

**ФГАОУ ВО «МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ»**

Лабораторная работа №8

Расширения реляционной модели

По дисциплине:

Базы данных

Выполнил студент 1-го курса группы 243-323

Онищенко А. А.

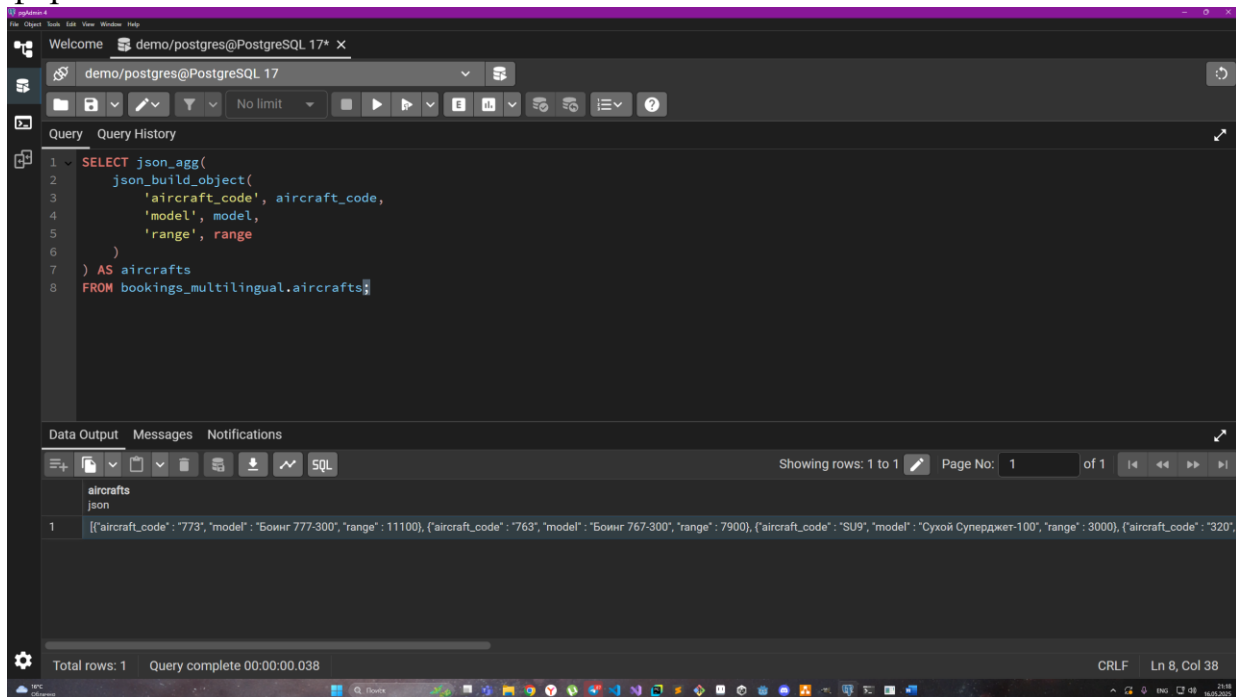
Проверил

_____ Красникова И.Н.

Москва, 2024

• Упражнение 8.1

Напишите запрос, выдающий список самолетов из демонстрационной базы в формате JSON.



The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL query:

```
1 SELECT json_agg(  
2     json_build_object(  
3         'aircraft_code', aircraft_code,  
4         'model', model,  
5         'range', range  
6     )  
7 ) AS aircrafts  
8 FROM bookings_multilingual.aircrafts;
```

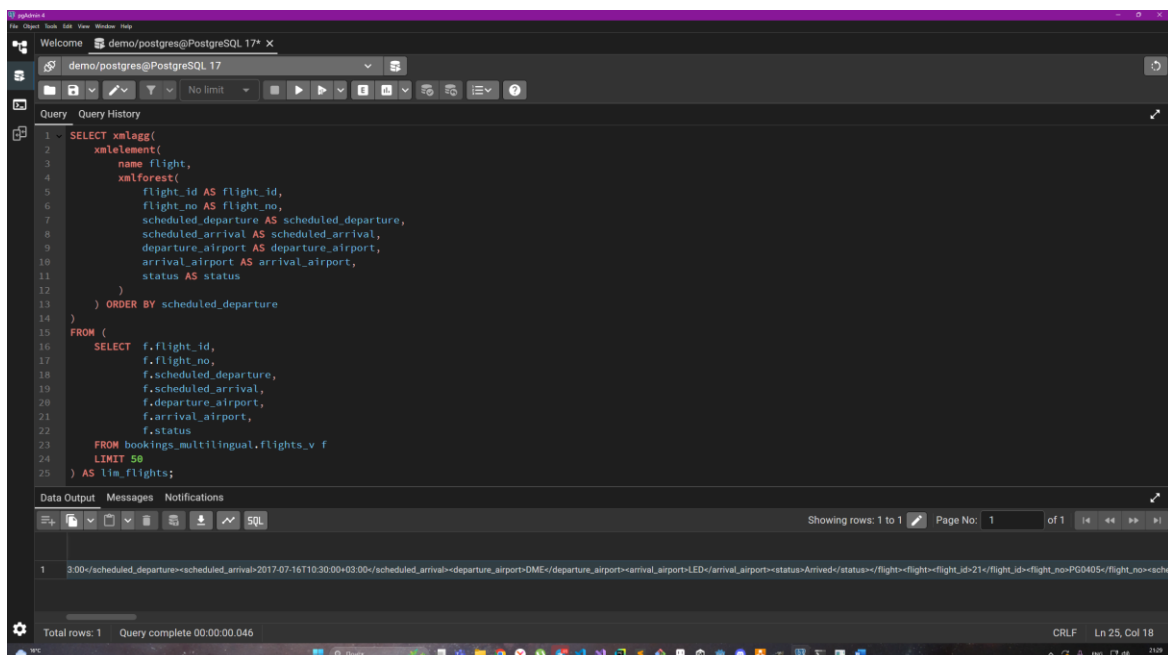
The Data Output pane shows the result in JSON format:

```
1 [{"aircraft_code": "773", "model": "Боинг 777-300", "range": 11100}, {"aircraft_code": "763", "model": "Боинг 767-300", "range": 7900}, {"aircraft_code": "SU9", "model": "Сухой Суперджет-100", "range": 3000}, {"aircraft_code": "320", "model": "Аэробус A320-200", "range": 5700}, {"aircraft_code": "321", "model": "Аэробус A321-200", "range": 5600}, {"aircraft_code": "319", "model": "Аэробус A319-100", "range": 6700}, {"aircraft_code": "733", "model": "Боинг 737-300", "range": 4200}, {"aircraft_code": "CN1", "model": "Сессна 208 Караван", "range": 1200}, {"aircraft_code": "CR2", "model": "Бомбардье CRJ-200", "range": 2700}]
```

```
[{"aircraft_code": "773", "model": "Боинг 777-300", "range": 11100}, {"aircraft_code": "763", "model": "Боинг 767-300", "range": 7900}, {"aircraft_code": "SU9", "model": "Сухой Суперджет-100", "range": 3000}, {"aircraft_code": "320", "model": "Аэробус A320-200", "range": 5700}, {"aircraft_code": "321", "model": "Аэробус A321-200", "range": 5600}, {"aircraft_code": "319", "model": "Аэробус A319-100", "range": 6700}, {"aircraft_code": "733", "model": "Боинг 737-300", "range": 4200}, {"aircraft_code": "CN1", "model": "Сессна 208 Караван", "range": 1200}, {"aircraft_code": "CR2", "model": "Бомбардье CRJ-200", "range": 2700}]
```

