

Kazakh-British Technical University
DATABASES
Laboratory work №10

Alexandr Kislitsin
23B031861

2. Create a stored procedure to update the status of a flight.

The screenshot shows a database IDE with the following components:

- Database Explorer:** Shows a database named 'airport_db' with tables including 'Airline', 'Airline_Info', 'Airport', 'Baggage', 'Baggage_check', 'Boarding_pass', 'Booking', 'Booking_flight', 'Flights', 'Passengers', 'Security_check', and 'routines'.
- Code Editor:** Contains the SQL code for creating and calling a stored procedure:

```
CREATE PROCEDURE update_flight_status(  
    IN p_flight_id INT,  
    IN p_new_status VARCHAR(50)  
)  
BEGIN  
    DECLARE v_rows_affected INT;  
  
    UPDATE Flights  
    SET status = p_new_status,  
        update_at = NOW()  
    WHERE flight_id = p_flight_id;  
  
    SET v_rows_affected = ROW_COUNT();  
  
    IF v_rows_affected > 0 THEN  
        SELECT CONCAT('Статус рейса ', p_flight_id, ' обновлен на ', p_new_status, '') AS result;  
        SELECT * FROM Flights WHERE flight_id = p_flight_id;  
    ELSE  
        SELECT CONCAT('Рейс с ID ', p_flight_id, ' не найден') AS result;  
    END IF;  
END;  
  
CALL update_flight_status( p_flight_id 1, p_new_status 'boarding');
```
- Output/Result:** Shows the result of the procedure call, displaying flight details for flight_id 1:

flight_id	flight_no	scheduled_departure	scheduled_arrival	departure_airport_id
1	FL0001	2024-01-22	2023-09-08	
- Services:** Shows the execution time for the 'last_labs.sql' file as 787 ms.

3. Create a stored procedure that returns a list of flights departing from a specific airport.

The screenshot shows a database IDE with the following components:

- Database Explorer:** Shows the same database structure as the previous screenshot.
- Code Editor:** Contains the SQL code for creating and calling a stored procedure:

```
CREATE PROCEDURE get_flights_from_airport(  
    f.status,  
    dep.airport_name AS departure_airport,  
    arr.airport_name AS arrival_airport,  
    a.airline_name  
FROM Flights f  
    JOIN Airport dep ON f.departure_airport_id = dep.airport_id  
    JOIN Airport arr ON f.arrival_airport_id = arr.airport_id  
    JOIN Airline a ON f.airline_id = a.airline_id  
WHERE f.departure_airport_id = p_airport_id  
ORDER BY f.scheduled_departure;  
  
SELECT COUNT(*) AS total_flights  
FROM Flights  
WHERE departure_airport_id = p_airport_id;  
END;  
  
CALL get_flights_from_airport( p_airport_id 1);
```
- Output/Result:** Shows the result of the procedure call, displaying the total number of flights departing from airport_id 1:

total_flights
49
- Services:** Shows the execution time for the 'last_labs.sql' file as 743 ms.

4. Create a function to calculate the average delay time of flights arriving at a specific airport.

The screenshot shows a database IDE with the following components:

- Database Explorer:** Shows the database structure with tables like Airline, Airline_Info, Airport, Baggage, Baggage_check, Boarding_pass, Booking, Booking_flight, Flights, and columns like flight_no, flight_no_2.
- SQL Editor:** Contains the following SQL code:

```
CREATE FUNCTION calculate_avg_delay(  
    p_airport_id INT  
)  
RETURNS DECIMAL(10,2)  
BEGIN  
    RETURN IFNULL(avg_delay, 0);  
END;  
  
SELECT calculate_avg_delay(p_airport_id 1) AS avg_delay_hours;  
  
SELECT  
    airport_id,  
    airport_name,  
    calculate_avg_delay(p_airport_id airport_id) AS avg_delay_hours  
FROM Airport  
LIMIT 5;
```
- Output:** Displays the results of the function call for airport_id 1:

airport_id	airport_name	avg_delay_hours
1	Akunag Heliport	3087.72
2	Alert Bay Airport	3689.00
3	Armidale Airport	3162.46
4	Bermuda Dunes Airport	3161.00
5	Darchula Airport	2845.85
- Services:** Shows the execution time for the last query as 403 ms.

5. Create a stored procedure that lists all passengers for a given flight number.

The screenshot shows a database IDE with the following components:

- Database Explorer:** Shows the database structure with tables like Boarding_pass, Booking, Booking_flight, Flights, Passengers, Security_check, and routines.
- SQL Editor:** Contains the following SQL code:

```
CREATE PROCEDURE get_passengers_for_flight(  
    p_flight_id INT  
)  
BEGIN  
    SELECT  
        bp.seat,  
        bp.boarding_time  
    FROM Passengers p  
    JOIN Booking b ON p.passenger_id = b.passenger_id  
    LEFT JOIN Boarding_pass bp ON b.booking_id = bp.booking_id  
    WHERE b.flight_id = p_flight_id  
    ORDER BY p.last_name, p.first_name;  
  
    SELECT COUNT(DISTINCT p.passenger_id) AS total_passengers  
    FROM Passengers p  
    JOIN Booking b ON p.passenger_id = b.passenger_id  
    WHERE b.flight_id = p_flight_id;  
END;  
  
CALL get_passengers_for_flight(p_flight_id 1);
```
- Output:** Displays the results of the procedure call for flight_id 1:

total_passengers
0
- Services:** Shows the execution time for the last query as 436 ms.

