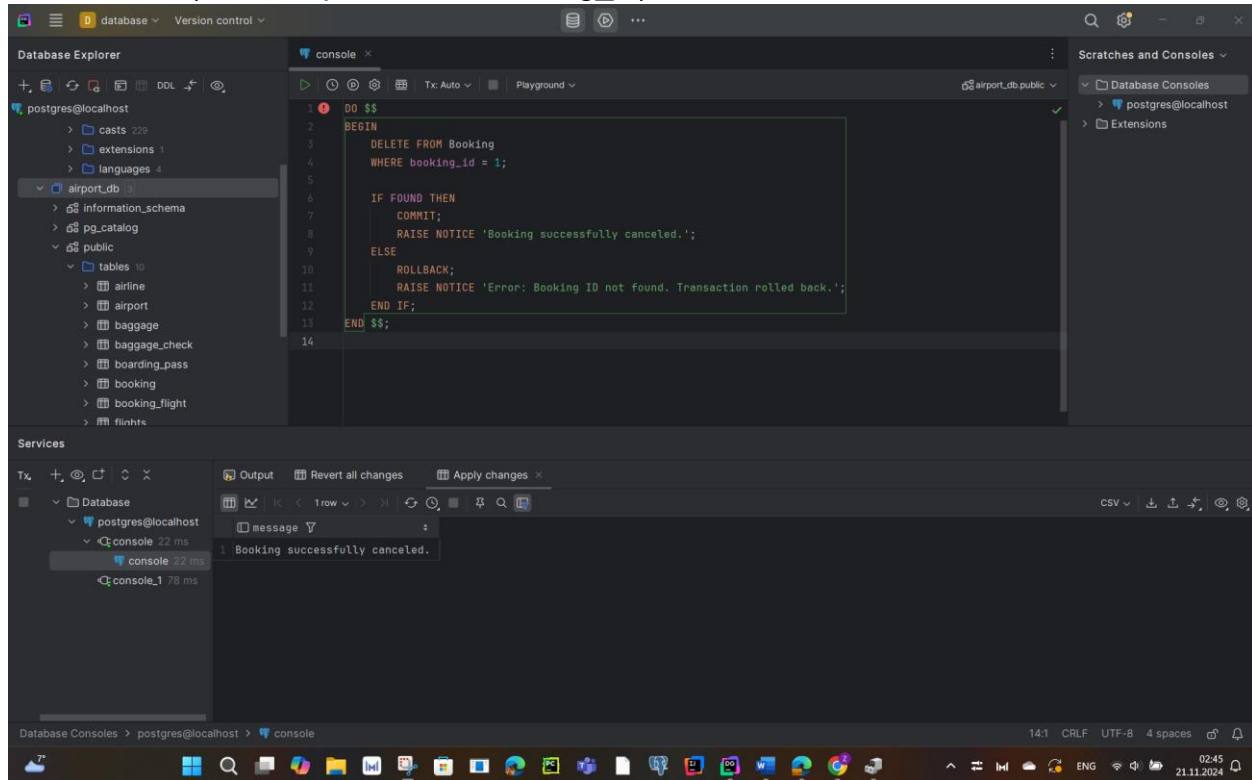


## Laboratory work 9

### TRANSACTION.

1. A passenger cancels their booking. You need to remove the booking for the flight. Ensure the 'booking' table no longer contains the booking. Simulate an error to test rollback (for example, invalid booking\_id).