• Упражнение 8.2

Напишите запрос, выдающий список рейсов из демонстрационной базы в формате XML



The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL query:

```
1 SELECT xmleagg(  
2     xMLElement(  
3         name flight,  
4         xmlforest(  
5             flight_id AS flight_id,  
6             flight_no AS flight_no,  
7             scheduled_departure AS scheduled_departure,  
8             scheduled_arrival AS scheduled_arrival,  
9             departure_airport AS departure_airport,  
10            arrival_airport AS arrival_airport,  
11            status AS status  
12        )  
13     ) ORDER BY scheduled_departure  
14 )  
15 FROM (  
16     SELECT f.flight_id,  
17            f.flight_no,  
18            f.scheduled_departure,  
19            f.scheduled_arrival,  
20            f.departure_airport,  
21            f.arrival_airport,  
22            f.status  
23     FROM bookings_multilingual.flights_v f  
24     LIMIT 50  
25 ) AS lia_flights;
```

The Data Output pane shows the result in XML format:

```
1 3:00/scheduled_departure=scheduled_arrival=2017-07-16T10:30:00+03:00/scheduled_arrival=departure_airport=DME/departure_airport=arrival_airport=LED/arrival_airport=status=Arrived/flight=flight=flight_id=21/flight_id=flight_no=PG405/flight_no=sche
```

```

"<flight><flight_id>1</flight_id><flight_no>PG0405</flight_no><scheduled_departure>2017-07-
16T09:35:00+03:00</scheduled_departure><scheduled_arrival>2017-07-
16T10:30:00+03:00</scheduled_arrival><departure_airport>DME</departure_airport><arrival_airport>LED</
arrival_airport><status>Arrived</status>
</flight><flight><flight_id>21</flight_id><flight_no>PG0405</flight_no><scheduled_departure>2017-07-
19T09:35:00+03:00</scheduled_departure><scheduled_arrival>2017-07-
19T10:30:00+03:00</scheduled_arrival><departure_airport>DME</departure_airport><arrival_airport>LED</
arrival_airport><status>Arrived</status>
...

```

• Упражнение 8.3

Напишите запрос, выдающий заданное бронирование в формате JSON, включая все входящие в него билеты и перелеты для каждого из билетов.

The screenshot shows the pgAdmin 4 interface. On the left, a query is written to select a JSON object for a specific booking reference. The query uses a series of JOINs to combine data from the bookings, tickets, and boarding_passes tables. The right pane shows the 'Data Output' for a query that selects all records from the bookings_multilingual table, ordered by booking reference. The table has three columns: book_ref (PK) character (6), book_date timestamp with time zone, and total_amount numeric (10,2). The results show 17 rows of booking data.

```

{"booking_reference": "00000F", "booking_date": "2017-07-05T03:12:00+03:00", "total_amount": 265700.00, "tickets": [{"ticket_no": "0005435838975", "passenger": {"passenger_id": "1708 262537", "passenger_name": "ANNA ANTONOVA", "contact_data": {"email": "annaantonova-19021973@postgrespro.ru", "phone": "+70938049942"}}, "boarding_pass": {"flight_id": "5995", "boarding_no": "15", "seat_no": "10E"}}, {"ticket_no": "0005435838975", "passenger": {"passenger_id": "1708 262537", "passenger_name": "ANNA ANTONOVA", "contact_data": {"email": "annaantonova-19021973@postgrespro.ru", "phone": "+70938049942"}}, "boarding_pass": {"flight_id": "18058", "boarding_no": "6", "seat_no": "5C"}}]}

```

• Упражнение 8.4

Решите задачу, обратную предыдущей: получив бронирование в формате JSON, вставьте в таблицы демонстрационной базы данных соответствующие строки.

pgAdmin 4

demo/postgres@PostgreSQL 17 X

```
1 CREATE OR REPLACE FUNCTION insert_booking_from_json(booking_json JSON)
2 RETURNS TEXT AS $$
3 DECLARE
4     booking_ref TEXT;
5     booking_date TIMESTAMPTZ;
6     total_amount NUMERIC;
7     ticket_record JSON;
8     flight_id INT;
9     boarding_no INT;
10 BEGIN
11     -- Извлекаем основные данные бронирования
12     booking_ref := (booking_json->>'booking_reference');
13     booking_date := (booking_json->>'booking_date')::TIMESTAMPTZ;
14     total_amount := (booking_json->>'total_amount')::NUMERIC;
15
16     -- Проверяем обязательные поля
17     IF booking_ref IS NULL THEN
18         RETURN 'Error: booking_reference is required';
19     END IF;
20
21     -- Вставляем запись в таблицу bookings
22     BEGIN
23         INSERT INTO bookings_multilingual.bookings (book_ref, book_date, total_amount)
24         VALUES (booking_ref, booking_date, total_amount);
25     ON CONFLICT (book_ref) DO NOTHING;
26     EXCEPTION WHEN OTHERS THEN
27         RETURN 'Error inserting booking: ' || SQLERRM;
28     END;
29
30     -- Проверяем наличие билетов
31     IF booking_json->>'tickets' IS NULL THEN
32         RETURN 'Warning: No tickets in booking ' || booking_ref;
33     END IF;
34
35     -- Обработаем каждый билет в бронировании
36     FOR ticket_record IN SELECT * FROM json_array_elements(booking_json->'tickets')
37     LOOP
38         -- Проверяем обязательные поля билета
39         IF ticket_record->'ticket_no' IS NULL THEN
40             CONTINUE; -- Пропускаем билеты без номера
41         END IF;
42
43         -- Вставляем данные билета
44         BEGIN
45             INSERT INTO bookings_multilingual.tickets (
46
```

Query returned successfully in 48 msec.

