET days_parked = 0;
SET parking_price = 0;

-- Calculate parking if the guest has a parking spot
IF has_parking THEN
    SELECT PSpotNum INTO parking_spot
    FROM P_Use
    WHERE GuestID = guest_id_param;

    -- total days parked
    SELECT DaysParked INTO days_parked
    FROM Parking
    WHERE PSpotNum = parking_spot;

    -- total parking price (20$ per day)
    SET parking_price = days_parked * 20;
END IF;

-- total bill before loyalty
SET total_bill = total_room_price + parking_price;

-- Apply loyalty discount if "Applied"
IF guest_loyalty = 'Applied' THEN
    SET discount_rate = 0.05; -- 5% discount
    SET final_bill = total_bill * (1 - discount_rate);
ELSE
    SET final_bill = total_bill;
END IF;

-- Result table
SELECT

```

```

    guest_id_param AS GuestID,
    guest_loyalty AS LoyaltyStatus,
    room_id_val AS RoomID,
    room_price AS RoomPricePerDay,
    duration_val AS DaysStayed,
    total_room_price AS TotalRoomPrice,
    parking_spot AS ParkingSpotNumber,
    days_parked AS DaysParked,
    parking_price AS TotalParkingPrice,
    total_bill AS TotalBillBeforeDiscount,
    CASE
        WHEN guest_loyalty = 'Applied' THEN 'Yes (5%)'
        ELSE 'No'
    END AS DiscountApplied,
    final_bill AS FinalBill;
END //

DELIMITER ;

```

Updating Loyalty Status:

The “LoyaltyGiver” stored procedure takes GuestID as a parameter and checks if the given GuestID exists within the database. If it exists, the procedure checks if that guest booked a reservation from the hotel 5 times or more, updates the guest’s LoyaltyProg status from “Not Applied” to “Applied” and outputs a message indicating that the guest has the Loyalty Status. If the guest reserved a room from the hotel less than 5 times, the procedure outputs a message stating that the guest doesn’t have the Loyalty Status.

Listing Tours Guests Attended:

The "GetGuestTours" stored procedure takes a guest's Name and Surname as parameters and checks if the given guest exists within the database. If found, the procedure joins the Guests, Attend, and Tours tables to retrieve all tours that guests have attended. The procedure returns the tour name, tour type, and tour company for each matching record. If no tours are found for the specified guest, the procedure will return an empty result set.