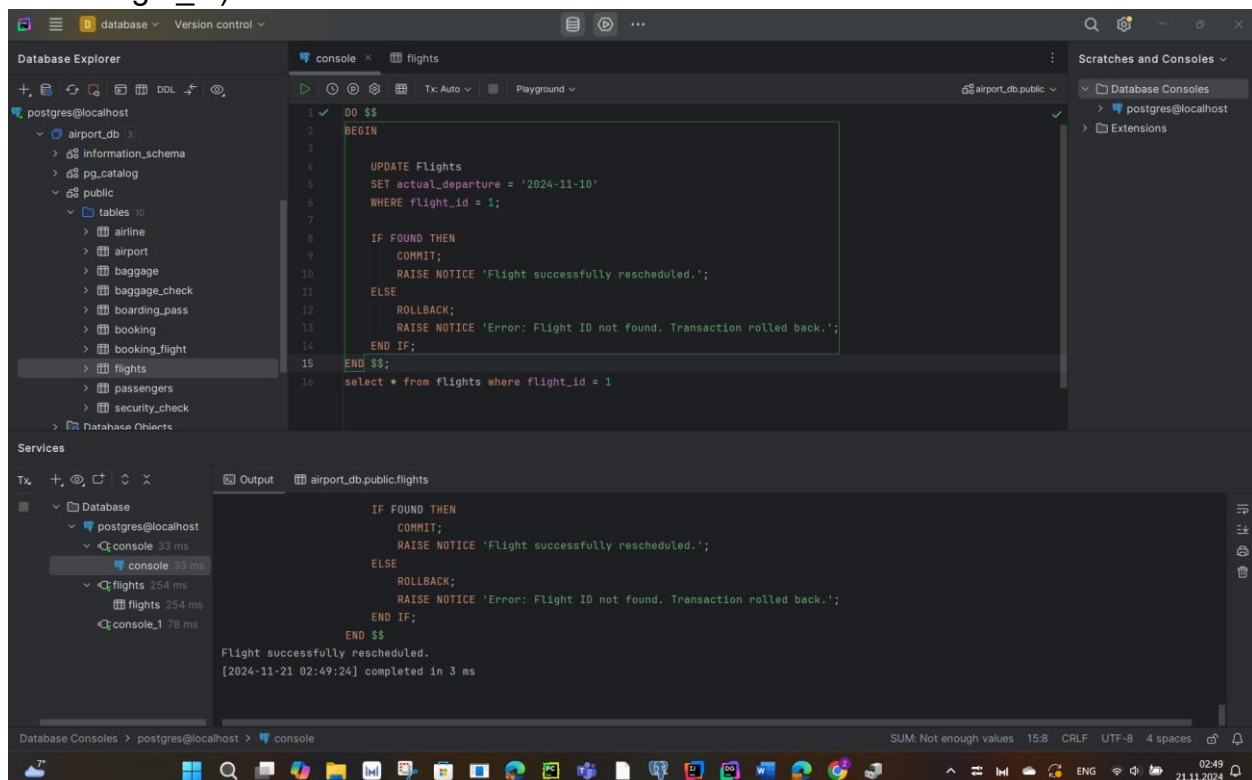
The screenshot shows a database management tool interface. On the left, the 'Database Explorer' pane shows a tree structure with 'airport\_db' selected, containing tables like 'airline', 'airport', 'baggage', 'baggage\_check', 'boarding\_pass', 'booking', 'booking\_flight', and 'flights'. The main editor displays a SQL transaction script:

```
1 DO $$
2 BEGIN
3   DELETE FROM Booking
4   WHERE booking_id = 1;
5
6   IF FOUND THEN
7     COMMIT;
8     RAISE NOTICE 'Booking successfully canceled.';
9   ELSE
10    ROLLBACK;
11    RAISE NOTICE 'Error: Booking ID not found. Transaction rolled back.';
12  END IF;
13 END $$;
```

The 'console' pane at the bottom shows the execution results:

```
1 Booking successfully canceled.
```

2. Rescheduling a flight. You need to reschedule a flight. Verify the 'flights' table reflects the new departure time. Simulate an error to test rollback (for example, invalid flight\_id).



The screenshot shows the same database management tool interface. The 'Database Explorer' pane shows the 'flights' table selected. The main editor displays a SQL transaction script:

```
1 DO $$
2 BEGIN
3   UPDATE Flights
4   SET actual_departure = '2024-11-10'
5   WHERE flight_id = 1;
6
7   IF FOUND THEN
8     COMMIT;
9     RAISE NOTICE 'Flight successfully rescheduled.';
10  ELSE
11    ROLLBACK;
12    RAISE NOTICE 'Error: Flight ID not found. Transaction rolled back.';
13  END IF;
14 END $$;
```

The 'console' pane at the bottom shows the execution results:

```
Flight successfully rescheduled.
[2024-11-21 02:49:24] completed in 3 ms
```

3. Updating ticket prices. You need to decrease the ticket price for a specific flight for all existing bookings. If an error occurs, no changes should be applied.