Total rows: Query complete 00:00:00.040

CRLF Ln 96, Col 21

demo/postgres@PostgreSQL 17 X

```
1 SELECT insert_booking_from_json('
2 {
3     "booking_reference": "000000A",
4     "booking_date": "2025-05-16T22:41:00Z",
5     "total_amount": 1.00,
6     "tickets": [
7         {
8             "ticket_no": "00005432001002",
9             "passenger": {
10                 "passenger_id": "3988 776655",
11                 "passenger_name": "ALEX OMI",
12                 "contact_data": {"phone": "+73219876543"}
13             },
14             "boarding_pass": {
15                 "flight_id": "S4321",
16                 "boarding_no": "A",
17                 "seat_no": "24C"
18             }
19         }
20     ]
21 }
```

insert_booking_from_json test

Showing rows: 1 to 1 Page No: 1 of 1

Total rows: 1 Query complete 00:00:00.038

CRLF Ln 15, Col 30

demo/postgres@PostgreSQL 17 X

```
1 SELECT * FROM bookings_multilingual.bookings ORDER BY book_ref;
```

Showing rows: 1 to 1000 Page No: 1 of 263

book_ref	book_date	total_amount
[PK] character (6)	timestamp with time zone	numeric (15,2)
1 000000	2025-07-17 11:45:00+03	7600.00
2 000000A	2025-05-17 01:41:00+03	1.00
3 00000F	2017-07-05 01:12:00+03	265700.00
4 00000H	2025-07-17 11:45:00+03	7600.00
5 000012	2017-07-14 09:02:00+03	37000.00
6 000008	2017-08-15 14:27:00+03	18100.00
7 000181	2017-08-10 13:28:00+03	131800.00
8 000208	2017-08-07 21:40:00+03	29600.00
9 000208	2017-07-29 06:30:00+03	101500.00

Total rows: 262793 Query complete 00:00:00.166

CRLF Ln 1, Col 64

pgAdmin 4

demo/postgres@PostgreSQL 17 X

```
43 -- Вставляем данные билета
44 BEGIN
45     INSERT INTO bookings_multilingual.tickets (
46         ticket_no,
47         book_ref,
48         passenger_id,
49         passenger_name,
50         contact_data
51     ) VALUES (
52         ticket_record->'ticket_no',
53         booking_ref,
54         (ticket_record->'passenger')->'passenger_id',
55         (ticket_record->'passenger')->'passenger_name',
56         (ticket_record->'passenger')->'contact_data'
57     ) ON CONFLICT (ticket_no) DO NOTHING;
58     EXCEPTION WHEN OTHERS THEN
59         RAISE NOTICE 'Error inserting ticket %: %', ticket_record->'ticket_no', SQLERRM;
60     END;
61
62     -- Проверяем наличие данных о посадочном талоне
63     IF ticket_record->'boarding_pass' IS NOT NULL
64     AND ticket_record->'boarding_pass' <> 'null'
65     AND (ticket_record->'boarding_pass')->'flight_id' IS NOT NULL THEN
66
67         -- Соединяем информацию flight_id и boarding_no
68         BEGIN
69             flight_id_int := (ticket_record->'boarding_pass')->'flight_id'::INTEGER;
70             boarding_no_int := (ticket_record->'boarding_pass')->'boarding_no'::INTEGER;
71
72             -- Вставляем данные посадочного талона
73             INSERT INTO bookings.boarding_passes (
74                 ticket_no,
75                 flight_id,
76                 boarding_no,
77                 seat_no
78             ) VALUES (
79                 ticket_record->'ticket_no',
80                 flight_id_int,
81                 boarding_no_int,
82                 (ticket_record->'boarding_pass')->'seat_no'
83             ) ON CONFLICT (ticket_no, flight_id) DO NOTHING;
84             EXCEPTION WHEN OTHERS THEN
85                 RAISE NOTICE 'Error inserting boarding pass for ticket %: %',
86                     ticket_record->'ticket_no', SQLERRM;
87             END;
88         END;
89     END;
90
91 CREATE FUNCTION
```

Query returned successfully in 48 msec.

Total rows: Query complete 00:00:00.040

CRLF Ln 96, Col 21

demo/postgres@PostgreSQL 17 X

```
1 SELECT insert_booking_from_json('
2 {
3     "booking_reference": "000000A",
4     "booking_date": "2025-05-16T22:41:00Z",
5     "total_amount": 1.00,
6     "tickets": [
7         {
8             "ticket_no": "00005432001002",
9             "passenger": {
10                 "passenger_id": "3988 776655",
11                 "passenger_name": "ALEX OMI",
12                 "contact_data": {"phone": "+73219876543"}
13             },
14             "boarding_pass": {
15                 "flight_id": "S4321",
16                 "boarding_no": "A",
17                 "seat_no": "24C"
18             }
19         }
20     ]
21 }
```

insert_booking_from_json test

Showing rows: 1 to 1 Page No: 1 of 1

Total rows: 1 Query complete 00:00:00.038

CRLF Ln 15, Col 30

demo/postgres@PostgreSQL 17 X

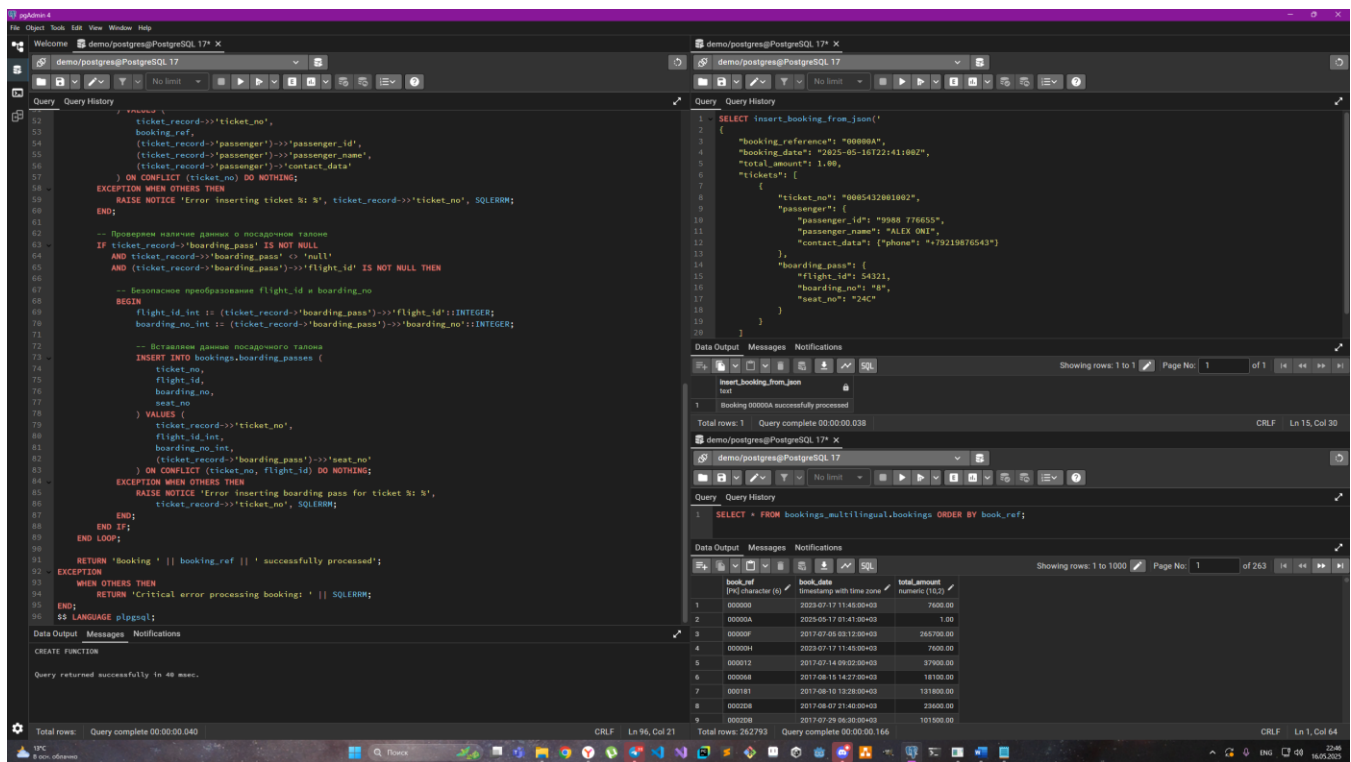
```
1 SELECT * FROM bookings_multilingual.bookings ORDER BY book_ref;
```