6. Create a stored procedure to find the passenger who has taken the greatest number of flights.

The screenshot shows a database IDE with the following components:

- Database Explorer:** Shows a database structure with tables like Boarding_pass, Booking, Booking_flight, Flights, Passengers, and Security_check. The 'Flights' table is selected.
- Code Editor:** Contains the following SQL code:

```
CREATE PROCEDURE find_most_frequent_passenger()
BEGIN
    SELECT
        p.passenger_id,
        CONCAT(p.first_name, ' ', p.last_name) AS full_name,
        p.passport_number,
        p.country_of_citizenship,
        COUNT(b.booking_id) AS total_flights
    FROM Passengers p
    JOIN Booking b ON p.passenger_id = b.passenger_id
    GROUP BY p.passenger_id, p.first_name, p.last_name, p.passport_number, p.country_of_citizenship
    ORDER BY total_flights DESC
    LIMIT 1;
END;

CALL find_most_frequent_passenger();
```
- Output:** Shows the result of the procedure call:

passenger_id	full_name	passport_number	country_of_citizenship	total_flights
68	Sheela Roux	014456243-X	Thailand	
- Services:** Shows the execution time for the 'last_Labs.sql' file as 396 ms.

7. Create a stored procedure to find all flights that are delayed by more than 24 hours.

The screenshot shows a database IDE with the following components:

- Database Explorer:** Similar to the previous screenshot, with the 'Flights' table selected.
- Code Editor:** Contains the following SQL code:

```
CREATE PROCEDURE find_delayed_flights_24h()
-- Дополнительно выводим статистику
SELECT
    COUNT(*) AS total_delayed_flights,
    AVG(TIMESTAMPDIFF(HOUR, scheduled_arrival, actual_arrival)) AS avg_delay_hours,
    MAX(TIMESTAMPDIFF(HOUR, scheduled_arrival, actual_arrival)) AS max_delay_hours
FROM Flights
WHERE TIMESTAMPDIFF(HOUR, scheduled_arrival, actual_arrival) > 24;
END;

CALL find_delayed_flights_24h();
```
- Output:** Shows the result of the procedure call:

total_delayed_flights	avg_delay_hours	max_delay_hours
507	2890.9349	8280
- Services:** Shows the execution time for the 'last_Labs.sql' file as 764 ms.

8. Create a function that counts the number of flights for each airline.

The screenshot shows the SQL Studio interface with the following components:

- Database Explorer:** Displays the database structure for 'last_labs.sql'. The 'Flights' table is highlighted, showing columns 'flightNo' and 'flightNo_2'.
- SQL Editor:** Contains the following SQL code:

```
SELECT
    a.airline_id,
    a.airline_name,
    a.airline_country,
    count_flights_by_airline(p.airline_id) AS total_flights
FROM Airline a
ORDER BY total_flights DESC;

SELECT count_flights_by_airline(p.airline_id) AS flights_count;
```
- Output:** Shows the result of the second query, displaying a single row with the value '32' for 'flights_count'.
- Services:** Lists the execution times for the queries: 'Passengers' (436 ms), 'last_labs.sql' (363 ms), and 'Flights' (637 ms).
- Notifications:** Shows a timeline of events, including 'Connection established' and 'Query executed'.

9. Create a stored procedure to calculate the average ticket price for a specific flight.

The screenshot shows the SQL Studio interface with the following components:

- Database Explorer:** Displays the database structure for 'last_labs.sql'. The 'Flights' table is highlighted, showing columns 'flightNo' and 'flightNo_2'.
- SQL Editor:** Contains the following SQL code:

```
CREATE PROCEDURE calculate_avg_ticket_price(
    v_min_price AS min_price,
    v_max_price AS max_price,
    v_booking_count AS total_bookings;
ELSE
    SELECT
        p_flight_id AS flight_id,
        'Нет бронирований для этого рейса' AS message;
END IF;
END;

CALL calculate_avg_ticket_price(p_flight_id 1);
```
- Output:** Shows the result of the stored procedure call, displaying a single row with the message 'Нет бронирований для этого рейса' (No bookings for this flight).
- Services:** Lists the execution times for the queries: 'Passengers' (436 ms), 'last_labs.sql' (359 ms), and 'Flights' (637 ms).
- Notifications:** Shows a timeline of events, including 'Connection established' and 'Query executed'.

10. Create a stored procedure to find the flight with the highest ticket price. The procedure should return the flight number, the departure and arrival airports, and the ticket price for the most expensive flight.

The screenshot shows a database IDE interface with the following components:

- Database Explorer:** Displays a tree view of the database structure. The 'Flights' table is selected, showing its columns and keys.
- Code Editor:** Contains the SQL code for creating and calling the stored procedure:

```
CREATE PROCEDURE find_most_expensive_flight()
FROM Booking b
JOIN Flights f ON b.flight_id = f.flight_id
JOIN Airport dep ON f.departure_airport_id = dep.airport_id
JOIN Airport arr ON f.arrival_airport_id = arr.airport_id
JOIN Airline a ON f.airline_id = a.airline_id
ORDER BY b.ticket_price DESC
LIMIT 5;
END;

CALL find_most_expensive_flight();
```
- Output Panel:** Shows the results of the stored procedure execution. The output is a table with the following columns: flight_id, departure_airport, arrival_airport, airline_name, ticket_price, and scheduled_time. The results are displayed in a grid format.
- Services Panel:** Shows the execution time for the stored procedure: 637 ms.