The screenshot shows a database console interface with a dark theme. On the left, the 'Database Explorer' pane shows a tree view of the database structure, including schemas like 'information\_schema', 'pg\_catalog', and 'public', and tables like 'airline', 'airport', 'baggage', 'baggage\_check', 'boarding\_pass', 'booking', 'booking\_flight', 'flights', 'passengers', and 'security\_check'. The 'booking' table is selected. The main console area displays a SQL transaction being executed. The transaction starts with 'DO \$\$', followed by a 'BEGIN' block. Inside the block, an 'UPDATE booking' statement is used to decrease the price by 50 for a specific booking\_id (50). This is followed by an 'IF FOUND THEN' block containing 'COMMIT;' and a 'RAISE NOTICE' statement. An 'ELSE' block contains 'ROLLBACK;' and another 'RAISE NOTICE' statement. The transaction ends with 'END IF;' and 'END \$\$'. The console output shows the transaction completed successfully, with a notice: 'Ticket prices successfully updated.' and a completion time of 7 ms. The bottom status bar indicates the connection is to 'postgres@localhost' and the console is in 'CRLF' mode with 'UTF-8' encoding and '4 spaces' indentation.

```
1 DO $$
2 BEGIN
3     UPDATE booking
4     SET price = price - 50
5     WHERE booking_id = 50;
6
7
8     IF FOUND THEN
9         COMMIT;
10        RAISE NOTICE 'Ticket prices successfully updated.';
11    ELSE
12        ROLLBACK;
13        RAISE NOTICE 'Error: No tickets found for the flight. Transaction rolled back.';
14    END IF;
15 END $$;
```

Ticket prices successfully updated.  
[2024-11-21 02:52:10] completed in 7 ms

4. A passenger updates their details. Ensure the update is reflected across all associated records, including bookings.

The screenshot shows the same database console interface. The 'Database Explorer' pane is still visible on the left. The main console area displays a new SQL transaction. This transaction starts with 'DO \$\$', followed by a 'BEGIN' block. Inside the block, an 'UPDATE Passengers' statement is used to update the first\_name, last\_name, and country\_of\_residence for a specific passenger\_id (123). This is followed by an 'IF NOT FOUND THEN' block containing 'ROLLBACK;' and a 'RAISE NOTICE' statement. A 'RETURN;' statement follows. Then, an 'UPDATE Booking' statement is used to update the booking\_platform to 'Frami Inc' for the same passenger\_id. This is followed by another 'IF NOT FOUND THEN' block containing 'ROLLBACK;' and a 'RAISE NOTICE' statement. The transaction ends with 'END IF;' and 'END \$\$'. The console output shows the transaction completed successfully, with a notice: 'Passenger details successfully updated.' and a completion time of 3 ms. The bottom status bar indicates the connection is to 'postgres@localhost' and the console is in 'CRLF' mode with 'UTF-8' encoding and '4 spaces' indentation.

```
1 DO $$
2 BEGIN
3
4     UPDATE Passengers
5     SET first_name = 'John', last_name = 'Doe', country_of_residence = 'USA'
6     WHERE passenger_id = 123;
7
8     IF NOT FOUND THEN
9         ROLLBACK;
10        RAISE NOTICE 'Error: Passenger ID not found. Transaction rolled back.';
11        RETURN;
12    END IF;
13
14    UPDATE Booking
15    SET booking_platform = 'Frami Inc'
16    WHERE passenger_id = 123;
17
18    IF NOT FOUND THEN
19        ROLLBACK;
20        RAISE NOTICE 'Error: Unable to update associated bookings. Transaction rolled back.';
21    END IF;
22 END $$;
```

Passenger details successfully updated.  
[2024-11-21 03:07:04] completed in 3 ms

5. A new passenger is registered, and a booking is created. Ensure the new passenger is added and the booking succeeds.

```
1 DO $$
2 DECLARE
3     new_passenger_id INT;
4 BEGIN
5
6     INSERT INTO Passengers (passenger_id, first_name, last_name, date_of_birth, gender, country_of_citizenship)
7     VALUES (passenger_id 1000, first_name 'Jane', last_name 'Smith', date_of_birth '1982-06-07', gender 'Male', country
8     RETURNING passenger_id INTO new_passenger_id;
9
10    IF NOT FOUND THEN
11        ROLLBACK;
12        RAISE NOTICE 'Error: Failed to insert passenger. Transaction rolled back.';
13        RETURN;
14    END IF;
15
16    INSERT INTO Booking (passenger_id, booking_id, booking_platform)
17    VALUES (passenger_id 500, booking_id 1000, booking_platform 'ifnise');
18
19    IF NOT FOUND THEN
20        ROLLBACK;
21        RAISE NOTICE 'Error: Failed to create booking. Transaction rolled back.';
22    END IF;
23
24    COMMIT;
25    RAISE NOTICE 'Passenger and booking successfully added.';
26
27 END $$
```