Showing rows: 1 to 1000 Page No: 1 of 263

book_ref	book_date	total_amount
[PK] character (6)	timestamp with time zone	numeric (15,2)
1 000000	2025-07-17 11:45:00+03	7600.00
2 000000A	2025-05-17 01:41:00+03	1.00
3 00000F	2017-07-05 01:12:00+03	265700.00
4 00000H	2025-07-17 11:45:00+03	7600.00
5 000012	2017-07-14 09:02:00+03	37000.00
6 000008	2017-08-15 14:27:00+03	18100.00
7 000181	2017-08-10 13:28:00+03	131800.00
8 000208	2017-08-07 21:40:00+03	29600.00
9 000208	2017-07-29 06:30:00+03	101500.00

Total rows: 262793 Query complete 00:00:00.166

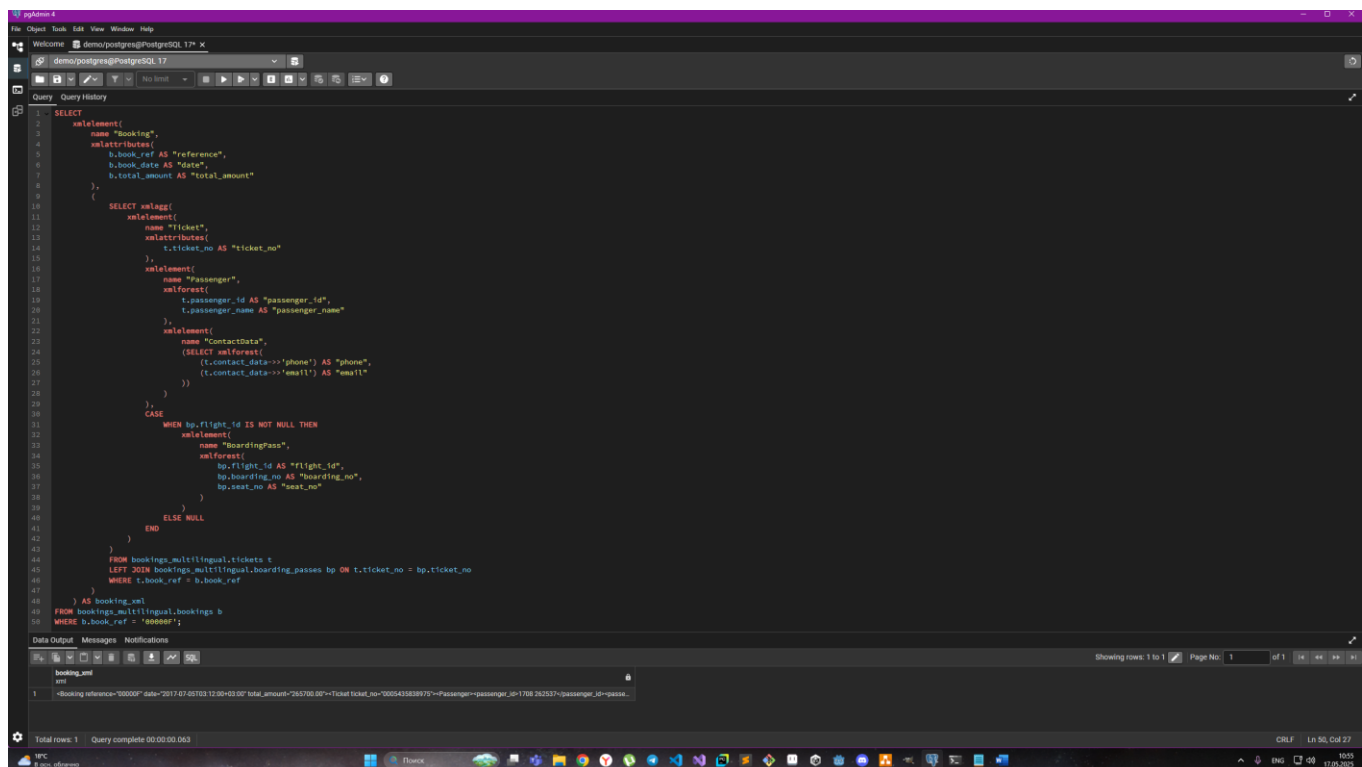
CRLF Ln 1, Col 64



• Упражнение 8.5

Выполните два предыдущих упражнения, используя формат XML вместо JSON

1. Повтор 8.3 с использованием xml



```

"<Booking reference=\""00000F\"" date=\""2017-07-05T03:12:00+03:00\"" total_amount=\""265700.00\"">
  <Ticket ticket_no=\""0005435838975\"">
    <Passenger>
      <passenger_id>1708 262537</passenger_id>
      <passenger_name>ANNA ANTONOVA</passenger_name>
      <ContactData>
        <phone>+70938049942</phone>
        <email>annaantonova-19021973@postgrespro.ru</email>
      </ContactData>
    </Passenger>
    <BoardingPass>
      <flight_id>5995</flight_id>
      <boarding_no>15</boarding_no>
      <seat_no>10E</seat_no>
    </BoardingPass>
  </Ticket>
  <Ticket ticket_no=\""0005435838975\"">
    <Passenger>
      <passenger_id>1708 262537</passenger_id>
      <passenger_name>ANNA ANTONOVA</passenger_name>
      <ContactData>
        <phone>+70938049942</phone>
        <email>annaantonova-19021973@postgrespro.ru</email>
      </ContactData>
    </Passenger>
    <BoardingPass>
      <flight_id>18058</flight_id>
      <boarding_no>6</boarding_no>
      <seat_no>5C</seat_no>
    </BoardingPass>
  </Ticket>
</Booking>"

```

2. Повтор 8.4 с использованием xml

```

1  -- Создаем временную таблицу для хранения XML-данных
2  CREATE TEMP TABLE xml_booking (
3    xml_data xml
4  );
5
6  -- Вставляем XML-данные в временную таблицу
7  INSERT INTO xml_booking (xml_data)
8  VALUES (
9    '
10    <booking>
11      <booking_reference>00000F</booking_reference>
12      <booking_date>2015-04-17T12:00:00+03:00</booking_date>
13      <total_amount>3.00</total_amount>
14      <ticket>
15        <ticket_no>00000000000015</ticket_no>
16        <passenger>
17          <passenger_id>11111 11111</passenger_id>
18          <passenger_name>LEX CORP</passenger_name>
19          <contact_data>
20            <email>"l_corp@postgrespro.ru">
21            <phone>"790909090909">
22          </contact_data>
23        </passenger>
24        <boarding_pass>
25          <flight_id></flight_id>
26          <boarding_no></boarding_no>
27          <seat_no></seat_no>
28        </boarding_pass>
29      </ticket>
30    </ticket>
31    <ticket>
32      <ticket_no>00000000000016</ticket_no>
33      <passenger>
34        <passenger_id>2222 22222</passenger_id>
35        <passenger_name>ALEXANDER LUTHOR</passenger_name>
36        <contact_data>
37          <email>"lor@examp1e.com">
38          <phone>"780808080808">
39        </contact_data>
40      </passenger>
41      <boarding_pass>
42        <flight_id></flight_id>
43        <boarding_no></boarding_no>
44        <seat_no></seat_no>
45      </boarding_pass>
46    </ticket>
47    <ticket>
48      <ticket_no>00000000000017</ticket_no>
49      <passenger>
50        <passenger_id>3333 33333</passenger_id>
51        <passenger_name>BRUS WAYNE</passenger_name>
52        <contact_data>
53          <email>"brus@examp1e.com">
54          <phone>"777777777777">
55        </contact_data>
56      </passenger>
57    </ticket>
58  </booking>
59  '
60  );

```

Query returned successfully in 32 msec.

```
pgAdmin 4
File Object Tools Edit View Window Help
demo/postgres@PostgreSQL 17 X
demo/postgres@PostgreSQL 17
Query Query History
47 <passenger_name>BRUS MAYNE</passenger_name>
48 <contact_data>
49   <email>"batman@example.com",
50   <phone>"*7777777777"</contact_data>
51 </passenger>
52 <boarding_pass>
53   <flight_id></flight_id>
54   <boarding_no></boarding_no>
55   <seat_no></seat_no>
56 </boarding_pass>
57 </ticket>
58 </bookings>
59 <ticket_no>8887432888888</ticket_no>
60 <passenger>
61   <passenger_id>4444 444444</passenger_id>
62   <passenger_name>CLARC KETN</passenger_name>
63   <contact_data>
64     <email>"superman@postgrespro.ru",
65     <phone>"*7666666666"</contact_data>
66 </passenger>
67 <boarding_pass>
68   <flight_id>448835</flight_id>
69   <boarding_no>99</boarding_no>
70   <seat_no>74</seat_no>
71 </boarding_pass>
72 </ticket>
73 </bookings>
74 );
75
76 -- Bctasnew gannow e rafnemy bookings
77 INSERT INTO bookings_multilingual.bookings (book_ref, book_date, total_amount)
78 SELECT
79   (xpath('/booking_reference/text()', x.xml_data)[1]::text AS book_ref,
80    to_timestamp(xpath('/booking_date/text()', x.xml_data)[1]::text,
81      'YYYY-MM-DD"Th24Mi:SS+00:00') AS book_date,
82   (xpath('/total_amount/text()', x.xml_data)[1]::text::numeric(18, 2) AS
83    total_amount
84  FROM xml_bookings;
85
86 -- Bctasnew gannow e rafnemy tickets
87 INSERT INTO bookings_multilingual.tickets (ticket_no, book_ref, passenger_id, passenger_name, contact_data)
88 SELECT
89   t.ticket_no,
90   (xpath('/booking_reference/text()', x.xml_data)[1]::text AS book_ref,
91    t.passenger_id,
92    t.passenger_name,
93    t.contact_data
94  FROM xml_bookings x,
95  LATERAL (
96    SELECT
97      (xpath('/ticket/ticket_no/text()', x.xml_data)[1]::text AS ticket_no,
98       (xpath('/ticket/passenger/passenger_id/text()', x.xml_data)[1]::text AS passenger_id,
99       (xpath('/ticket/passenger/passenger_name/text()', x.xml_data)[1]::text AS passenger_name,
100      (xpath('/ticket/passenger/contact_data/text()', x.xml_data)[1]::text AS contact_data
101    ) t;
102
103 -- Bctasnew gannow e rafnemy boarding_passes
104 INSERT INTO bookings_multilingual.boarding_passes (ticket_no, flight_id, boarding_no, seat_no)
105 SELECT
106   b.ticket_no,
107   COALESCE(b.flight_id, NULL) AS flight_id,
108   COALESCE(b.boarding_no, NULL) AS boarding_no,
109   b.seat_no
110  FROM xml_bookings x,
111  LATERAL (
112    SELECT
113      unnest(xpath('/ticket/ticket_no/text()', x.xml_data)[1]::text AS ticket_no,
114      unnest(xpath('/ticket/boarding_pass/flight_id/text()', x.xml_data)[1]::text AS flight_id,
115      unnest(xpath('/ticket/boarding_pass/boarding_no/text()', x.xml_data)[1]::text AS boarding_no,
116      unnest(xpath('/ticket/boarding_pass/seat_no/text()', x.xml_data)[1]::text AS seat_no
117    ) b;
118  WHERE b.flight_id IS NOT NULL OR b.boarding_no IS NOT NULL OR b.seat_no IS NOT NULL;
119
120 Data Output Messages Notifications
121 INSERT 0 1
122
123 Query returned successfully in 52 msec.
124
125 Total rows: Query complete 00:00:00.052
```

```
pgAdmin 4
File Object Tools Edit View Window Help
demo/postgres@PostgreSQL 17 X
demo/postgres@PostgreSQL 17
Query Query History
60 <boarding_no>99</boarding_no>
61 <seat_no>74</seat_no>
62 </boarding_pass>
63 </ticket>
64 </bookings>
65 );
66
67 -- Bctasnew gannow e rafnemy bookings
68 INSERT INTO bookings_multilingual.bookings (book_ref, book_date, total_amount)
69 SELECT
70   (xpath('/booking_reference/text()', x.xml_data)[1]::text AS book_ref,
71    to_timestamp(xpath('/booking_date/text()', x.xml_data)[1]::text,
72      'YYYY-MM-DD"Th24Mi:SS+00:00') AS book_date,
73   (xpath('/total_amount/text()', x.xml_data)[1]::text::numeric(18, 2) AS
74    total_amount
75  FROM xml_bookings;
76
77 -- Bctasnew gannow e rafnemy tickets
78 INSERT INTO bookings_multilingual.tickets (ticket_no, book_ref, passenger_id, passenger_name, contact_data)
79 SELECT
80   t.ticket_no,
81   (xpath('/booking_reference/text()', x.xml_data)[1]::text AS book_ref,
82    t.passenger_id,
83    t.passenger_name,
84    t.contact_data
85  FROM xml_bookings x,
86  LATERAL (
87    SELECT
88      unnest(xpath('/ticket/ticket_no/text()', x.xml_data)[1]::text AS ticket_no,
89      unnest(xpath('/ticket/passenger/passenger_id/text()', x.xml_data)[1]::text AS passenger_id,
90      unnest(xpath('/ticket/passenger/passenger_name/text()', x.xml_data)[1]::text AS passenger_name,
91      unnest(xpath('/ticket/passenger/contact_data/text()', x.xml_data)[1]::text AS contact_data
92    ) t;
93
94 -- Bctasnew gannow e rafnemy boarding_passes
95 INSERT INTO bookings_multilingual.boarding_passes (ticket_no, flight_id, boarding_no, seat_no)
96 SELECT
97   b.ticket_no,
98   COALESCE(b.flight_id, NULL) AS flight_id,
99   COALESCE(b.boarding_no, NULL) AS boarding_no,
100   b.seat_no
101  FROM xml_bookings x,
102  LATERAL (
103    SELECT
104      unnest(xpath('/ticket/ticket_no/text()', x.xml_data)[1]::text AS ticket_no,
105      unnest(xpath('/ticket/boarding_pass/flight_id/text()', x.xml_data)[1]::text AS flight_id,
106      unnest(xpath('/ticket/boarding_pass/boarding_no/text()', x.xml_data)[1]::text AS boarding_no,
107      unnest(xpath('/ticket/boarding_pass/seat_no/text()', x.xml_data)[1]::text AS seat_no
108    ) b;
109  WHERE b.flight_id IS NOT NULL OR b.boarding_no IS NOT NULL OR b.seat_no IS NOT NULL;
110
111 Data Output Messages Notifications
112 INSERT 0 1
113
114 Query returned successfully in 52 msec.
115
116 Total rows: Query complete 00:00:00.052
```