[2024-11-21 03:15:28] [23503] ОШИБКА: INSERT или UPDATE в таблице "booking" нарушает ограничение внешнего ключа "booking\_passenger\_id\_fkey"

6. Increase the ticket price for all bookings on a specific flight by a fixed amount.

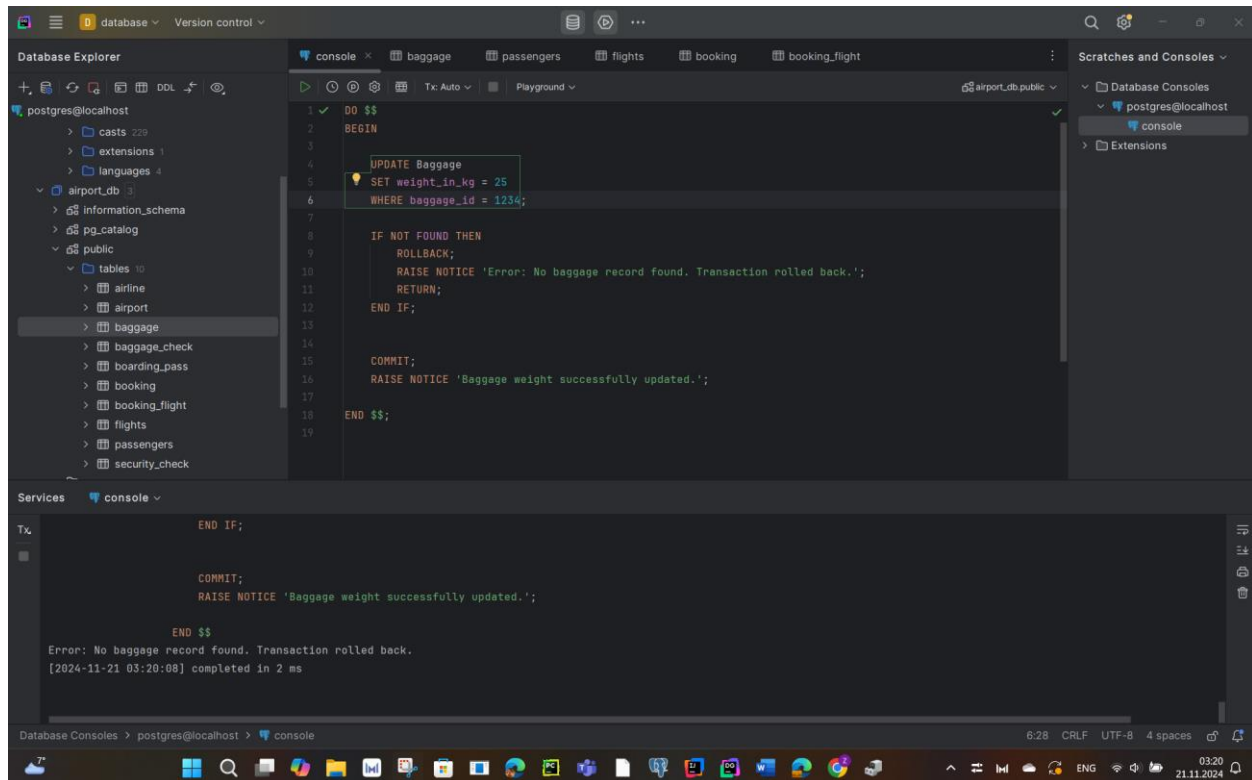
```
1 DO $$
2 BEGIN
3
4     UPDATE booking
5     SET price = price + 100
6     WHERE passenger_id = 499;
7
8     IF NOT FOUND THEN
9        ROLLBACK;
10        RAISE NOTICE 'Error: No tickets found for the flight. Transaction rolled back.';
11        RETURN;
12    END IF;
13
14    COMMIT;
15    RAISE NOTICE 'Ticket prices successfully increased.';
16
17 END $$;
```