The screenshot displays three separate SQL query windows in pgAdmin 4, each connected to a 'demo/postgres@PostgreSQL 17' database. The first window on the left shows a query: `select * from bookings_multilingual,bookings order by book_ref;` with results showing columns `book_ref`, `book_date`, and `total_amount`. The middle window shows a query: `select * from bookings_multilingual,tickets order by book_ref;` with results showing columns `ticket_no`, `book_ref`, and `passenger_name`. The third window on the right shows a query: `select * from bookings_multilingual,boarding_passes order by ticket_no;` with results showing columns `ticket_no`, `flight_id`, `boarding_no`, and `seat_no`.

• Упражнение 8.6

Создайте триггер, реализующий правило целостности в демонстрационной базе: рейсы могут совершать только те типы самолетов, максимальная дальность полета которых превышает расстояние между аэропортами. Для расчета расстояния воспользуйтесь расширением `earthdistance`

The screenshot shows a single SQL query window in pgAdmin 4 with the following code:

```

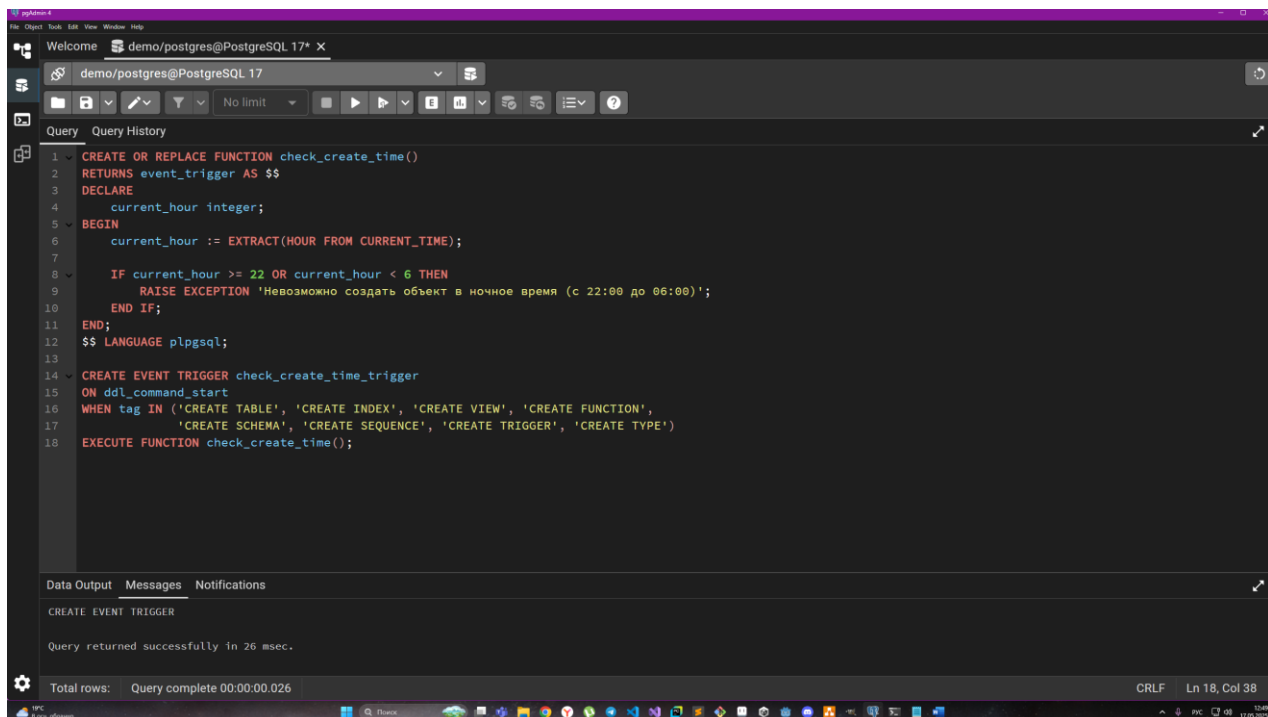
1 CREATE EXTENSION IF NOT EXISTS cube;
2 CREATE EXTENSION IF NOT EXISTS earthdistance;
3
4 CREATE OR REPLACE FUNCTION check_aircraft_range()
5 RETURNS TRIGGER AS $$
6 DECLARE
7     departure_coords point;
8     arrival_coords point;
9     aircraft_range integer;
10    distance float;
11 BEGIN
12     SELECT coordinates INTO departure_coords FROM bookings_multilingual.airports WHERE airport_code = NEW.departure_airport;
13     SELECT coordinates INTO arrival_coords FROM bookings_multilingual.airports WHERE airport_code = NEW.arrival_airport;
14     SELECT range INTO aircraft_range FROM bookings_multilingual.aircrafts WHERE aircraft_code = NEW.aircraft_code;
15     SELECT earth_distance(ll_to_earth(departure_coords[1], departure_coords[2]), ll_to_earth(arrival_coords[1], arrival_coords[2])) INTO distance;
16
17     IF aircraft_range < distance THEN
18         RAISE EXCEPTION 'дальность полета судна не позволяет осуществить рейс';
19     END IF;
20     RETURN NEW;
21 END;
22 $$ LANGUAGE plpgsql;
23
24 CREATE TRIGGER check_aircraft_range_trigger
25 BEFORE INSERT OR UPDATE ON bookings_multilingual.flights_v
26 FOR EACH ROW
27 EXECUTE FUNCTION check_aircraft_range();

```

The bottom of the window shows the query execution status: "Query returned successfully in 75 msec." and "Total rows: Query complete 00:00:00.075".

• Упражнение 8.7

Создайте в базе данных триггер, который не позволит выполнять операторы CREATE в ночное время.



```
1 CREATE OR REPLACE FUNCTION check_create_time()
2 RETURNS event_trigger AS $$
3 DECLARE
4     current_hour integer;
5 BEGIN
6     current_hour := EXTRACT(HOUR FROM CURRENT_TIME);
7
8     IF current_hour >= 22 OR current_hour < 6 THEN
9         RAISE EXCEPTION 'Невозможно создать объект в ночное время (с 22:00 до 06:00)';
10    END IF;
11 END;
12 $$ LANGUAGE plpgsql;
13
14 CREATE EVENT TRIGGER check_create_time_trigger
15 ON ddl_command_start
16 WHEN tag IN ('CREATE TABLE', 'CREATE INDEX', 'CREATE VIEW', 'CREATE FUNCTION',
17             'CREATE SCHEMA', 'CREATE SEQUENCE', 'CREATE TRIGGER', 'CREATE TYPE')
18 EXECUTE FUNCTION check_create_time();
```

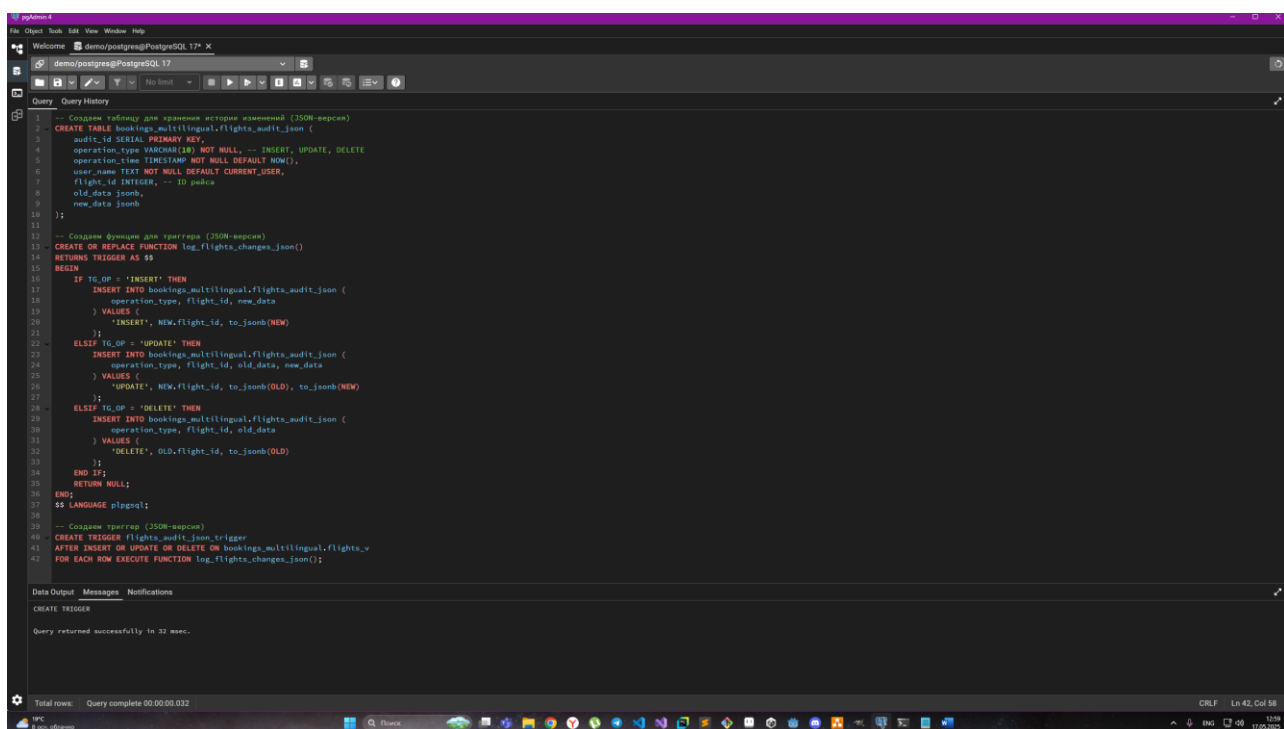
CREATE EVENT TRIGGER

Query returned successfully in 26 msec.

Total rows: Query complete 00:00:00.026

• Упражнение 8.8

Создайте в демонстрационной базе вспомогательную таблицу и триггеры для аудита изменений рейсов. Изменения можно записывать в таблицу с тем же набором полей, а можно — в один JSON-столбец (что позволит избежать проблем при изменении структуры таблицы).



```
1 -- Создаем таблицу для хранения истории изменений (JSON-версия)
2 CREATE TABLE bookings_multilingual.flights_audit_json (
3     audit_id SERIAL PRIMARY KEY,
4     operation_type VARCHAR(16) NOT NULL, -- INSERT, UPDATE, DELETE
5     operation_time TIMESTAMPTZ NOT NULL DEFAULT NOW(),
6     user_name TEXT NOT NULL DEFAULT CURRENT_USER,
7     flight_id INTEGER, -- ID полета
8     old_data jsonb,
9     new_data jsonb
10 );
11
12 -- Создаем функцию для триггера (JSON-версия)
13 CREATE OR REPLACE FUNCTION log_flights_changes_json()
14 RETURNS TRIGGER AS $$
15 BEGIN
16     IF TG_OP = 'INSERT' THEN
17         INSERT INTO bookings_multilingual.flights_audit_json (
18             operation_type, flight_id, new_data
19         ) VALUES (
20             'INSERT', NEW.flight_id, to_jsonb(NEW)
21         );
22     ELSEIF TG_OP = 'UPDATE' THEN
23         INSERT INTO bookings_multilingual.flights_audit_json (
24             operation_type, flight_id, old_data, new_data
25         ) VALUES (
26             'UPDATE', NEW.flight_id, to_jsonb(OLD), to_jsonb(NEW)
27         );
28     ELSEIF TG_OP = 'DELETE' THEN
29         INSERT INTO bookings_multilingual.flights_audit_json (
30             operation_type, flight_id, old_data
31         ) VALUES (
32             'DELETE', OLD.flight_id, to_jsonb(OLD)
33         );
34     END IF;
35     RETURN NULL;
36 END;
37 $$ LANGUAGE plpgsql;
38
39 -- Создаем триггер (JSON-версия)
40 CREATE TRIGGER flights_audit_json_trigger
41 AFTER INSERT OR UPDATE OR DELETE ON bookings_multilingual.flights_v
42 FOR EACH ROW EXECUTE FUNCTION log_flights_changes_json();
```