Services All Services

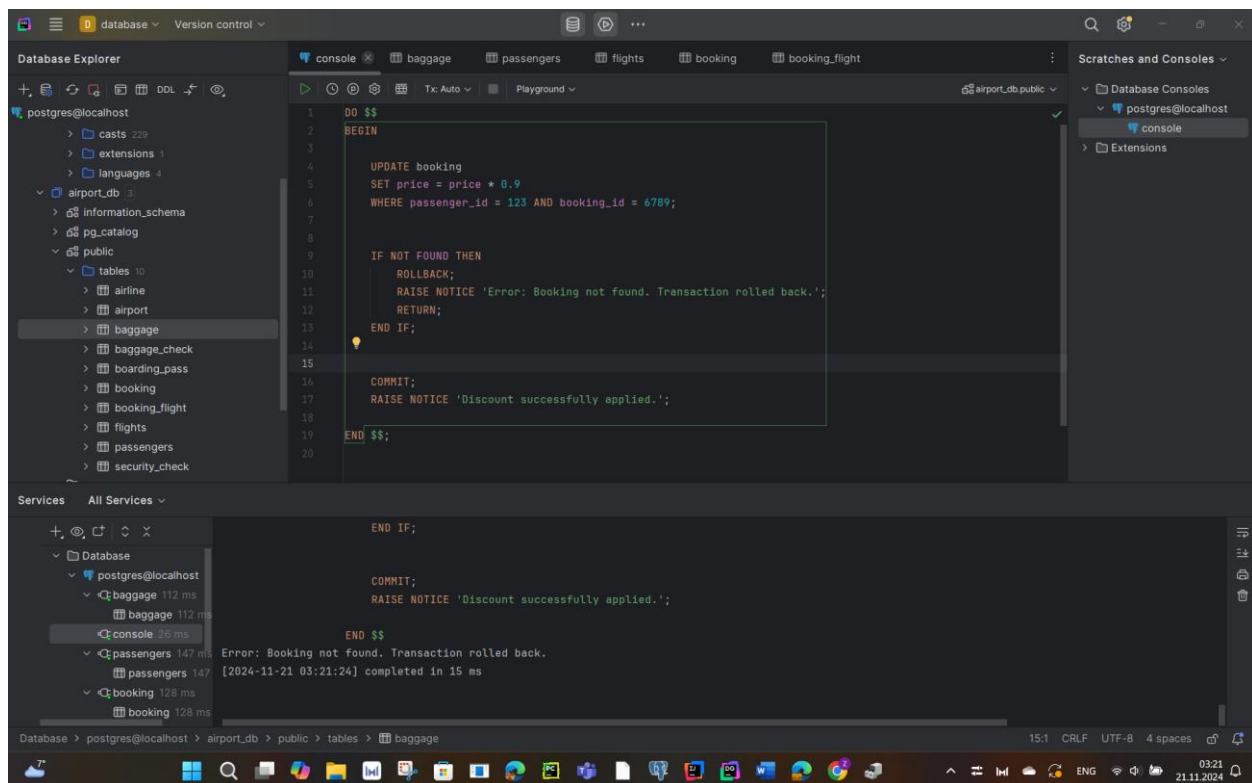
- postgres@localhost
- console 16 ms
- passengers 147 ms
- booking 128 ms
- flights 109 ms

Error: No tickets found for the flight. Transaction rolled back.  
[2024-11-21 03:18:13] completed in 3 ms

7. Update a baggage weight. A passenger updates the declared weight of their baggage. Ensure that the change is correctly reflected in the database.



8. Apply a discount to a booking for a specific passenger. If any error occurs, roll back.



9. Reschedule all bookings for a flight to a new flight.

database Version control

Database Explorer

- postgres@localhost
  - casts 229
  - extensions 1
  - languages 4
  - airport\_db 3
    - information\_schema
    - pg\_catalog
    - public
      - tables 10
        - airline
        - airport
        - baggage
        - baggage\_check
        - boarding\_pass
        - booking
        - booking\_flight
        - flights
        - passengers
        - security\_check

console

```
1 DO $$
2 BEGIN
3
4     UPDATE booking_flight
5     SET flight_id = 9
6     WHERE flight_id = 5;
7
8     IF NOT FOUND THEN
9         ROLLBACK;
10        RAISE NOTICE 'Error: No bookings found for the flight. Transaction rolled back.';
11        RETURN;
12    END IF;
13
14    COMMIT;
15    RAISE NOTICE 'All bookings successfully rescheduled.';
16
17 END $$;
```

Scratches and Consoles

- Database Consoles
  - postgres@localhost
    - console
    - Extensions

Services All Services

- Database
  - postgres@localhost
    - baggage 112 ms
    - baggage 112 ms
    - console 15 ms
    - passengers 147 ms All bookings successfully rescheduled.
    - passengers 147 [2024-11-21 03:22:56] completed in 6 ms
    - booking 128 ms
    - booking 128 ms

Database Consoles > postgres@localhost > console

5:22 CRLF UTF-8 4 spaces

03:23 21.11.2024