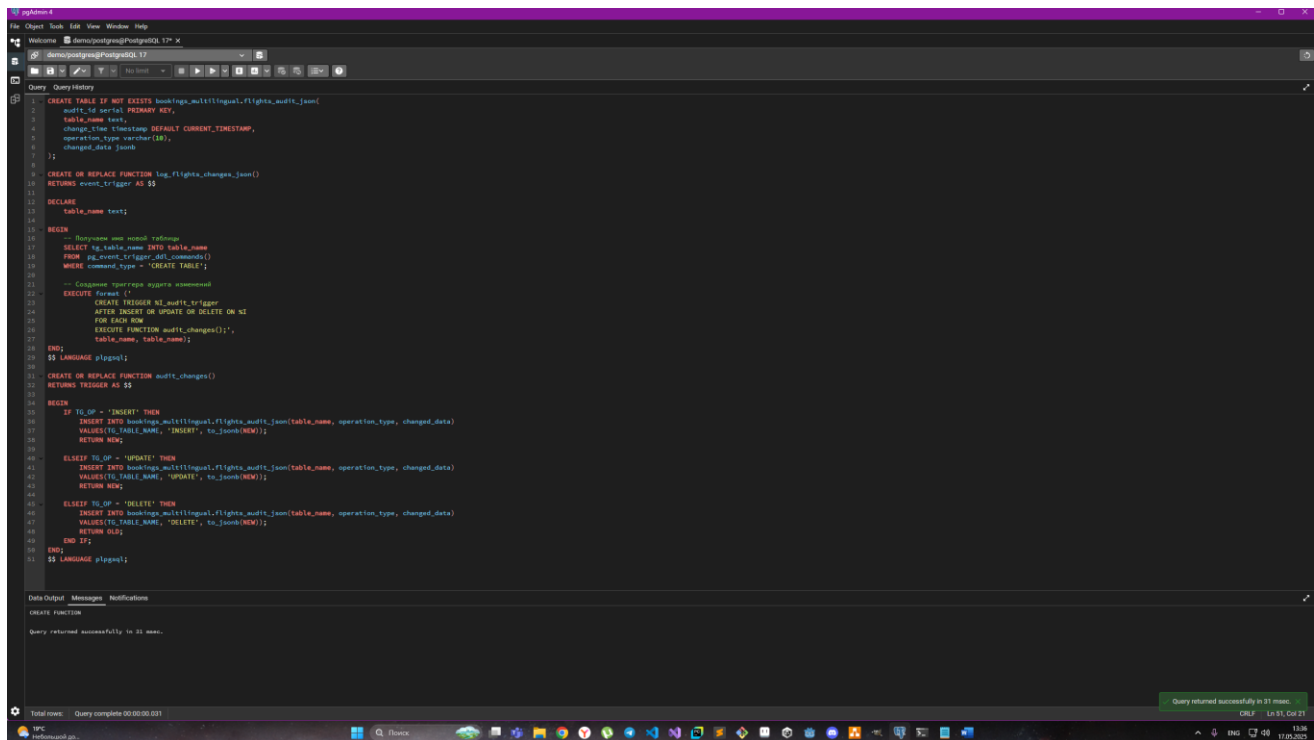
CREATE TRIGGER

Query returned successfully in 32 msec.

Total rows: Query complete 00:00:00.032

• Упражнение 8.9

Создайте в демонстрационной базе событийный триггер, автоматически создающий для новых таблиц обычные триггеры для аудита изменений в этих таблицах.



```
1 CREATE TABLE IF NOT EXISTS bookings_multilingual.flights_audit_json(
2     audit_id serial PRIMARY KEY,
3     table_name text,
4     change_time timestamp DEFAULT CURRENT_TIMESTAMP,
5     operation_type varchar(10),
6     changed_data jsonb
7 );
8
9 CREATE OR REPLACE FUNCTION log_flights_changes_json()
10 RETURNS event_trigger AS $$
11
12 DECLARE
13     table_name text;
14
15 BEGIN
16     -- Remove one record
17     SELECT * FROM table_name INTO table_name
18     FROM pg_event_trigger_modify_commands()
19     WHERE command_type = 'CREATE TABLE';
20
21     -- Create one record
22     EXECUTE format (
23         CREATE TRIGGER %s_audit_trigger
24         AFTER INSERT OR UPDATE OR DELETE ON %s
25         FOR EACH ROW
26         EXECUTE FUNCTION audit_changes();',
27         table_name, table_name);
28
29 END;
30
31 $$ LANGUAGE plpgsql;
32
33 CREATE OR REPLACE FUNCTION audit_changes()
34 RETURNS trigger AS $$
35
36 BEGIN
37     IF TG_OP = 'INSERT' THEN
38         INSERT INTO bookings_multilingual.flights_audit_json(table_name, operation_type, changed_data)
39         VALUES(TG_TABLE_NAME, 'INSERT', to_jsonb(NEW));
40         RETURN NEW;
41     ELSEIF TG_OP = 'UPDATE' THEN
42         INSERT INTO bookings_multilingual.flights_audit_json(table_name, operation_type, changed_data)
43         VALUES(TG_TABLE_NAME, 'UPDATE', to_jsonb(NEW));
44         RETURN NEW;
45     ELSEIF TG_OP = 'DELETE' THEN
46         INSERT INTO bookings_multilingual.flights_audit_json(table_name, operation_type, changed_data)
47         VALUES(TG_TABLE_NAME, 'DELETE', to_jsonb(OLD));
48         RETURN OLD;
49     END IF;
50
51 END;
52
53 $$ LANGUAGE plpgsql;
```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 31 msec.

Total rows: Query complete 00:00:00.001

Query returned successfully in 31 msec.

Ответы на контрольные вопросы

1. Для чего используется триггер? Приведите примеры.

Триггер — это специальная процедура в базе данных, которая автоматически выполняется при наступлении определенного события, такого как вставка, обновление или удаление данных. Триггеры используются для обеспечения целостности данных, автоматизации задач и реализации бизнес-правил.

Примеры использования триггеров:

Аудит изменений: Триггер может записывать изменения в таблице в отдельную таблицу аудита, чтобы отслеживать, кто и когда изменил данные.

Валидация данных: Триггер может проверять данные перед вставкой или обновлением, чтобы убедиться, что они соответствуют определенным критериям (например, проверка диапазона значений).

Автоматическое обновление: Триггер может автоматически обновлять связанные записи в других таблицах при изменении данных (например, обновление статуса заказа при изменении статуса его позиций).

2. Как вывести список в формате JSON?

Чтобы вывести список данных в формате JSON в PostgreSQL, можно использовать функцию `json_agg()` для агрегирования строк в массив JSON.

Например:

```
SELECT json_agg(row_to_json(t))
FROM (SELECT flight_id, departure_airport, arrival_airport
FROM new_schema.flights_v) t;
```

Этот запрос создаст JSON-массив, содержащий объекты, представляющие строки из таблицы `flights_v`

3. Как вывести список в формате XML?

Для вывода данных в формате XML в PostgreSQL можно использовать функцию `xmlagg()` для агрегирования строк в XML.

Например:

```
SELECT xmlagg(x)
FROM (
    SELECT xmlelement(name flight, xmlattributes(flight_id AS
    id), xmlforest(departure_airport AS departure,
    arrival_airport AS arrival) ) AS x FROM
    new_schema.flights_v ) AS subquery;
```

Этот запрос создаст XML-документ, содержащий элементы с атрибутами и подэлементами для каждого рейса.

4. Что такое коллекции?

Коллекции в контексте баз данных и программирования — это структуры данных, которые позволяют хранить и управлять множеством элементов. В PostgreSQL коллекции могут включать массивы, JSON и XML. Коллекции позволяют удобно работать с группами данных, обеспечивая возможность их агрегации, сортировки и фильтрации

5. Что такое функция? Приведите примеры.

Функция — это блок кода, который выполняет определенную задачу и может возвращать значение. Функции могут принимать параметры и использовать их для выполнения операций. В PostgreSQL функции могут быть написаны на различных языках, включая PL/pgSQL, SQL и другие.

Примеры функций:

Функция для вычисления суммы:

```
CREATE OR REPLACE FUNCTION calculate_sum(a INTEGER, b
INTEGER) RETURNS INTEGER AS $$
BEGIN
    RETURN a + b;
END;
$$ LANGUAGE plpgsql;
```